

# **Semantic XML Tagging of Domain-Specific Text Archives: A Knowledge Discovery Approach**

## **Dissertation**

zur Erlangung des akademischen Grades

## **Doktoringenieur (Dr.-Ing.)**

angenommen durch die Fakultät für Informatik  
der Otto-von-Guericke-Universität Magdeburg

von Diplom-Kaufmann Peter Karsten Winkler,  
geboren am 1. Oktober 1971 in Berlin

Gutachterinnen und Gutachter:

Prof. Dr. Myra Spiliopoulou

Prof. Dr. Gunter Saake

Prof. Dr. Stefan Conrad

Ort und Datum des Promotionskolloquiums:

Magdeburg, 22. Januar 2009

Karsten Winkler. Semantic XML Tagging of Domain-Specific Text Archives: A Knowledge Discovery Approach. Dissertation, Faculty of Computer Science, Otto von Guericke University Magdeburg, Magdeburg, Germany, January 2009.

# Contents

<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Algorithms</b>	<b>xi</b>
<b>Abstract</b>	<b>xiii</b>
<b>Zusammenfassung</b>	<b>xv</b>
<b>Acknowledgments</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 The Abundance of Text . . . . .	1
1.2 Defining Semantic XML Markup . . . . .	3
1.3 Benefits of Semantic XML Markup . . . . .	9
1.4 Research Questions . . . . .	12
1.5 Research Methodology . . . . .	14
1.6 Outline . . . . .	16
<b>2 Literature Review</b>	<b>19</b>
2.1 Storage, Retrieval, and Analysis of Textual Data . . . . .	19
2.1.1 Knowledge Discovery in Textual Databases . . . . .	19
2.1.2 Information Storage and Retrieval . . . . .	23
2.1.3 Information Extraction . . . . .	25
2.2 Discovering Concepts in Textual Data . . . . .	26
2.2.1 Topic Discovery in Text Documents . . . . .	27
2.2.2 Extracting Relational Tuples from Text . . . . .	31
2.2.3 Learning Taxonomies, Thesauri, and Ontologies . . . . .	35
2.3 Semantic Annotation of Text Documents . . . . .	39
2.3.1 Manual Semantic Text Annotation . . . . .	40
2.3.2 Semi-Automated Semantic Text Annotation . . . . .	43
2.3.3 Automated Semantic Text Annotation . . . . .	47
2.4 Schema Discovery in Marked-Up Text Documents . . . . .	50
2.5 Summary . . . . .	53

<b>3</b>	<b>DIAsDEM Framework</b>	<b>57</b>
3.1	Terminology . . . . .	57
3.2	Objectives and Overview . . . . .	62
3.3	Knowledge Discovery Phase . . . . .	65
3.4	Knowledge Application Phase . . . . .	67
3.5	Summary . . . . .	68
<b>4</b>	<b>DIAsDEM Knowledge Discovery Process</b>	<b>71</b>
4.1	Terminology . . . . .	71
4.2	Pre-Processing of Text Documents . . . . .	75
4.2.1	Creating and Tokenizing Text Units . . . . .	75
4.2.2	Extracting Named Entities . . . . .	80
4.2.3	Lemmatizing Words and Word Sense Disambiguation . . . . .	83
4.2.4	Establishing a Controlled Vocabulary . . . . .	86
4.2.5	Mapping Text Units onto Text Unit Vectors . . . . .	94
4.3	Clustering of Text Unit Vectors . . . . .	99
4.3.1	Clustering Textual Data: An Overview . . . . .	99
4.3.2	Selecting a Clustering Algorithm . . . . .	107
4.3.3	Ranking Clusters of Text Unit Vectors . . . . .	115
4.3.4	Iterative Clustering of Text Unit Vectors . . . . .	124
4.4	Post-Processing of Discovered Patterns . . . . .	132
4.4.1	Recommending Semantic Cluster Labels . . . . .	133
4.4.2	Establishing a Concept-Based XML DTD . . . . .	138
4.4.3	Semantic XML Tagging of Text Documents . . . . .	143
4.5	Bridging Knowledge Discovery and Knowledge Application . . . . .	148
4.6	Process Automation vs. Expert Involvement . . . . .	152
4.7	Summary . . . . .	154
<b>5</b>	<b>DIAsDEM Workbench</b>	<b>157</b>
5.1	Key Characteristics and Architecture . . . . .	157
5.2	Overview of Core Tasks . . . . .	160
5.2.1	Pre-Processing of Text Documents . . . . .	161
5.2.2	Iterative Clustering of Text Unit Vectors . . . . .	166
5.2.3	Post-Processing of Discovered Patterns . . . . .	169
5.3	Summary . . . . .	171
<b>6</b>	<b>Experimental Evaluation</b>	<b>173</b>
6.1	Assessing the Quality of Semantic XML Markup . . . . .	173
6.1.1	Quality Criteria for Semantic XML Markup . . . . .	174
6.1.2	Extending DIAsDEM Workbench . . . . .	179
6.2	Real-World Applications of the DIAsDEM Framework . . . . .	180
6.2.1	Semantic XML Markup for Competitive Intelligence . . . . .	180
6.2.2	German Commercial Register Entries . . . . .	181

6.2.3	News about U.S. Mergers and Acquisitions . . . . .	194
6.3	Summary . . . . .	206
<b>7</b>	<b>Conclusions</b>	<b>209</b>
7.1	Summary and Contribution . . . . .	209
7.2	Future Research . . . . .	212
7.2.1	Structuring the Concept-Based XML DTD . . . . .	212
7.2.2	Temporal Aspects of Discovered Knowledge . . . . .	213
7.2.3	Towards Automated Knowledge Discovery . . . . .	214
7.3	Concluding Remarks . . . . .	215
<b>A</b>	<b>Contents of the Supplementary Web Site</b>	<b>217</b>
<b>B</b>	<b>Specifications of Abstract Data Types</b>	<b>219</b>
B.1	ADT Notation, Primitive Data Types, and Arrays . . . . .	219
B.2	ADT for Strings . . . . .	220
B.3	ADT for Vectors . . . . .	220
B.4	ADT for Text Documents . . . . .	220
B.5	ADT for Text Archives . . . . .	221
B.6	ADT for Text Units . . . . .	221
B.7	ADT for Text Unit Layers . . . . .	222
B.8	ADT for Concepts . . . . .	222
B.9	ADT for Sets of Concepts . . . . .	223
B.10	ADT for Named Entity Types . . . . .	223
B.11	ADT for Sets of Named Entity Types . . . . .	224
B.12	ADT for Named Entities . . . . .	224
B.13	ADT for Sets of Named Entities . . . . .	225
B.14	ADT for Semantically Marked-Up Text Units . . . . .	225
B.15	ADT for Semantically Marked-Up Text Unit Layers . . . . .	226
B.16	ADT for Semantically Marked-Up Text Documents . . . . .	226
B.17	ADT for Semantically Marked-Up Text Archives . . . . .	227
B.18	ADT for Conceptual Document Structures . . . . .	227
B.19	ADT for KDT Algorithms . . . . .	228
B.20	ADT for KDT Process Flows . . . . .	229
B.21	ADT for Tokens . . . . .	229
B.22	ADT for Tokenized Text Units . . . . .	229
B.23	ADT for Intermediate Named Entities . . . . .	230
B.24	ADT for Sets of Intermediate Named Entities . . . . .	231
B.25	ADT for Text Unit Vectors . . . . .	231
B.26	ADT for Intermediate Text Units . . . . .	232
B.27	ADT for Intermediate Text Unit Layers . . . . .	233
B.28	ADT for Intermediate Text Documents . . . . .	234
B.29	ADT for Intermediate Text Archives . . . . .	234

B.30 ADT for Controlled Vocabulary Terms . . . . .	235
B.31 ADT for Controlled Vocabularies . . . . .	235
B.32 ADT for Text Unit Clusters . . . . .	236
B.33 ADT for Text Unit Clusterings . . . . .	238
B.34 ADT for Descriptor Weighting Schemata . . . . .	239
B.35 ADT for Clustering Algorithms . . . . .	240
B.36 ADT for Cluster Quality Criteria . . . . .	241
B.37 ADT for Iteration Metadata . . . . .	241
B.38 ADT for KDT Process Metadata . . . . .	242
<b>C List of Relevant German Vocabulary</b>	<b>245</b>
<b>List of Abbreviations</b>	<b>251</b>
<b>Notation and List of Symbols</b>	<b>253</b>
<b>Bibliography</b>	<b>257</b>
<b>Erklärung</b>	<b>295</b>

# List of Figures

1.1	Taxonomy of Markup in Text Documents . . . . .	4
1.2	Triangle of Reference . . . . .	5
1.3	Systems Development Research Process and Specific Research Process . . .	15
2.1	Process of Knowledge Discovery in Textual Databases . . . . .	22
2.2	Fundamental and Complementary Research Areas . . . . .	54
3.1	Outline of the Two-Phase DIASDEM Framework . . . . .	64
3.2	Knowledge Discovery Process of the DIASDEM Framework . . . . .	65
3.3	Knowledge Application Process of the DIASDEM Framework . . . . .	67
4.1	Text Document $\check{t}_{E3}$ Decomposed into Three Distinct Text Unit Layers . . .	76
4.2	Iterative Clustering in the DIASDEM Knowledge Discovery Process . . .	126
4.3	Illustration of Pattern Discovery in Four Clustering Iterations . . . . .	128
4.4	Iterative Classification in the DIASDEM Knowledge Application Process .	150
4.5	Generic Process of Knowledge Discovery in Textual Databases . . . . .	155
5.1	Architectural Overview of DIASDEM WORKBENCH . . . . .	158
5.2	Screen Shot of DIASDEM WORKBENCH GUI CLIENT . . . . .	159
5.3	Screen Shot of the REPLACE NAMED ENTITIES 2.1 Task . . . . .	160
5.4	Screen Shot of the BATCH SCRIPT EDITOR Tool . . . . .	161
5.5	Screen Shot of the THESAURUS EDITOR Tool . . . . .	165
5.6	Screen Shot of the CLUSTER TEXT UNIT VECTORS (HYPKNOWSYS) Task . . . . .	167
5.7	Visualization of Text Unit Clustering Created by the MONITOR CLUSTER QUALITY 2.2 Task . . . . .	168
5.8	Visualization of Text Unit Cluster Created by the MONITOR CLUSTER QUALITY 2.2 Task . . . . .	169
5.9	Screen Shot of the DERIVE CONCEPTUAL DTD 2.2 Task . . . . .	170
6.1	Screen Shot of the TAGGING QUALITY EVALUATOR 2.2 Tool . . . . .	180



# List of Tables

1.1	Excerpt of XML Document Containing Semantically Marked-Up Text . . . .	8
2.1	Knowledge Discovery in Textual Databases: Tasks and Applications . . . .	21
4.1	Five Reuters News Items Used in Examples . . . . .	72
4.2	Text Document $\check{t}_{E1}$ Decomposed into Text Unit Layer $\check{r}_{E1}$ . . . . .	76
4.3	Tokenized Text Units after Tokenizing the Elements of Text Unit Layer $\check{r}_{E1}$	79
4.4	Excerpt from the Extended Named Entity Hierarchy and Exemplary Named Entities . . . . .	80
4.5	Processed Text Units of Intermediate Text Unit Layer $\bar{r}_{E1}$ after Extracting Named Entities . . . . .	82
4.6	Intermediate Named Entities Identified in Tokenized Text Units of Intermediate Text Unit Layer $\bar{r}_{E1}$ . . . . .	83
4.7	Processed Text Units of Intermediate Text Unit Layer $\bar{r}_{E1}$ after Lemmatization and Word Sense Disambiguation . . . . .	86
4.8	Excerpt of ISO-2788 Thesaurus for Text Documents $\check{t}_{E1}$ through $\check{t}_{E5}$ and Corresponding DIASDEM-Specific Controlled Vocabulary Terms . . . . .	92
4.9	Text Unit Descriptors of the Controlled Vocabulary $V_E$ and Weighting Components for Exemplary Text Documents $\check{t}_{E1}$ through $\check{t}_{E5}$ . . . . .	97
4.10	Text Unit Vectors of Intermediate Text Unit Layer $\bar{r}_{E1}$ . . . . .	97
4.11	Common Proximity Measures between Two Text Documents Represented by $m$ -Dimensional Property Vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ . . . . .	102
4.12	Five Relative Cluster Validity Indices . . . . .	106
4.13	Summary of Three Proposed Clustering Algorithms w.r.t. the Fulfillment of DIASDEM-Specific Selection Criteria . . . . .	110
4.14	Sentences Assigned to Qualitatively Acceptable Text Unit Cluster 4 in Iteration 1 . . . . .	116
4.15	Sentences Assigned to Qualitatively Unacceptable, Inhomogeneous Text Unit Cluster 7 in Iteration 1 . . . . .	117
4.16	Sentence Assigned to Qualitatively Unacceptable, Small Text Unit Cluster 0 in Iteration 1 . . . . .	117
4.17	Text Unit Descriptors Occurring in Qualitatively Acceptable Text Unit Cluster 4 in Iteration 1 . . . . .	123
4.18	Text Unit Descriptors Occurring in Qualitatively Unacceptable, Inhomogeneous Text Unit Cluster 7 in Iteration 1 . . . . .	123

4.19	Text Unit Clustering after Executing the Bisecting 8-Means in Iteration 1 .	124
4.20	Summary of Pattern Discovery in Four Clustering Iterations . . . . .	129
4.21	Sentences Assigned to Qualitatively Acceptable Text Unit Cluster 1 in Iteration 2 . . . . .	129
4.22	Sentences Assigned to Qualitatively Acceptable Text Unit Cluster 3 in Iteration 2 . . . . .	130
4.23	Sentences Assigned to Qualitatively Unacceptable, Inhomogeneous Text Unit Cluster 2 in Iteration 2 . . . . .	130
4.24	Summary of Semantic Cluster Labeling in Four Clustering Iterations . . .	136
4.25	Sentences Assigned to Qualitatively Acceptable Text Unit Cluster 6 in Iteration 2 . . . . .	137
4.26	Sentences Assigned to Qualitatively Acceptable Text Unit Cluster 5 in Iteration 3 . . . . .	137
4.27	Labels in Conceptual Document Structure <code>NewsItem</code> Describing the Exem- plary Text Archive . . . . .	140
4.28	Line-Numbered, Concept-Based XML Document Type Definition of the Conceptual Document Structure <code>NewsItem</code> . . . . .	141
4.29	Semantically Marked-Up Text Units of the Text Document $\check{t}_{E1}$ . . . . .	144
4.30	Line-Numbered, Semantically Marked-Up XML Document Created by Tag- ging the Text Document $\check{t}_{E1}$ . . . . .	145
4.31	Line-Numbered, Semantically Marked-Up XML Document Created by Tag- ging the Text Document $\check{t}_{E4}$ . . . . .	146
4.32	Summary of Domain Knowledge Recommended for Incorporation into the DIASDEM Knowledge Discovery Process . . . . .	153
6.1	Four Sentences Illustrating the Types of XML Tag Names w.r.t. Markup Quality . . . . .	176
6.2	German Commercial Register Entries and English Translations . . . . .	183
6.3	Overview of Text Archives Used in the Commercial Register Case Study .	184
6.4	Named Entity Types Extracted in the Commercial Register Case Study . .	186
6.5	Summary of Pattern Discovery in 12 Clustering Iterations in the Commer- cial Register Case Study . . . . .	188
6.6	Semantically Marked-Up XML Document Comprising the First Exemplary German Commercial Register Entry . . . . .	189
6.7	Semantically Marked-Up XML Document Comprising the Second Exem- plary German Commercial Register Entry . . . . .	190
6.8	Semantically Marked-Up XML Document Comprising the Third Exem- plary German Commercial Register Entry . . . . .	191
6.9	Results of the Semantic XML Markup Quality Assessment in the Commer- cial Register Case Study . . . . .	192
6.10	News Stories about U.S. Mergers and Acquisitions . . . . .	196
6.11	Overview of Text Archives Used in the Reuters News Case Study . . . . .	197

6.12	Named Entity Types Extracted in the Reuters News Case Study . . . . .	198
6.13	Summary of Pattern Discovery in Eight Clustering Iterations . . . . .	200
6.14	Semantically Marked-Up XML Document Comprising the First Exemplary Reuters News Item . . . . .	202
6.15	Semantically Marked-Up XML Document Comprising the Second Exem- plary Reuters News Item . . . . .	203
6.16	Semantically Marked-Up XML Document Comprising the Third Exem- plary Reuters News Item . . . . .	204
6.17	Results of the Semantic XML Markup Quality Assessment in the Reuters News Case Study . . . . .	205



# List of Algorithms

4.1	DecomposeAndTokenizeTextDocuments . . . . .	79
4.2	ExtractNamedEntities . . . . .	83
4.3	LemmatizeAndDisambiguateWords . . . . .	86
4.4	PerformClusteringIteration (Outline) . . . . .	132
4.5	PerformClusteringIteration (Complete) . . . . .	138
4.6	EstablishConceptualDocumentStructure . . . . .	142
4.7	CreateSemanticallyMarkedUpTextArchive . . . . .	143
4.8	CreateSemanticallyTaggedXmlDocuments . . . . .	147
4.9	Generic KDT Process Flow for the Knowledge Discovery Phase . . . . .	149
4.10	PerformClassificationIteration . . . . .	151
4.11	Generic KDT Process Flow for the Knowledge Application Phase . . . . .	151



# Abstract

Facing ever increasing volumes of computer-accessible textual data, semantic XML tagging of text archives creates value by providing embedded metadata. In this work, names of semantic XML tags explicitly convey informal metadata by concisely describing concepts that domain experts typically associate with marked-up text units, which represent structural document parts (e.g., sentences or paragraphs). Optionally, attributes of semantic XML tags make explicit named entities (e.g., names of companies) occurring in marked-up text units. The markup syntax is defined in a concept-based XML document type definition, and its meaning is informally specified in the DTD documentation.

Organizations benefit from exploiting semantic metadata in two broad ways. Firstly, semantic XML markup tends to improve the effectiveness and efficiency of information retrieval. Secondly, semantic XML markup facilitates applications that leverage semantic metadata in texts, such as document warehousing and information integration.

How to employ knowledge discovery techniques for transforming domain-specific text archives into semantically marked-up XML documents? This research objective is addressed by (i) establishing a conceptual framework for semantic XML tagging of domain-specific text archives, (ii) developing a research prototype that implements the entire framework, and (iii) evaluating the markup quality by processing real-world text archives.

In phase one of the two-phase DIASDEM<sup>1</sup> framework, an interactive knowledge discovery process discovers and semantically labels concepts occurring at the text unit level of plain texts, derives a concept-based XML document type definition based on identified, frequently occurring concepts and prevailing named entity types, as well as semantically tags text documents to enable quality assessment. In the second phase termed knowledge application, large volumes of thematically similar text documents are automatically tagged by utilizing the classification knowledge acquired in phase one.

Subsequent to extensive text pre-processing, the knowledge discovery process iteratively groups text units based on content similarity to discover both specific and general, as well as more and less frequently occurring concepts. In each iteration, a clustering algorithm is selected, parameterized, and executed. Framework-specific cluster quality criteria are employed to distinguish between qualitatively acceptable and unacceptable text unit clusters. Acceptable clusters reduce the input data of the next iteration. They are automatically labeled with default names, which are refined by domain experts. Ultimately, labels of acceptable clusters serve as names of semantic XML tags.

---

<sup>1</sup>The framework for semantic XML tagging, the knowledge discovery process, and the prototype system are named after the research project DIASDEM supported by Deutsche Forschungsgemeinschaft.



# Zusammenfassung

Angesichts der stetigen Zunahme elektronisch verfügbarer Texte generiert die semantische XML-Auszeichnung von Textarchiven Mehrwert durch direkte Einbettung von Metadaten. In dieser Arbeit bezeichnen Namen semantischer XML-Textmarken als explizite und informelle Metadaten sehr konzise Konzepte, die Experten des Fachgebiets typischerweise mit den ausgezeichneten strukturellen Texteinheiten (z.B. Sätze oder Absätze) assoziieren. Attribute von XML-Textmarken enthalten optional benannte Entitäten (z.B. Personennamen), die in ausgezeichneten Texteinheiten erwähnt werden. Die Syntax der Auszeichnung ist in der konzeptuellen XML-Dokumenttypdefinition festgelegt, während deren inhaltliche Bedeutung informell in der DTD-Dokumentation spezifiziert ist.

Organisationen profitieren von der Verwertung semantischer Metadaten in zweierlei Hinsicht. Die semantische XML-Auszeichnung zielt erstens auf eine Verbesserung der Effektivität und Effizienz von Information-Retrieval-Systemen ab. Zweitens werden Anwendungen unterstützt, die semantische Metadaten in Texten verwerten. Beispiele hierfür sind Applikationen des Document Warehousing und der Informationsintegration.

Sind Methoden der Wissensentdeckung zur Transformation fachspezifischer Textarchive in semantisch ausgezeichnete XML-Dokumente einsetzbar? Dieses Forschungsziel wird durch (i) Spezifikation eines konzeptionellen Bezugsrahmens für die semantische XML-Auszeichnung fachspezifischer Textarchive, (ii) Entwicklung eines den gesamten Bezugsrahmen unterstützenden Prototypen und (iii) Evaluation der Auszeichnungsqualität durch Verarbeitung real existierender Textarchive adressiert.

Die erste Phase des zweiphasigen DIASDEM<sup>2</sup> Bezugsrahmens ist ein interaktiver Wissensentdeckungsprozess, der auf Ebene der Texteinheiten Konzepte entdeckt und inhaltsbezogen benennt, eine konzeptuelle XML-Dokumenttypdefinition aus häufig auftretenden Konzepten und dominierenden benannten Entitäten ableitet sowie Textdokumente für die Qualitätskontrolle semantisch auszeichnet. In der als Wissensanwendung bezeichneten zweiten Phase werden Texte desselben Fachgebiets durch Anwendung des in der ersten Phase entdeckten Klassifikationswissens automatisch ausgezeichnet.

Nach umfangreicher Vorverarbeitung der Texte segmentiert der Wissensentdeckungsprozess Texteinheiten entsprechend ihrer inhaltlichen Ähnlichkeit iterativ, um sowohl spezielle und allgemeinere Konzepte als auch mehr und weniger häufig verwendete Konzepte zu entdecken. Dazu wird ein Clustering-Algorithmus in jeder Iteration ausgewählt, parametrisiert und ausgeführt. Bezugsrahmensspezifische Cluster-Qualitätskriterien wer-

---

<sup>2</sup>Der Bezugsrahmen für die semantische XML-Auszeichnung, der Wissensentdeckungsprozess und der Prototyp sind in Anlehnung an das von der Deutschen Forschungsgemeinschaft geförderte Forschungsprojekt DIASDEM benannt.

den angewendet, um qualitativ akzeptable von nicht akzeptablen Texteinheitssegmenten zu unterscheiden. Die akzeptablen Segmente reduzieren einerseits die Eingabedaten der nächsten Iteration und werden andererseits automatisch mit Namensvorschlägen etikettiert, deren Bezeichner anschließend von Experten des Fachgebiets verfeinert werden. Diese Bezeichner entsprechen letztlich den Namen der semantischen XML-Textmarken.

# Acknowledgments

The author's work was facilitated by Humboldt University Berlin, Leipzig Graduate School of Management, and Otto von Guericke University Magdeburg. In particular, the author wants to thank his adviser and mentor Myra Spiliopoulou for academic guidance and support throughout the doctoral studies. Special thanks go to Stefan Conrad and Gunter Saake for reviewing the dissertation. The author enjoyed working with Evguenia Altareva, Hans-Knud Arndt, Markus Banach, Steffan Baron, Marko Brunzel, Martin Christian, Henner Graubitz, Oliver Günther, Kerstin Kaldenhoff, Tamara Kurz, Pierfrancesco La Mura, Carsten Pohle, René Schult, Anja Schulz, Lutz Wille, and many other colleagues during his doctoral studies in Berlin, Leipzig, and Magdeburg.

The research project DIAsDEM was kindly supported by Deutsche Forschungsgemeinschaft (DFG) grant SP 572/4-1. It is gratefully acknowledged that both text archives of the first case study in Subsection 6.2.2 were provided for research purposes by Heins + Partner GmbH, Bielefeld, Germany. Furthermore, it is gratefully acknowledged that both text archives of the second case study in Subsection 6.2.3 were provided by Reuters Limited via the Reuters Corpus, Vol. 1, English Language, 1996-08-20 to 1997-08-19.



# 1 Introduction

*We are drowning in information but starved for knowledge.*  
—John Naisbitt (1982, p. 24)

How to add value to huge volumes of existing textual data? This work focuses on one aspect of value creation: the enhancement of text documents by means of semantic XML markup. Adopting an information systems perspective and taking a knowledge discovery approach, we propose a novel framework for transforming large archives of domain-specific text documents into semantically marked-up XML documents. To begin, we briefly elaborate on the observed abundance of textual data and associated problems in the next section. In Section 1.2, the pivotal term semantic XML markup is defined. We discuss benefits of semantic XML markup in Section 1.3 to demonstrate the relevance of this research. Subsequently, the research questions addressed in this work are raised in Section 1.4. Finally, we describe the adopted research methodology in Section 1.5 and outline the remaining chapters of this work in Section 1.6.

## 1.1 The Abundance of Text

Companies, non-profit organizations, as well as public authorities create, store, and update vast, continuously growing volumes of computer-accessible data. According to Lyman and Varian (2003), about five exabytes (i.e.,  $5 \cdot 10^{18}$  bytes) of new information were produced on print, film, magnetic, and optical storage media in 2002. Gantz et al. (2007) estimated that the amount of digital information created, captured, and replicated in 2006 was 161 exabytes. The authors expect an increase in the annually added digital information from 161 to 988 exabytes between 2006 and 2010. Steadily falling prices for storage capacity seem to be a major driving force behind this growth rate.

**Unstructured Textual Data** With respect to the degree of internal structure, database and information retrieval researchers typically classify this huge amount of data into three categories (e.g., Buneman et al., 1997; Halevy et al., 2003; Weiss et al., 2005, p. 2). Structured data, such as relational data, are constrained by a rigid, a priori fixed conceptual schema. In contrast, semi-structured data (e.g., HTML files) can be informally characterized as “schemaless” or “self-describing” (Abiteboul et al., 2000, p. 11). Unlike the first category, semi-structured data do not adhere to a rigid, explicitly specified schema. Nevertheless, they do exhibit some structure and may implicitly adhere to a loose schema (Wang and Liu, 2000, p. 353). Finally, unstructured data adhere neither to a loose nor to

a rigid schema definition. Textual content (cf. Feldman and Sanger, 2007), images, audio files, and video sequences are examples of unstructured data.

The written word has been an important means of human communication for centuries. According to Dörre et al. (2001, p. 465), unstructured data constitute at least 90% of the data centrally managed by corporate information technology departments. Plain texts and files, which can easily be transformed into texts using, for instance, optical character recognition techniques, appear by far to be the largest proportion of the data stored in corporate information systems. Sullivan (2001, p. 84) estimated that up to 80% of business information consists of unstructured text, including important free-form texts like archived business letters, e-mails, technical handbooks, annual reports, and project memos. In addition to in-house repositories, the World Wide Web is another tremendous resource of textual, business-related content (cf. Hackathorn, 1999, pp. 3–25). Both internal and public text archives are without doubt a major source of codified knowledge, which potentially serves as a basis for creating sustainable competitive advantages.

**Tackling Information Overload** Despite, or rather due to, this plethora of textual data, humans often desperately struggle to find relevant information when it is needed to fulfill their duties and responsibilities. This widely recognized phenomenon is termed information overload (cf. Edmunds and Morris, 2000; Hurst, 2007). According to Farhoomand and Drury (2002, p. 127), information overload can be characterized in two broad ways: Losee (1989) defined information overload as the receipt of more information than needed or desired to function effectively and to further individual or organizational goals. Alternatively, Schick et al. (1990) emphasized that information overload may occur if the information processing demand on individuals exceeds their respective processing capacity. Despite the lack of a universally agreed upon definition, studies indicate that managerial information overload is a widespread issue (cf. Reuters Business Information, 1996; Farhoomand and Drury, 2002). It often results in personal frustration, delays decisions, and causes economic losses due to the time wasted in trying to locate relevant information. Obviously, “more information is not always better” (Case, 2002, p. 289).

Due to the rapid growth of text repositories and the resulting demand for information overload reduction, information retrieval and knowledge discovery in textual databases have become active research fields in the past decade. Information retrieval denotes the activity of locating and presenting mainly textual information that is relevant to a user-specific information need expressed as a query (Korfhage, 1997, p. 324). The term knowledge discovery in textual databases was coined by Feldman and Dagan (1995) and denotes the application of knowledge discovery techniques to textual, instead of structured, data. Knowledge discovery in databases (KDD) aims at extracting previously unknown, statistically valid, and actionable knowledge from data (Frawley et al., 1991; Fayyad et al., 1996b). KDD combines various methods from statistics, machine learning, artificial intelligence, and database research in a unifying, process-centric framework.

Many large archives do not contain texts covering a variety of topics, but instead comprise domain-specific documents of relatively homogeneous content. Examples of this

kind of text collection are financial news, industry-specific new product announcements, quarterly reports to shareholders, and regulatory filings. Furthermore, content-based text classification techniques (cf. Sebastiani, 2002; Feldman and Sanger, 2007, pp. 64–81) allow for the extraction of documents featuring specific topics from heterogeneous archives. Announcements of stock repurchase programs could, for instance, be extracted from a continuous stream of general business news and stored in a separate text archive.

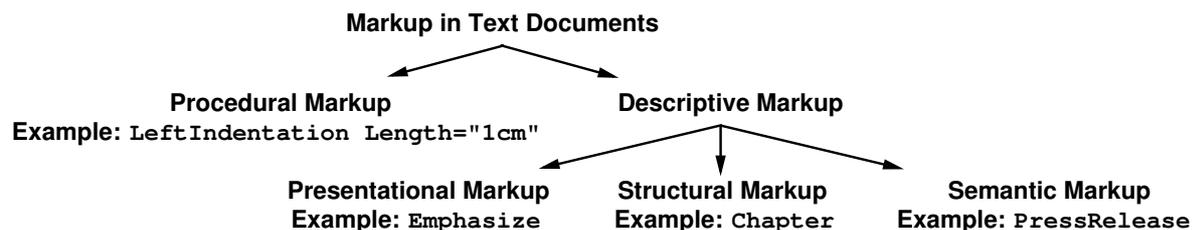
Although focusing on the same central topic, documents in a thematic collection tend to cover a variety of specialized, but nevertheless frequently recurring, subtopics at finer-grained text structures. Furthermore, documents in thematic archives often describe similar facts (e.g., resignation of the company’s chairman) about different persons or objects (e.g., resigning chairmen of specific companies) at the sentence or paragraph level. If domain-specific documents share an implicit thematic structure, their internal subject matters along with referenced persons or objects can be explicated by annotating the text accordingly. Electronic text is typically annotated by inserting so-called markup codes. For instance, all sentences mentioning the resignation of chairmen could be annotated with the content-descriptive markup code `ResignationOfChairman`. In addition, this code may be extended to reveal details of the actual resigning manager (e.g., `Person="Name: Barry Bankrupt"`). Markup adds value to archives by conveying additional, queryable information about the marked-up text passages. In particular, content-descriptive markup facilitates highly focused information retrieval and thus alleviates information overload. We propose a new framework for the fine-grained, content-descriptive markup of text documents that exploits the above-mentioned particularities of domain-specific text documents.

## 1.2 Defining Semantic XML Markup

Before introducing our notion of semantic text annotation, we concisely explain the term markup. Subsequently, key properties of the Extensible Markup Language (XML) are summarized because semantically marked-up texts are stored as XML documents within our framework. Finally, we define semantic XML markup for the scope of this work.

**Markup** In the pre-electronic publishing process, copy editors marked up manuscripts by adding handwritten directions for the typesetter regarding, for instance, the desired type fonts, their sizes, and spacings (Barron, 1989, p. 4). Taking a broader perspective, Coombs et al. (1987, p. 934) identified punctuational markup (e.g., ending sentences with full stops) and presentational markup (e.g., numbering pages) as the two main forms of traditional markup used directly by authors. In both cases, scribal markup always clarifies a written expression. Markup is not part of the textual content itself, but rather reveals additional information about the marked-up text.

As the automation of typesetting and layout advanced, the meaning of markup was extended as well. At that time, the term covered various kinds of markup codes, or markup



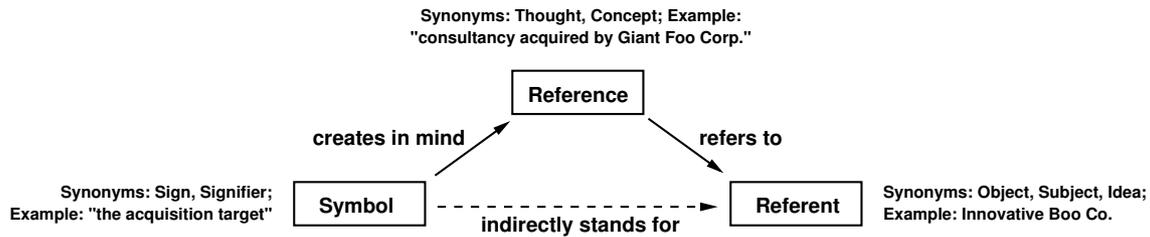
**Figure 1.1:** Taxonomy of Markup in Text Documents

languages, that were inserted into electronic text to trigger particular text processing functions (see Burnard, 1995, p. 42). Generalizing from the publishing domain, Burnard defined markup as “any means of making explicit an interpretation of text.” A markup language may hence be no more than a loosely accepted set of markup conventions for encoding a specific text collection. However, Burnard underlined that markup languages must specify the names of mandatory and optional markup elements, their syntactical distinction from the text to be marked up, and the meaning of markup elements.

The term metadata is often informally defined as data about data (e.g., see Geroimenko, 2004, p. 94). In particular, this term refers to data that describe specific properties of other data, such as their meaning, source, validity, or data type. Using a markup language, metadata are added to a text document by direct insertion of markup codes, or markup tags, at the respective places within the document (cf. Geroimenko, 2004, pp. 91 and 94). Markup thus conveys metadata about the tagged text passages, which are typically surrounded by a start and a matching end tag.

Although criticized, for example, by Renear (2000) and Piez (2001), markup languages are traditionally categorized into procedural markup and descriptive markup based on the characteristics of the metadata they convey (Goldfarb, 1981). The former exactly specifies system-specific processing functions to be performed on the marked-up text. In contrast, descriptive markup describes logical text elements by marking text passages with mnemonic tags that best characterize them. Instead of giving processing instructions, descriptive and therefore system-independent markup rather represents metadata about the logical structure of documents or the meaning of tagged content.

Maler and Andaloussi (1996, pp. 93–103) identified presentational, structural, and content-based text components that should be recognized during the development of specific markup languages, so-called document type definitions, for text documents. Analogously, we subdivide descriptive markup in text documents into presentational, structural, and semantic markup. Figure 1.1 depicts this proposed taxonomy of markup. Unlike procedural markup, presentational markup does not include specific processing commands for rendering content, but rather describes the desired visualization of marked-up text passages on an abstract and system-independent level. Structural markup reflects the logical structure of text documents by tagging text parts, such as headings, chapters, and paragraphs (cf. Tompa, 1989; Heeman, 1992). Finally, semantic markup characterizes the



**Figure 1.2:** Triangle of Reference (Modified from Ogden and Richards, 1994, p. 16)

surrounded text passages in terms of their actual content. Inserting semantic tags into documents thereby aims at explicating the meaning of marked-up text.

**Semantics** Semantics has been the subject of thought in philosophy, linguistics, psychology, artificial intelligence, and computer science for decades, if not centuries. Despite the generally accepted conception that semantics concerns the relationship between linguistic expressions and their meanings, there exists a long-standing academic dispute over the question of explicating this relation (Gärdenfors, 2000, p. 151). In linguistics, semantics denotes the study of meaning communicated through language (Saeed, 2003, p. 3). The difficulty in defining the meaning of meaning has resulted in various schools of linguistic thought on semantics (Goddard, 1998, pp. 6–11). In computer science, for example, the notion of data semantics is a matter of discussions among scholars (Sheth, 1996). In artificial intelligence, the representation of semantics in machine-processable data structures is a fundamental research area (Woods, 1975; Sowa, 1991; Fensel, 2004).

Creating and representing meaning via signs and symbols is the subject of semiotics (Chandler, 2002, pp. 2–7). Swiss linguist Ferdinand de Saussure (1857–1913) and American philosopher Charles S. Peirce (1839–1914) are regarded as the co-founders of this academic discipline. In the semiotic sense, signs might, for example, take the form of text, images, or sound. The semiotic approach to semantics has been adopted in, for instance, linguistics (Goddard, 1998, pp. 10–11), knowledge representation (Maedche et al., 2003, pp. 318–325), as well as library and information science (Mai, 2001). Instead of questioning ‘what meaning is’, Goddard emphasized, semioticians scrutinize ‘how meaning is conveyed’ in communication. First published in 1923, Ogden and Richards (1994, pp. 1–32) illustrated the semiotic relationship between a symbol, its mental reference, and the real-world referent by the triangle of reference depicted in Figure 1.2.

Whenever humans communicate about actually existing subjects, objects, or ideas (i.e., referents), they make use of symbols like gestures, words, or sentences. Even entire documents can be regarded as a single sign (Mai, 2001, p. 603). As the triangle of reference indicates, there is no direct semantic relationship between symbols and their referents. Instead, the meaning associated with a symbol is indirectly conveyed via a reference, or concept, created in the recipient’s mind. Humans mentally grasp an understanding of communication symbols by interpreting them in context and by considering available

background knowledge. Only concepts formed by interpretation are mentally linked to the respective referents in reality. The meaning of a text document is thus conveyed via concepts, which are first invoked in the reader's mind and thereafter associated with real-world entities and issues featured in the text. However, the mental process of concept formation is a matter under academic discussion (Staab, 2002, p. 85).

**Semantic Markup** Since we adopt the semiotic approach to meaning, semantic markup reflects concepts that adequately educated domain experts typically associate with tagged text passages. Resembling thematic markup (cf. Bayerl et al., 2003a), semantic markup provides a short, concept-based summary of topics, issues, and subject matters covered in tagged content. Abstract descriptions, as Woods (1991, p. 48) noted, are a useful way of representing concepts. Our notion of semantic markup is exemplified by the semantic tag `AcquisitionAnnouncement` that marks up the sentence 'Exampleville-based Giant Foo Corp. announced Friday it will acquire Innovative Boo Co. headquartered in Somewhere City for 2 million dollars.' This tag concisely summarizes the annotated text on an abstract level by making explicit the concept 'acquisition announcement'.

Rather than disambiguating the word sense of single terms (cf. Manning and Schütze, 1999, pp. 229–263), we focus on semantic markup that represents metadata about text units comprising at least a few words. Text units reflect the logical structure of text and represent structural document parts (e.g., chapters, paragraphs, or sentences). In this work, a text unit represents the textual content of an arbitrary structural document part that comprises at least two consecutive words. The granularity of semantic markup is thus determined by appropriately choosing the granularity of text units. Fine-grained semantic markup may require text units corresponding, for example, to sentences whereas coarse-grained semantic tags might necessitate text units comprising entire paragraphs. Text units resemble content objects introduced by DeRose et al. (1990, pp. 3–6). The authors modeled text as an ordered hierarchy of nested, meaningful content objects (e.g., chapters, sections, subsections, and paragraphs).

Besides providing a concise and concept-based summary of text units, semantic markup makes explicit the occurrence of domain-specific named entities in annotated text units. Named entities are specific instances of abstract classes or numerical expressions (Sekine et al., 2002). For example, the string 'Giant Foo Corp.' instantiates the abstract class 'company'. Henceforth, the term named entity type denotes an abstract class (e.g., person, company, or product) or a generic numerical expression (e.g., date or share price). For example, the literal '2 million dollars' is an instance of named entity type 'amount of money'. Consequently, the semantic tag exemplified above may include the referenced deal value as well: `AcquisitionAnnouncement AmountOfMoney="2000000.00 USD"`.

**Extensible Markup Language** The Extensible Markup Language, abbreviated XML, is a metalanguage for creating specific markup languages (Geroimenko, 2004, p. 195). The term metalanguage emphasizes that XML is not a markup language itself, but rather defines common properties of all XML-based markup languages. In particular, the Exten-

sible Markup Language only comprises a set of syntactic rules for defining domain-specific, special purpose, and XML-based markup languages, such as the Extensible Business Reporting Language (cf. Bergeron, 2003). First published in 1998, XML is an open standard endorsed by the World Wide Web Consortium (2000). XML is a subset of the widely accepted Standard Generalized Markup Language (SGML; ISO 8879, 1986).

During the initial design phase of XML, the World Wide Web Consortium paid particular attention to the following requirements: straightforward usage over the Internet, support by a wide range of applications, compatibility with SGML, and above all simplicity in creating both XML documents and software for processing them. These design objectives have largely contributed to the rapid increase in popularity and adoption of XML in the past years. For example, McComb (2004, p. 207) stated that the Extensible Markup Language “has gone from relative obscurity to virtual ubiquity in just a few years.” Binstock et al. (2003, pp. 4–5) positioned XML as the definitive, if not the de facto standard for transferring data between distinct pieces of software. “The entire computer industry as a whole, including most of the text processing community, has adopted XML”, according to Weiss et al. (2005, p. 18), “as its standard exchange format.” The successful emergence of this metalanguage is also reflected in the considerable number of both native XML and XML-enabled database systems (cf. Chaudhri et al., 2003).

An XML-based markup language is commonly defined by an XML document type definition (DTD) and its documentation. Each DTD specifies the structure shared by and the text components frequently occurring in the documents to be modeled (Geroimenko, 2004, pp. 42–43). Hence, a document type definition serves as a grammar for the underlying XML documents (Abiteboul et al., 2000, p. 38–45). It declares syntactically valid markup by listing elements and attributes that are allowed to occur in valid XML documents. In addition, a document type definition may define a hierarchy of nested elements or impose an ordering on the elements in valid XML documents. According to Abiteboul et al., a DTD can thus serve, to a very limited extent, as a schema for the data represented by XML documents. The accompanying DTD documentation is essential because it explicates the rationale behind elements of a markup language and therefore ensures appropriate DTD use (Maler and Andaloussi, 1996, p. 313).

**Semantic XML Markup** Due to its flexibility, simplicity, and widespread support, we employ the Extensible Markup Language to represent semantically annotated text documents. For the purpose of this work, semantic XML markup<sup>1</sup> is defined as follows:

**Definition 1 (Semantic XML Markup)** *Semantic XML markup denotes pairs of matching XML start and end tags that enclose text units in documents conforming to the Extensible Markup Language. Names of semantic tags explicitly convey informal metadata about the meaning of marked-up content by concisely describing concepts that domain*

---

<sup>1</sup>Henceforth, the terms content-based and content-descriptive are used synonymously with semantic. Furthermore, annotation is a synonym of markup. Moreover, the activity of inserting markup into electronic text is synonymously referred to as marking up, tagging, or annotating text.

**Table 1.1:** Excerpt of XML Document Containing Semantically Marked-Up Text

---

```
<BriefAcquisitionAnnouncement Company="Name: Giant Foo [AND] Name: Innovative Boo">
USA: Giant Foo to acquire Innovative Boo.</BriefAcquisitionAnnouncement>
<AcquisitionAnnouncement Company="Name: Giant Foo Corp.; Place: Exampleville [AND]
Name: Innovative Boo Co.; Place: Somewhere City" AmountOfMoney="2000000.00 USD">
Exampleville-based Giant Foo Corp. announced Friday it will acquire Innovative Boo Co. head-
quartered in Somewhere City for 2 million dollars.</AcquisitionAnnouncement><AnnualRevenues
Company="Name: Innovative Boo" AmountOfMoney="5300000.00 USD">Innovative Boo is an IT
consultancy with annual revenues of about 5.3 million dollars.</AnnualRevenues> "The acquisition
of Innovative Boo's advanced business intelligence expertise adds a valuable new dimension to our
group" a Giant Foo spokesman said in a statement on Friday.
```

---

*experts typically associate with marked-up text units. Optionally, attributes of semantic tags make explicit named entities occurring in marked-up text units. The syntax of semantic XML markup is defined in an XML document type definition, and the meaning of semantic XML markup is informally specified in the accompanying DTD documentation.*

Table 1.1 illustrates our notion of semantic XML markup. Sentences correspond to text units, three of which are semantically annotated and one of which remains untagged. Each semantically marked-up text unit is surrounded by two matching XML tags enclosed in angle brackets. The concept-based summary of topics, issues, and subject matters covered in tagged content is conveyed by the element name, which is part of XML start and end tags (e.g., `AnnualRevenues`). Furthermore, most start tags have attributes whose names correspond to domain-specific named entity types. Attribute values represent concrete instances of named entity types identified in marked-up text units. For example, the attribute name and attribute value pair `Company="Name: Innovative Boo"` makes explicit the occurrence of the company ‘Innovative Boo’ in the annotated text. In this work, attributes of semantic XML tags do not implicate relationships between extracted named entities. They merely enumerate named entities that occur in semantically marked-up text units.

Standard-conforming XML documents contain data and metadata in an easily processable plain text format. Despite a common misconception about so-called self-describing XML documents, machine-readability does not imply that software is capable of automatically ‘understanding’ the meaning of markup (Renear et al., 2002). In fact, XML DTDs merely define the syntax of valid XML documents. They do not formally specify the meaning of procedural, presentational, structural, or even semantic tags. Introducing a semantic continuum, Uschold (2003) distinguished between explicit and implicit, as well as formal and informal specifications of semantics intended for either human or machine processing. Given an appropriate documentation regarding the meaning of markup constructs (cf. Maler and Andaloussi, 1996, pp. 313–323), semantic XML markup as introduced above is explicit and expressed in an informal manner. In contrast, implicit semantics are shared by human consensus only whereas formally specified semantics are

expressed in a formal notation for semantic representation (Woods, 1975, p. 45).

## 1.3 Benefits of Semantic XML Markup

Facing ever increasing volumes of computer-accessible textual data, semantically tagging text archives adds value by providing embedded metadata about concepts and named entities occurring in marked-up documents. Organizations benefit from exploiting semantic metadata in two broad ways. Firstly, semantic XML markup tends to improve the effectiveness and efficiency of humans in retrieving information that is relevant to their specific information needs. Secondly, semantic metadata facilitates or perhaps even enables additional, advanced, or automated text processing and analysis.

**Retrieving Information** Unlike plain texts, semantically annotated XML documents are semi-structured data because marked-up text units are surrounded by XML tags whose names explicitly map textual content onto content-descriptive concepts. Users may thus take advantage of the documented DTD that defines all archive-specific concepts. Resembling an archive-specific ‘table of contents’, each document type definition serves as a ‘road map’ for browsing and querying the respective collection of XML documents (Wang and Liu, 2000, p. 354). In addition to a conventional full-text search, users may retrieve information from semantically tagged XML documents by specifying relevant concepts and named entities occurring on the fine-grained level of text units.

From the semiotic perspective, users trying to locate information do not search for documents containing specific keywords (i.e., symbols). They rather look for documents that deal with concepts (i.e., references) corresponding to their information needs. However, classical models in information retrieval (IR) represent documents by a set of index terms (Baeza-Yates and Ribeiro-Neto, 1999, p. 24). Consequently, these IR models only support keyword-based queries. Instead of merely relying on the presence or absence of index terms, conceptual information retrieval aims at capturing the meaning behind words to improve retrieval performance (Mauldin, 1991, pp. xvii). Focusing on domain-specific text archives, both Mauldin as well as Osborn and Sterling (1999) reported improvements of retrieval performance through concept-based search. Considering, for example, Table 1.1 on page 8, users of conceptual IR systems can search directly for documents mentioning the concept ‘acquisition announcement’ instead of having to search for texts including the index terms ‘acquisition’ and ‘announcement’. The latter full-text query might, for instance, return irrelevant texts that ‘announce’ the completion of an ‘acquisition’ instead of reporting on first-time public statements about acquisitions.

Furthermore, the relevance of retrieved documents can be increased if queries involving domain-specific named entities are fully supported. Huffman and Baudin (1997) addressed the problem of finding information about important entities, such as persons and consulting skills, across a large semi-structured information space. The authors reported that providing a structured search functionality for named entities dramatically improves the

retrieval quality in comparison with full-text search. In addition, Thompson and Dozier (1999) concluded that explicit name searching can improve the retrieval performance especially if a large proportion of queries contain personal names. Considering again Table 1.1 on page 8, semantic XML markup might be exploited by explicitly searching for text units that feature the concept ‘acquisition announcement’ and mention the company ‘Giant Foo Corp. based in Exampleville’.

Due to the extensive XML support by commercial and free software (cf. Chaudhri et al., 2003; Melton and Buxton, 2006), conceptual information retrieval via semantic XML markup is applicable in both academia and industry. Nevertheless, most XML query languages focus on numerical and short character data (Fuhr, 2003). Therefore, Fuhr and Großjohann (2004) designed the query language XIRQL, which additionally implements fundamental IR concepts, such as term weighting and relevance-oriented search, for effective information retrieval in text-centric XML archives. Query languages like XIRQL, XML Fragments (see Carmel et al., 2003; Chu-Carroll et al., 2006), or NEXI (see Trotman and Sigurbjörnsson, 2005; Kamps et al., 2006) aim at crossing the structure chasm (cf. Halevy et al., 2003) in text-centric XML documents between the semi-structured grammar imposed by XML tags on one side and unstructured textual content on the other. Consequently, query languages for text-centric XML documents enable users to fully reap the benefits of semantic XML markup by enabling (i) conceptual information retrieval based on content-descriptive names of XML tags, (ii) explicit queries involving named entities listed in attributes of XML tags, as well as (iii) full-text search in both marked-up and untagged text units.

Moreover, a considerable body of IR research is devoted to retrieval models that combine textual content with information on the structure of documents (Baeza-Yates and Ribeiro-Neto, 1999, pp. 61–65). Structure-aware retrieval models become semantics-aware as well if the granularity of structural document components corresponds to the granularity of semantic markup. Navarro and Baeza-Yates (1997) proposed a retrieval model based on proximal nodes that supports hierarchical index structures over text. Thereby, full-text queries can be restricted to certain structural text components, such as chapters or sections. Extra value is added to this retrieval model by marking up structural text units with content-descriptive XML tags. In this case, users could, for instance, search for paragraphs discussing the concept ‘acquisition announcement’ and mentioning the company ‘Innovative Boo’.

**Leveraging Semantics** Applications that support the W3C-recommended Extensible Style Sheet Language family are capable of converting XML documents into other formats, like transformed XML documents or plain text files (Geroimenko, 2004, pp. 211–215). For that reason, semantically tagged XML documents can also be considered as an intermediate representation for content-based text analysis, conversion, or processing. McComb (2004) illustrated the broad range of use cases for leveraging semantics in business systems, which spans from enterprise application integration, Web services to the Semantic Web. However, our notion of semantic XML markup primarily facilitates

document warehousing, information integration, and Semantic Web applications:

1. The term document warehouse, according to Sullivan (2001, pp. 11–20 and 524), denotes an integrated repository of textual documents and associated metadata, which is set up for providing text-based decision support. Tseng and Chou (2006) stressed the importance of document warehousing and proposed a framework that combines text processing with multi-dimensional OLAP techniques. Document warehouses comprise various types of documents collected from multiple sources. Essential content-descriptive features are automatically extracted from new documents to establish associative links between related content. Semantic markup obviously constitutes an advanced starting point for efficiently extracting content-based features because it explicitly identifies themes and named entities occurring in texts. During document loading and pre-processing, however, semantic markup must be converted into content-based metadata that meet warehouse-specific standards regarding syntax and semantics. In particular, names of semantic XML tags denoting thematic concepts must be unambiguously aligned with organization-specific terminology for encoding metadata stored in the repository.
2. The final goal of information integration, as Hearst (1998b, p. 12) put it, is to better serve the information needs of users by making disparate data sources work together. Information integration is an inherently complex task that aims at physically or logically providing access to complementary, yet distributed or heterogeneous, data sources (Jhingran et al., 2002, pp. 555–556). In today’s economy that is “based on a vast infrastructure of computer networks and the ability of applications to share data with XML,” (Halevy et al., 2006, p. 13) emphasized that “data needs to be shared appropriately between different service providers, and individuals need to be able to find the right information at the right time no matter where it resides.” Due to the abundance of textual content, increasing research efforts have been directed towards integrating unstructured, semi-structured, and structured data. For example, Bergamaschi et al. (1999), Somani et al. (2002), Grossman and Frieder (2004, pp. 211–255), as well as Chakaravathy et al. (2006) addressed this research challenge. Kashyap and Sheth (2000, pp. 1–87) emphasized the critical role of semantic, in particular content-dependent metadata in brokering information across heterogeneous data sources.

Semantic XML markup explicitly provides content-descriptive metadata. Thus, embedded semantic XML tags along with a fully documented XML document type definition may be leveraged to integrate semantically semi-structured XML archives with related semi-structured or structured data. For example, announcements of mergers and acquisitions, as exemplified in Table 1.1 on page 8, might be integrated with a relational database providing detailed financial statements of companies. Analogous to document warehousing, syntax and semantics of disparate data sources have to be aligned either manually or by employing existing schema matching algorithms (e.g., cf. Aguilera et al., 2002; Poggi and Abiteboul, 2005; Tansalarak

and Claypool, 2007; Do and Rahm, 2007).

3. Berners-Lee et al. (2001, p. 29) envisioned a “new form of Web content that is meaningful to computers” and that “will unleash a revolution of new possibilities.” The so-called Semantic Web strives for associating information published on the existing World Wide Web with machine-understandable metadata about its meaning. Based on content-descriptive metadata, semantic-driven applications can deliver superior solutions for accessing information, managing knowledge, or doing business on the Web (Sheth et al., 2002; Fensel et al., 2003, pp. 1–25). However, commercially viable applications ultimately depend on the large-scale availability of semantically annotated data (Handschuh and Staab, 2003a, pp. v–vii). Although mostly considered as semi-structured data due to HTML encoding, Web pages typically lack explicit semantic metadata about textual content.

Again, semantic XML markup may serve as a basis for efficiently augmenting textual content with machine-understandable metadata for use on the Semantic Web. This use case, nevertheless, requires the mapping of explicit, yet informal, semantic XML markup onto a formal representation of meaning intended for machine processing (cf. Uschold, 2003; Stuckenschmidt and Harmelen, 2005, pp. 3–61). Providing a shared and common understanding of a domain, formal ontology languages are an accepted means of representing formal semantics (McGuinness, 2003; Fensel, 2004). Hence, concepts and named entities (i.e., semantic XML markup) can either be mapped onto concepts listed in existing ontologies or may constitute pre-processed input data to algorithms for learning formal ontologies from text (cf. Maedche, 2002; Cimiano, 2006).

In addition, concepts and named entities made explicit by semantic XML markup may serve as semantically enriched input data for the purpose of knowledge discovery in textual databases (e.g., cf. Feldman and Dagan, 1995; Feldman and Sanger, 2007) and content analysis (e.g., cf. Krippendorff, 2004). Finally, semantic XML markup constitutes beneficial metadata in the context of integrating database and IR techniques for structured document handling (e.g., see Sengupta, 2000; Zhang et al., 2001; Cui et al., 2003). For example, Amer-Yahia et al. (2005) and Weikum (2007) discussed the necessity of truly merging database and information retrieval technologies, which have evolved largely independent of each other. Analogous to social tagging mentioned by Weikum (2007, pp. 26–27), large volumes of real-world data, which are capable of bootstrapping joint database and IR research efforts, can be generated by automatically transforming unstructured texts into semi-structured, semantically tagged XML documents.

## 1.4 Research Questions

As discussed in the preceding section, semantic XML markup facilitates the retrieval, integration, analysis, and utilization of textual content. Before organizations can reap

the benefits accrued from content-descriptive metadata, however, documents have to be enriched by inserting high-quality, consistent semantic markup. This task poses a serious obstacle, the semantic annotation bottleneck, to leveraging semantics in information systems. In principle, this obstacle can be overcome by manual, semi-automated, or entirely automated annotation of text documents. In reality, the immense human efforts required for manual annotation restrict the applicability of the first approach to a few archives of limited size, but exceptional importance (Butler et al., 2000). Due to the costs and tedious efforts involved, humans understandably tend to prefer exploiting semantic markup to creating it by manual annotation (Erdmann et al., 2001; Tallis et al., 2001).

The huge volumes of existing legacy textual data and continuously published text documents necessitate an approach to semantic XML markup that simultaneously minimizes human involvement and provides high-quality markup. This challenge may be effectively addressed by adopting the process-centric framework for knowledge discovery in textual databases (Frawley et al., 1991; Feldman and Dagan, 1995; Fayyad et al., 1996b). Consequently, the following primary research question is addressed by this work:

*Can techniques for knowledge discovery in textual databases be employed to convert large archives comprising domain-specific text documents of homogeneous content into semantically marked-up XML documents?*

A knowledge discovery approach to semantic XML tagging requires the identification of frequently occurring thematic concepts at the text unit level. In addition, discovered concepts must be concisely described by semantic labels that reflect topics, issues, and subject matters mentioned in respective text units. Furthermore, domain-specific named entities have to be extracted from text units because they serve as attribute values of semantic XML tags. Subsequently, an archive-specific XML document type definition has to be derived, in which names of semantic XML tags correspond to labels of frequently occurring concepts. On the basis of actually occurring concepts and named entities, text files must be automatically transformed into semantically tagged XML documents that adhere to the derived DTD. Finally, the markup quality has to be evaluated to assess the statistical validity of the generated semantic XML markup. Consequently, the primary research question stated above implies the following secondary research questions:

1. *How to discover frequently recurring thematic concepts at the text unit level?*
2. *How to assign semantic, content-descriptive labels to identified thematic concepts?*
3. *How to extract domain-specific named entities from text units?*
4. *How to establish a domain-specific, concept-based XML document type definition that reflects the thematic structure on the text unit level by enumerating frequently occurring thematic concepts and associated named entity types?*
5. *How to automatically convert text documents into semantically annotated XML documents by marking up text units according to the concept-based XML DTD?*

### 6. *How to evaluate the quality of automatically created semantic XML markup?*

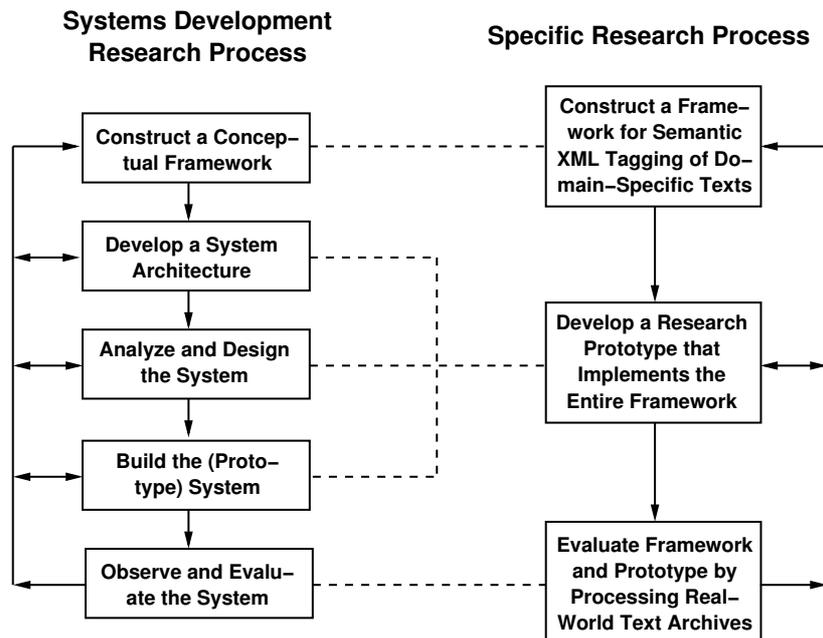
We intentionally focus on marking up domain-specific archives comprising documents of relatively homogeneous content, as opposed to general text collections. Thereby, the chance of actually finding frequently recurring concepts among text units is sufficiently high. Developing a document type definition always requires designers to limit the scope of considered document types to avoid a “lack of commonality” (Maler and Andaloussi, 1996, pp. 79–80). In addition, existing vocabulary collections compiled from domain-specific languages for special purposes (Bowker and Pearson, 2002, pp. 25–41) may be utilized for discovering thematic concepts. To ensure a high quality of semantic XML markup, domain knowledge supplied by human experts is incorporated, if necessary and appropriate. Due to the costs of utilizing expert knowledge, however, we focus on processing sufficiently large text collections only. When tagging large archives, the valued benefits of automatically creating semantic XML markup are likely to outweigh the initial costs of adopting an inherently semi-automated knowledge discovery approach.

## 1.5 Research Methodology

A research methodology is a specific combination of process, methods, and tools that are employed in conducting research (Nunamaker et al., 1991, p. 91). Different research methodologies have been advocated for the field of information systems (Galliers, 1991). To review this wide range, Galliers distinguished scientific, or empirical, (e.g., surveys) from interpretivist (e.g., action research) approaches to information systems research.

Referring to March and Smith (1995), Hevner et al. (2004) argued that acquiring knowledge in information systems involves two distinct, but complementary, research paradigms: behavioral science and design science. Rooted in natural-science research, the former seeks to develop and justify theories that explain or predict organizational and human phenomena occurring in the context of information technology. The latter, problem-solving paradigm of research is rooted in engineering. Design science seeks to create innovative artifacts (e.g., ideas or products) that contribute to the effective and efficient employment of information technology in organizations. According to March and Smith (1995, p. 253), behavioral or “natural science tries to understand reality” whereas “design science attempts to create things that serve human purposes.”

Following the design-science paradigm, the research underlying this work was specifically conducted by adopting the systems development methodology advocated by Nunamaker et al. (1991). Systems development, as the authors emphasized, is a key element in forming a well-grounded research program in information systems. This multi-methodological approach to information systems research encompasses four interdependent research strategies: theory building, systems development, experimentation, and observation. However, artifact development assumes a central role in this integrated approach because it bridges theory building, which conceptualizes artifacts, and experimentation/observation, which measures the impact of artifacts.



**Figure 1.3:** Systems Development Research Process (Source: Nunamaker et al., 1991, p. 98) and Specific Research Process

The systems development methodology comprises a generic research process (Nunamaker et al., 1991, pp. 97–101), which is depicted on the left side of Figure 1.3. As indicated by solid arrows, this process is inherently iterative in nature and thus underpins the idea that artifact design is an iterative search process for satisfactory solutions (Hevner et al., 2004, pp. 88–90). Once a new, creative, and important research question is identified, theory building corresponds to constructing a conceptual framework that must be useful in organizing ideas and suggesting actions. Systems development comprises three stages: developing the system architecture, system analysis and design, and constructing the (prototype) system. Prototypes allow researchers to quickly assess the feasibility of conceptualized solutions. After building the (prototype) system, it must be thoroughly evaluated in the experimentation/observation phase. Evaluation results have to be interpreted with respect to the established conceptual framework. Since systems development is an evolutionary process, evaluation results are typically fed back to theory building and the development stages or may even raise new research questions.

By means of dashed lines, Figure 1.3 additionally maps the generic systems development research process onto the specific process adopted within this research. As depicted on the right side of Figure 1.3, this specific process consists of three main phases. Concerning theory building, the main contribution of this work is a novel conceptual framework for semantic XML markup of domain-specific text archives. This framework represents a knowledge discovery approach to semantic XML tagging. Constructing the conceptual

framework addresses the first five secondary research questions raised in the preceding Section 1.4. The systems development phase corresponds to designing and building a research prototype that implements the entire conceptual framework. Obviously, prototype development is an essential prerequisite for evaluating the proposed conceptual framework. Subsequently, the quality of automatically created semantic XML markup is evaluated by employing the prototype system to semantically annotate real-world text archives. Insights from observing and evaluating the system are used to incrementally refine both the framework and the prototype. The evaluation phase addresses the sixth secondary research question raised in the preceding subsection.

## 1.6 Outline

How to employ knowledge discovery techniques for transforming domain-specific text archives into semantically marked-up XML documents? This overall research objective is henceforth addressed by (i) establishing a conceptual framework for semantic XML tagging of domain-specific text archives, (ii) developing a research prototype that implements the entire framework, and (iii) evaluating the quality of semantic XML markup by processing real-world text archives. The remainder of this work is organized as follows:

**Theory Building** Before discussing our approach to semantic XML tagging, the underlying discipline knowledge discovery in textual databases (KDT) has to be introduced. Chapter 2 (Literature Review) thus gives an overview of knowledge discovery in texts with emphasis upon the KDT process. In addition, basic techniques for pre-processing and storing textual data as well as extracting named entities are briefly explained. Subsequently, research from three relevant areas is reviewed: concept discovery in textual data, semantic text annotation, and schema discovery in marked-up documents.

An integrated overview on the approach taken shall be given before discussing details of individual research issues. Therefore, Chapter 3 (DIAsDEM<sup>2</sup> Framework) defines fundamental terminology and presents an introductory overview of our two-phase framework for semantic XML tagging of domain-specific text archives. In phase one, an interactive knowledge discovery process (i) discovers and labels concepts occurring at the text unit level of plain texts, (ii) derives a concept-based XML document type definition on the basis of identified concepts and prevailing named entity types, as well as (iii) semantically tags all text documents to enable an assessment of markup quality. In the second, rather application-oriented phase of our framework, large volumes of domain-specific text documents are automatically tagged by utilizing classification knowledge acquired in phase one.

Simultaneously providing high-quality semantic XML markup and reducing human efforts requires a dedicated knowledge discovery solution. Chapter 4 (DIAsDEM Knowledge

---

<sup>2</sup>The framework for semantic XML tagging, the knowledge discovery process, and the prototype system are named after the research project DIAsDEM supported by Deutsche Forschungsgemeinschaft.

Discovery Process) contributes to theory building by proposing a specific knowledge discovery process for the semantic XML annotation of text documents, which constitutes phase one of our conceptual framework. Chapter 4 provides details about pre-processing textual data, searching for patterns (i.e., classification knowledge), and post-processing discovered patterns. Concretely, solutions are suggested for the following research objectives: Identification of frequently occurring concepts at the text unit level, extraction of named entities from text units, semantic labeling of discovered concepts, as well as establishing a concept-based XML DTD, and converting training texts into semantically marked-up XML documents. Finally, the chapter discusses the challenge of balancing the need for KDT process automation with the required incorporation of domain knowledge.

**Systems Development** To experimentally evaluate the framework specified in Chapters 3 and 4, a functional prototype system must be implemented that supports all features of the DIASDEM framework for semantic XML tagging of domain-specific text archives. To that end, Chapter 5 (DIASDEM Workbench) gives an overview on the prototype system developed as an integral part of this research. Chapter 5 outlines system requirements and describes the architecture of DIASDEM WORKBENCH. Subsequently, the core functionality of this research prototype is explained.

**Experimentation/Observation** Ultimately, the applicability of our framework can only be verified in case studies involving real-world text archives. Therefore, Chapter 6 (Experimental Evaluation) initially establishes an evaluation schema for assessing the quality of semantic XML markup. Specifically, the overall markup quality is determined by appropriately assigned semantic XML tags and correctly extracted named entities.

Competitive intelligence denotes both the process of conducting a competitive analysis and the resulting actionable information (Herring, 1988, p. 5). Particularly, competitive analysis is a phase in the strategic management process. This phase covers the systematic and legal collection, storage, and analysis of publicly available information about current and potential competitors, other stakeholders, and the environment to anticipate trends and strategic moves (Winkler, 2003, pp. 4–5). A successful competitive analysis strongly depends on leveraging computer-accessible texts about trends and stakeholders. For example, Sullivan (2001, pp. 407–436) and Chen et al. (2002) consistently stressed the importance of knowledge discovery methods in the competitive intelligence field.

For that reason, we selected real-world archives from the competitive intelligence domain to assess the quality of automatically created semantic XML markup. In Chapter 6, two case studies of transforming text archives into semantically marked-up XML documents are thus described. We processed a collection of German Commercial Register entries in the first case. English news items about United States mergers and acquisitions were semantically annotated in the second case study.

**Looking Forward** Chapter 7 (Conclusions) summarizes the contribution of this work and indicates promising research challenges to enhance the conceptual framework and thereby improve the capabilities of DIASDEM WORKBENCH.



## 2 Literature Review

This chapter sets our research in the context of related work. Firstly, Chapter 2 introduces the research area of knowledge discovery in textual databases and surveys important text processing techniques from two other relevant disciplines. Secondly, this chapter outlines the state of research on concept discovery in text documents, semantic text annotation, as well as schema discovery in marked-up documents. The chapter summary differentiates our objectives and our chosen approach from related work.

Research into identifying concepts in text documents is highly related to semantic XML markup on an abstract level because semantic markup, as defined in Section 1.2, explicates concepts that domain experts typically associate with marked-up text units. Research on semantic text annotation offers various notions of semantic markup and above all distinct techniques for actually annotating documents, which are surveyed in detail as well. Furthermore, research into deriving a conceptual schema from marked-up documents is relevant because our framework outputs a concept-based XML document type definition. In addition, related work affecting only specific aspects of our conceptual framework (e.g., cluster labeling) is discussed in the respective chapters.

### 2.1 Storage, Retrieval, and Analysis of Textual Data

Since the DIASDEM framework adopts a knowledge discovery approach to semantic XML tagging, the underlying discipline is introduced in the next subsection. The following two subsections survey highly relevant methods developed by the information retrieval and information extraction research community, respectively.

#### 2.1.1 Knowledge Discovery in Textual Databases

Feldman and Dagan (1995) coined the term knowledge discovery in textual databases that refers to applying knowledge discovery techniques to textual, instead of structured, data. Within the underlying discipline of knowledge discovery in databases, the nontrivial process of identifying implicit, previously unknown, statistically valid, and actionable patterns in data constitutes the object of research (Frawley et al., 1991; Fayyad et al., 1996b). A pattern is an expression in some language that describes a specific subset of data without exhaustively enumerating all facts or that represents a model applicable to a data set (Fayyad et al., 1996a, pp. 40–41). Before elaborating on knowledge discovery in detail, three important terms remain to be delineated for the scope of this work.

**Data, Information, and Knowledge** Typically stored in databases or flat files, data henceforth denote a collection of facts, observations, or measurements (Fayyad et al., 1996b, p. 6). Textual data, and synonymously text, stand for a specific data category: written or recorded spoken natural language. When referring to data, emphasis is placed on the mere existence, processing issues, or syntactic properties of facts, observations, or measurements. Therefore, their actual meaning is not considered at all. In contrast, information is described as “data endowed with relevance and purpose” (Drucker, 1988, p. 46) and “data that change individual decisions” (Hackathorn, 1999, p. 33). For example, raw data is transformed into purposefully organized information by contextualization, categorization, analysis, error removal, or summarization (Davenport and Prusak, 2000, pp. 3–5). When referring to information, both semantic aspects and the subjectively perceived relevance of data are thus emphasized.

Davenport and Prusak (2000, p. 5) pragmatically defined knowledge as “a fluid mix of framed experience, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information.” The authors underlined that knowledge is mainly created by thinking or the interaction of human beings. Besides residing in the human mind, knowledge is often embedded in documents or organizational routines. Taking the KDD perspective, Fayyad et al. (1996b, pp. 8–9) argued that patterns in data represent knowledge if they exceed a user-imposed, domain-specific interestingness threshold. Within this work, however, knowledge encompasses only patterns that are sufficiently interesting, meaningful, and relevant to affect organizational processes. This notion is influenced by Hackathorn (1999, p. 33): “Knowledge is information that changes organizational processes.” Consequently, patterns in data constitute information if they make sense and are relevant to individuals only.

**Knowledge Discovery** In contrast to straightforward computations (e.g., of descriptive statistics), knowledge discovery is a nontrivial process since it involves search or inference algorithms to find new, previously unknown patterns in data (Fayyad et al., 1996a, p. 41). To constitute knowledge, extracted patterns must be valid and actionable. The former criterion requires patterns to be statistically valid, with a certain degree of certainty, on data not utilized during the discovery phase (Vazirgiannis et al., 2003, p. 13). The latter criterion stipulates that patterns must be ultimately understandable and potentially useful to the human target group. The overall value of patterns is captured by the important notion of interestingness (cf. Silberschatz and Tuzhilin, 1996; Geng and Hamilton, 2006) that combines novelty, validity, simplicity, and usefulness. Silberschatz and Tuzhilin distinguished objective interestingness measures (e.g., statistical validity) from subjective ones (e.g., unexpectedness and actionability). Independent of concretely applied measures, humans ultimately set the desired interestingness threshold, above which extracted patterns are considered to be information or valuable knowledge.

Knowledge discovery in textual databases (KDT) is an inherently multi-disciplinary research area, which borrows methods from statistics, machine learning, artificial intelligence, and database research. In addition, analyzing unstructured text requires techniques

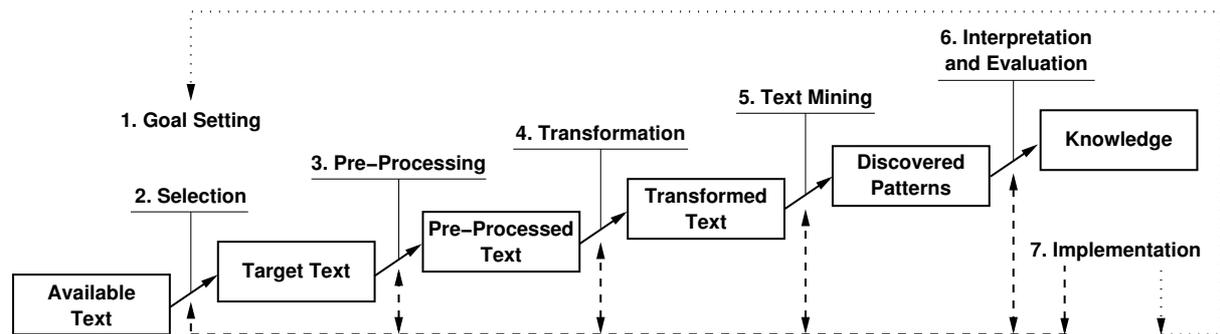
**Table 2.1:** Knowledge Discovery in Textual Databases: Tasks and Applications

<i>Knowledge Discovery Task</i>	<i>Exemplary KDT Application</i>
Exploratory Data Analysis	Kohonen (2001, pp. 296–299) constructed a self-organizing map that allows a visual, content-based exploration of 6,840,568 patent abstracts.
Descriptive Modeling	Forman et al. (2006) employed a clustering algorithm to identify common and emerging issues in textual, product-related help-desk records.
Predictive Modeling	Weiss et al. (2005, pp. 172–178) described a text categorization system that automatically assigns pre-defined topic names to newswire articles.
Discovering Rules	Using an algorithm for association rules discovery, Feldman et al. (1998) found frequently occurring concepts in 51,725 financial news stories.
Retrieval by Content	Employing text classification based on decision trees, Lee and Lu (2003) designed an adaptive recommender system for mobile news services.

from computational statistics, information extraction, statistical linguistics, and information retrieval (Renz and Franke, 2003, pp. 2–5). According to Renz and Franke, knowledge discovery in texts is different from KDD because it usually involves high-dimensional, but rather sparsely populated, data sets. Furthermore, features of data sets representing textual content are often semantically interrelated. For example, the two features representing the terms ‘company’ and ‘firm’ are obviously related. However, after appropriately pre-processing textual data to alleviate text-related particularities, standard KDD algorithms may be employed to solve KDT tasks as well.

Based on the objectives of humans analyzing data, Hand et al. (2001, pp. 11–15) identified five generic knowledge discovery tasks. Firstly, exploratory data analysis refers to the interactive, mostly visual exploration of massive data sets without having clear ideas regarding the patterns to look for. Secondly, descriptive modeling aims at globally describing a data set or the respective process creating data. Thirdly, predictive modeling is employed to fit a global model to a data set that permits the value of one attribute to be predicted given values of other features. Fourthly, the goal of discovering rules is to infer rule-based descriptions of interesting, but locally occurring, phenomena in large data sets. Finally, retrieval by content supports users in finding data that are similar to a given pattern of interest. Table 2.1 summarizes an application of each task.

**The KDT Process** As illustrated in Figure 2.1, discovering knowledge in textual data is modeled as a multi-step process (cf. Fayyad et al., 1996b, pp. 9–11). This process is inherently interactive and iterative since “one cannot expect to obtain useful knowledge simply by pushing a lot of data to a black box” (Mannila, 1997, p. 42). Dashed lines in Figure 2.1 indicate possible iterations and loops between any two phases. For example, interpreting discovered patterns may cause the user to modify parameters and re-execute a text mining algorithm. The process starts with setting the overall analysis goals and ends with implementing discovered knowledge to change and improve, respectively, or-



**Figure 2.1:** Process of Knowledge Discovery in Textual Databases (Adapted and Extended from Fayyad et al., 1996b, p. 10)

ganizational processes. The dotted line in Figure 2.1 implies that acting on identified knowledge may raise new questions and necessitate further analyses.

After acquiring a thorough domain understanding, the objectives of discovering knowledge are synchronized with goals pursued by the target group (Fayyad et al., 1996b, p. 10). Subsequently, textual data required for the analysis is selected and/or sampled from archives comprising available text documents. Thereafter, selected textual data is cleansed and pre-processed. Besides standard KDD pre-processing procedures (e.g., removing noise if appropriate), this phase includes various operations to transform text into a numerical representation suitable for applying knowledge discovery algorithms. Important pre-processing operations are introduced in Subsection 2.1.2. Subsequent to pre-processing text, the transformation phase comprises data reduction and projection, which strongly depend on the specific objectives of knowledge discovery. For example, pre-processed textual data may be transformed by reducing the typically huge number of features that may initially represent thousands of individual words. However, dimensionality reduction algorithms must preserve important semantic and domain-specific characteristics of original text documents as much as necessary.

Although both words are often used interchangeably, text mining is not a synonym for knowledge discovery in textual databases in this work. The former term describes an important, but arguably not the most important, phase of the knowledge discovery process. The text mining step encompasses three main activities. At first, the overall objective (e.g., finding typical service calls to improve help-desk applications) must be mapped onto a particular text mining method, such as clustering (Fayyad et al., 1996b, pp. 10–11). Second is choosing and parameterizing a suitable text mining algorithm. For instance, the  $k$ -means clustering algorithm may be selected to discover  $k = 100$  typical service calls. Thirdly, the chosen algorithm is executed to search for, or infer, ultimately interesting patterns in pre-processed and transformed data. As depicted in Figure 2.1, extracted patterns (e.g.,  $k = 100$  clusters) are interpreted and evaluated based on a priori chosen interestingness measures. In this decisive phase, useless patterns are separated

from novel and interesting ones that constitute the desired knowledge. Evaluation results often trigger iterations whereby preceding process steps are executed once again. Finally, a successful knowledge discovery process is terminated by acting on extracted, novel, and interesting patterns to attain the initial organizational objectives.

### 2.1.2 Information Storage and Retrieval

Information retrieval systems process text documents and requests for information by identifying and subsequently retrieving documents in response to information requests (Salton, 1989, pp. 229–231). Textual documents are retrieved based on their similarity to an information need expressed as a query. To meet an information request, the similarity between stored documents and a query is determined by comparing values of attributes (i.e., content identifiers) that are attached to stored documents and the query. Content identifiers attached to documents are known as descriptors, index terms, or keywords.

**Document Pre-Processing** By assigning content identifiers to documents (i.e., by indexing), unstructured textual content is transformed into structured document surrogates (Salton, 1989, pp. 275–277). Storing and successfully retrieving texts thus involves extensive pre-processing of documents to assign them adequate content identifiers. These pre-processing techniques are highly applicable for discovering knowledge in textual data (Renz and Franke, 2003, p. 4). The knowledge discovery approach to semantic XML tagging pursued in this work is no exception to the rule. Baeza-Yates and Ribeiro-Neto (1999, pp. 165–173) discussed five main operations for pre-processing text:

1. *Lexical text analysis* converts character-based textual data into a sequence of clearly separated words, numbers, and punctuation marks. For example, multi-token words (e.g., ‘news agency’) and sentence boundaries are identified during the tokenization process (Manning and Schütze, 1999, pp. 123–131 and 134–136).
2. *Eliminating stopwords* means removing words that occur too frequently to be good discriminators from the document-specific list of potential content identifiers. Meaningless articles, prepositions, and conjunctions are stopword candidates because they are typically useless for text retrieval due to their high frequency.
3. *Stemming the remaining words* is performed by replacing terms with their respective root forms. Stemming algorithms (e.g., Porter, 1980a) reduce syntactical diversity, but preserve the meaning of terms. For example, inflected verb forms (e.g., ‘acquired’ and ‘acquiring’) are substituted by their infinitives (e.g., ‘acquire’).
4. *Selecting index terms* refers to carefully choosing words or multi-word phrases that finally serve as content identifiers for text documents. Index terms were manually selected by domain experts in the past. Nowadays, algorithms exist that automatically extract useful index terms from text (Moens, 2000, pp. 77–102).

5. *Constructing term categorization structures* refers to establishing a controlled vocabulary. For instance, thesauri (ISO 2788, 1986) provide a standard vocabulary for consistent indexing, searching, and automatic query expansion. A thesaurus includes domain-specific words along with related terms (e.g., synonyms).

**Vector-Space Model** Various mathematical models have been suggested to numerically represent text (Baeza-Yates and Ribeiro-Neto, 1999, pp. 24–61). However, the vector-space model introduced by Salton (1968, pp. 236–243) is an efficient document representation for information retrieval and knowledge discovery (Sullivan, 2001, pp. 328–337).

The vector-space model assumes a collection comprising  $n \in \mathbb{N}$  text documents and a set  $D = \{d_1, d_2, \dots, d_m\}$  of  $m \in \mathbb{N}$  distinct, a priori selected content identifiers and descriptors, respectively (Salton and Lesk, 1968). Text document  $i$  ( $i = 1, 2, \dots, n$ ) is modeled as an  $m$ -dimensional property vector  $\mathbf{t}_i = [\tau_{i,1}, \tau_{i,2}, \dots, \tau_{i,m}]$ . Coefficient  $\tau_{i,j}$  ( $j = 1, 2, \dots, m$ ) represents the weight of content descriptor  $d_j$  in document  $i$ . Weight  $\tau_{i,j}$  should reflect the importance, the presumed value of content descriptor  $d_j$  for retrieving text document  $i$  (Salton and Buckley, 1988, p. 516). Consequently, the entire text collection is modeled as an  $(n \times m)$  matrix  $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n]^T$ . The rows of matrix  $\mathbf{T}$  represent  $n$  document vectors. Columns of matrix  $\mathbf{T}$  correspond to  $m$  content descriptors.

According to Salton and Buckley (1988, pp. 516–517), term weighting involves three components to ensure a high overall retrieval performance. The term frequency component is solely based on the frequency of descriptor  $d_j$  in text document  $i$  whereas the collection frequency component depends on the frequency of  $d_j$  in the entire collection. Additionally, the normalization component accounts for varying document sizes in a collection. Salton and Buckley therefore emphasized that creating a weighting scheme requires considering vocabulary characteristics, the frequency of collection updates, and the variation of document size. For average text collections, Baeza-Yates and Ribeiro-Neto (1999, pp. 29–30) argued, the best known weighting schemes calculate weights by multiplying the normalized term frequency  $\text{ntf}(i, j)$  of content descriptor  $d_j$  in document  $i$  by the inverse document frequency  $\text{idf}(j)$  of content descriptor  $d_j$  in the collection:

$$\tau_{i,j} = \text{ntf}(i, j) \cdot \text{idf}(j) = \frac{\text{tf}(i, j)}{\max_{k=1, \dots, m} \text{tf}(i, k)} \cdot \log \frac{n}{\text{df}(j)} \quad (2.1)$$

In Expression 2.1,  $\text{tf}(i, j)$  denotes the frequency of descriptor  $d_j$  in document  $i$ . Furthermore,  $\text{df}(j)$  denotes the number of documents in the entire collection that contain descriptor  $d_j$ . Intuitively, the best index terms “occur frequently in individual documents but rarely in the remainder of the collection” (Salton, 1989, p. 280). To balance these two effects, weights computed according to Expression 2.1 increase if descriptor  $d_j$  frequently appears in text document  $i$  or if  $d_j$  rarely occurs in the entire text archive  $\mathbf{T}$ .

The popular vector-space model has many advantages for storing and retrieving textual data. It is simple, fast, and offers a retrieval performance that tends to be superior to, or almost as good as, alternative models (Baeza-Yates and Ribeiro-Neto, 1999, p. 30). A

disadvantage is the assumed mutual independence of content descriptors. However, Baeza-Yates and Ribeiro-Neto argued that this assumption is negligible for practical applications. Since text is mapped onto a bag-of-content-descriptors representation, any information concerning the ordering of occurring descriptors is intentionally disregarded. This fact must be taken into account for specific knowledge discovery applications.

### 2.1.3 Information Extraction

Research in information extraction (IE) investigates how to identify facts of interest in text documents (cf. Pazienza, 1997, p. v). Grishman (1997, p. 10) defined information extraction as “identification of instances of a particular class of events or relationships in natural language text” to extract relevant attribute values of the event or relationship. The specification of one particular event or relationship is referred to as a scenario. Taking a broader perspective, Moens (2006, p. 4) defined IE as the task of making unstructured data sources “suitable for information processing tasks” by identifying, classifying, and/or structuring specific information therein into semantic classes that are known in advance. After retrieving relevant documents using IR systems, information extraction techniques are capable of identifying a priori specified facts inside the documents (Cowie and Lehnert, 1996). In an information retrieval context, as Moens (2006, p. 13) emphasized, information extraction techniques facilitate a highly focused retrieval.

Since information extraction aims at finding facts inside documents, only text passages are analyzed in detail (cf. Moens, 2006, p. 7). In particular, IE algorithms attempt to transform factual information into a structured representation by finding values for predefined slots, or attributes, of scenario-specific templates (Manning and Schütze, 1999, p. 376). An empty template comprises attribute names that correspond to important features of the respective scenario. A filled template denotes the final, tabular information extraction output, which represents a concrete scenario occurrence in terms of attributes and their extracted values. The information extraction task of filling a priori defined templates is related to identifying semantic concepts in text. Thus, representative research projects on extracting relational data from texts are reviewed in Subsection 2.2.2.

Between 1987 and 1998, seven Message Understanding Conferences, abbreviated MUC-1 through MUC-7, facilitated the application-oriented research on information extraction from unstructured texts (Jackson and Moulinier, 2002, pp. 76–78). Sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA), researchers from academia and industry were invited to solve several predefined information extraction tasks (e.g., extracting details of terrorist attacks) in a competitive environment. The availability of real-world training and test documents allowed a rigorous evaluation of all solutions. The quality of information extraction is assessed in terms of precision, recall, and metrics derived thereof (Grishman, 1997, pp. 20–21). Precision is defined as the ratio of correctly filled template slots and the total number of slots filled. In contrast, recall denotes the percentage of correctly filled template slots to the total number of slot fillers in the respective text. Analogous to information retrieval, there exists a trade-off between precision

and recall in IE as efforts to increase recall tend to worsen precision. The evaluation of IE techniques is discussed by Moens (2006, pp. 179–197) in great detail.

After retrieving news items about corporate takeovers, for example, information extraction techniques are capable of extracting detailed data that explicate who acquired whom for what price. An instantiated scenario template `CorporateAcquisition` hence contains values filling the slots `AcquiringCompany`, `AcquiredFirm`, and `MonetaryDealValue`. By employing appropriate IE algorithms, which are reviewed by Moens (2006, pp. 23–150) as well as Feldman and Sanger (2007, pp. 119–145), the filled template `CorporateAcquisition: AcquiringCompany="Giant Foo"; AcquiredFirm="Innovative Boo"; MonetaryDealValue="2000000.00 USD"` can be extracted from the press release illustrated in Table 1.1 on page 8. Since named entities serve as slot fillers, recognizing them is an important subtask of information extraction.

**Named Entity Recognition** According to Sekine et al. (2002), named entities (e.g., ‘Giant Foo Corp.’ and ‘5.3 million dollars’) are specific instances of named entity types that are either abstract classes, like ‘company’, or numerical expressions, such as ‘amount of money’. Unlike complex events or scenario patterns (e.g., ‘corporate acquisition’), abstract classes denote basic objects (e.g., ‘product’) or subjects (e.g., ‘person’). Sekine et al. proposed a hierarchy of about 150 named entity types whose instances are likely to occur in general news stories. In our framework, domain-specific named entities are extracted from text units as they are potential attribute values of semantic XML tags.

Recognizing named entities requires extracting various types of proper names and special expressions (Grishman, 1997, p. 15). The latter (e.g., telephone numbers and dates) are typically identified by applying carefully handcrafted regular expressions (Manning and Schütze, 1999, p. 131). Algorithms designed to extract and disambiguate proper names often rely on various dictionaries, rules, and heuristics (e.g., Volk and Clematide, 2001; Navarro et al., 2003). Other approaches to recognizing proper names exploit techniques from computational linguistics and machine learning to minimize costly human efforts (e.g., Bunescu and Mooney, 2004; Turmo et al., 2006; Downey et al., 2007). Jackson and Moulinier (2002, p. 77) summarized that extracting proper names of persons, companies, and places from English newswire texts “was more or less a solved problem” in 1998. Feldman and Sanger (2007, p. 96) and Moens (2006, p. 203) argued analogously.

## 2.2 Discovering Concepts in Textual Data

Most automatic indexing techniques select natural language index terms directly from documents to be indexed (Moens, 2000, pp. 77–102). Thereby, index terms in the form of single words or multi-word phrases are assumed to appropriately reflect the subject matters of indexed documents. As introduced in Section 1.3, conceptual information retrieval pursues a different approach by capturing the meaning behind often ambiguous words. To that end, semantic concepts are assigned to documents instead of indexing selected

natural language terms. Woods (1997, pp. 6–12), for instance, considered conceptual indexing a better way to organize knowledge than classical, term-based indexing.

In the context of indexing, the term concept is standardized to denote a unit of thought whose semantic content can be re-expressed by a combination of other and different concepts (ISO 5963, 1985). In this section, however, the term concept, or synonymously semantic concept, pragmatically denotes any abstract and unambiguous representation of the meaning of textual content. This view corresponds to the notion of concepts adopted by Loh et al. (2000), Nasukawa and Nagano (2001, p. 969), as well as Feldman and Sanger (2007, pp. 6–7). In particular, semantic concepts are distinct from individual, often ambiguously used words. Unlike, for example, Thompson and Dozier (1999), identified named entities (e.g., names of companies) are not considered to be concepts henceforth.

Before reviewing related research explicitly aimed at semantic text annotation in Section 2.3, we subsequently survey research whose objective is to discover semantic concepts in text documents. Although these techniques neither output semantically tagged XML documents nor insert semantic annotations into text files, they are related to the main research question pursued in this work. In general, identifying semantic concepts in text documents is a prerequisite for their content-based markup. Therefore, we discuss representative work in topic discovery, extraction of relational tuples from text, as well as approaches to inferring taxonomies, thesauri, and ontologies from text documents.

### **2.2.1 Topic Discovery in Text Documents**

Topics of text documents concisely characterize the subject matters, or themes, discussed therein. Identifying document-specific topics hence corresponds to answering the question of what the respective document is about. Hutchins (1997) discussed the notion of aboutness, or topicality, in the context of subject indexing. According to Hutchins, it is a common view that the aboutness of documents is expressed by index terms. Typically, index terms are keywords and phrases that are significant indicators of the content and thereby provide a thematic summary of the document. The traditional approach to indexing thus requires human indexers who are capable of stating what a document is about by formulating a summary and selecting good index terms.

Moens (2000, pp. 12–13) distinguished the intrinsic subject, aboutness, or topicality of documents from their context-specific meaning. A document usually has a relatively permanent and agreed upon aboutness whereas its concrete meaning depends on characteristics of the particular communication context. Specifically, the meaning of documents is determined by several cognitive factors like task, knowledge, or attitude of readers. The objective notion of aboutness resembles semantic metadata, as defined in Section 1.2.

In our framework, names of semantic XML tags convey the meaning of marked-up text units by summarizing concepts that domain experts typically associate with marked-up content. Identifying topics in a collection of text documents is a related research question because extracted topic descriptors may serve as semantic markup for the respective text units. However, we only consider semi-automated and automated approaches to topic

identification that are designed to discover new, previously unknown topics in an archive. Thereby, we deliberately exclude manual techniques as well as the broad range of text classification algorithms (cf. Sebastiani, 2002; Feldman and Sanger, 2007, pp. 64–81). Unlike our exploratory framework, text classification is focused on assigning documents to existing, pre-defined topic categories. Techniques for summarizing or abstracting text documents (cf. Moens, 2000, pp. 133–154) are not discussed either because condensed, but nevertheless rather lengthy, summaries cannot serve as semantic XML tags.

The aboutness of text documents can be determined at different granularity levels (Moens, 2000, p. 25). On the one hand, overall topics characterize subject matters of the entire text document. Subtopics represent subjects, or themes, of certain text passages on the other hand. Following this observation, we subsequently survey representative research into identifying overall topics at the text level. Thereafter, specific research into discovering topics of more fine-grained text structures is discussed. Complementarily to extracting individual topics, Subsection 2.2.3 surveys techniques for inferring complex taxonomies, thesauri, and ontologies from concepts occurring in a text collection.

**Topic Discovery at the Text Level** Clustering, or unsupervised learning, is a common approach to topic discovery. A clustering algorithm groups documents such that texts assigned to the same cluster are more similar than texts assigned to different clusters (cf. Jain and Dubes, 1988, pp. 1–6). Documents assigned to the same cluster are assumed to cover similar topics (e.g., cf. Chin et al., 2006; Pons-Porrata et al., 2007), which nevertheless remain to be characterized in a content-descriptive and human-understandable way. However, (semi-) automatically extracting the aboutness, or topicality, of documents is not always the objective of text clustering. Finding groups of similar documents without explicitly specifying their salient topics is often sufficient in exploratory data analysis.

SCATTER/GATHER is a cluster-based approach to browsing large document collections (Cutting et al., 1992). Initially, the entire document collection is segmented (i.e., scattered) into a small number of clusters, whose prevalent topics are visualized in the form of content-descriptive summaries. On the basis of these cluster-specific topic descriptions, the user selects one or more interesting clusters for further browsing. Subsequently, the documents of all selected clusters are merged (i.e., gathered) to form a new sub-collection of texts that constitutes the input to the next scatter/gather iteration. Cutting et al. summarized the main topics of each cluster in a so-called cluster digest. A generated cluster digest contains the titles of documents near the cluster centroid and a list of words that occur most frequently in documents assigned to the respective cluster. Inspired by SCATTER/GATHER, Agrawal et al. (2000) presented the interactive and iterative clustering algorithm C-EVOLVE, which, for efficiency reasons, directly generates cluster digests instead of clustering the entire (sub-) collection of text documents. Cluster digests comprise a limited number of representative text documents to provide cluster descriptions.

After clustering a text archive to discover topics, Ertöz et al. (2004) characterized the topicality of each cluster by extracting words occurring most frequently in the respective texts. Analogously, Karypis and Han (2000b) as well as Dhillon and Modha (2001) selected

the ten highest weighted features, which are typically keywords, in the centroid vector of each text document cluster to summarize its aboutness. Describing salient topics in text document clusters only on the basis of frequently occurring keywords is a common, but by no means the most effective, solution. To that end, related work on cluster labeling is discussed in detail in Subsection 4.4.1.

Based on the self-organizing map algorithm for unsupervised learning (Kohonen, 2001), WEBSOM is another technique for thematically organizing large text collections (Kaski et al., 1998; Kohonen, 2001, pp. 286–299). WEBSOM maps documents onto a two-dimensional array of nodes (i.e., clusters) such that documents assigned to the same node feature similar topics. Texts assigned to nearby nodes of the array are more similar than texts assigned to distant map nodes. Hence, the two-dimensional document map provides an intuitively understandable, graphical, and content-based segmentation of the entire text archive. Some nodes of the map, so-called landmarks, are automatically labeled to quickly provide insights into the topics discussed in texts assigned to the respective map area (Kohonen, 2001, pp. 295–296). Landmark labels are words that frequently occur in articles within the labeled map area and rarely elsewhere. Alternatively, the LABELSOM algorithm may be utilized to automatically label the nodes of document maps (Rauber, 1999; Rauber and Merkl, 2003). Topic descriptions induced by LABELSOM comprise features, typically keywords, of document vectors that are most relevant for their mapping onto a particular node of the document map.

Another strain of research into topic discovery referred to as statistical topic modeling is, for example, represented by Griffiths and Steyvers (2004) as well as Newman et al. (2007). Both research groups employed a generative, probabilistic model for document collections (i.e., latent Dirichlet allocation) proposed by Blei et al. (2003). This approach resembles model-based clustering (cf. Han and Kamber, 2006, pp. 429) as it requires learning topic-word and document-topic probability distributions for a text archive in an unsupervised manner (Newman et al., 2007, p. 367). Statistical topic modeling simultaneously discovers topics in text collections, characterizes topics by their most important words, and assigns degrees of topic membership to documents. Newman et al. (2007, p. 369) emphasized the necessity to manually assign meaningful labels to topics.

The research area of topic detection and tracking focuses on temporal aspects of event-based topic discovery (cf. Allan, 2002). This research community aims at identifying reports about distinct real-world events in a continuous stream of broadcast news stories. Consequently, the input stream of news is initially separated into discrete stories (Allan, 2002, pp. 1–26). This important pre-processing task is related to discovering topics at the text passage level discussed below. Thereafter, new topics are recognized in the stream of separated news stories, which may, for instance, be triggered by a political event. Allan et al. (2002, pp. 219–222) characterized topics by the most important keywords, named entities, or noun groups. In parallel, all incoming follow-up news stories of known topics are clustered into groups of stories discussing the same topic. Tracking topics refers to identifying topic-specific follow-up reports on the basis of a few sample documents. Essentially, topic detection and labeling algorithms (e.g., Ponte and Croft, 1997; Mori

et al., 2006; Pons-Porrata et al., 2007) are fine-tuned to process continuous streams of incoming text documents. They are not directly related to our work on semantic XML markup because we assume the entire text archive of training documents to be accessible.

**Topic Discovery at the Text Passage Level** The above-mentioned techniques for identifying salient topics at the text level can also be applied to determine the aboutness of a priori defined text passages. To that end, structural text units at the required granularity level (e.g., sections or paragraphs) are extracted from each text document prior to topic identification. Unlike this straightforward approach to topic discovery at the passage level, however, identifying the internal, rather fine-grained thematic structure of documents requires discovering distinct topics within the text as well as segmenting the text based on topic changes (Li and Yamanishi, 2003, pp. 521–522).

When retrieving information from collections of documents comprising a variety of subtopics, the retrieval performance can be considerably improved by employing passage retrieval strategies (Salton et al., 1993; Callan, 1994; Jiang and Zhai, 2006, p. 296). Instead of searching for complete documents, passage retrieval systems locate finer-grained text segments that match a particular information need. To determine subtopical text segments in long documents covering subjects of principal interest, Salton et al. (1996) proposed computing the pairwise similarity between text segments of varying, pre-defined length. Pairwise similarities of text segments exceeding a threshold are edge weights of an undirected graph, and text segments are represented by graph nodes. Subsequently, themes are identified by a graph-based clustering algorithm that extracts highly connected subgraphs comprising at least three nodes. Salton et al. did not consider techniques for labeling themes to characterize their subject matters.

TEXTTILING is a technique for structuring expository text documents into contiguous, non-overlapping, multi-paragraph units covering the same subtopic (Hearst and Plaunt, 1993; Hearst, 1997). In particular, Hearst (1997) introduced the final TEXTTILING algorithm for subdividing texts into subtopic segments, which identifies major topic shifts based on lexical co-occurrence patterns. After tokenizing a text document, the algorithm computes a similarity score for each pair of consecutive sentences. Subtopic boundaries are inserted between adjacent text segments whose similarity score is a local minimum with respect to the left and right neighboring scores. Hearst deliberately excluded the question of characterizing, or labeling, subject matters of discovered subtopic segments.

Reynar (1999) introduced two algorithms for identifying topic boundaries without addressing the question of topic labeling either. The first algorithm is solely based on analyzing word frequency statistics whereas the second algorithm utilizes features like the occurrence of domain-specific cue phrases (e.g., ‘brought to you by’) or the repetition of named entities to compute the likelihood of an existing topic boundary between two text segments. Combining the TEXTTILING approach and previous work on text summarization, Mather and Note (2000) proposed an algorithm that first discovers topic boundaries and then labels subtopic regions with frequently occurring noun phrases.

Li and Yamanishi (2003) emphasized that topic analysis consists of two main tasks:

topic identification and text segmentation based on topic changes. To that end, Li and Yamanishi proposed an unsupervised statistical learning approach based on a stochastic topic model. After text segmentation, each identified subtopic block is labeled with words that are closely related to the prevailing topics therein. In this context, Jiang and Zhai (2006) presented a method based on hidden Markov models that detects both thematically coherent and relevant passages of varying lengths in response to a query.

**Assessment** As defined in Section 1.2, names of semantic XML tags concisely summarize the topics, or subject matters, of marked-up text units. More specifically, semantic XML tags reflect semantic concepts that domain experts typically associate with marked-up content. Regardless of the granularity level, topic discovery in text documents is thus an important body of relevant research. Consequently, clustering techniques that group texts or text segments based on their content are reviewed in detail in Section 4.3 as unsupervised learning is employed in our knowledge discovery process as well. Due to the overall objective of semantic XML tagging, however, our research requires a different approach to topic discovery than pursued by most techniques discussed above. First of all, identified topics must be characterized in a concise, content-descriptive, and human-comprehensible way because topic labels subsequently serve as names of semantic XML tags. Furthermore, input documents are not segmented based on domain-specific subtopic regions although we aim at discovering topics at the passage, or text unit, level. Instead of marking up consecutive subtopic regions of varying length, our framework is designed to semantically annotate pre-defined structural text units (e.g., sentences).

## 2.2.2 Extracting Relational Tuples from Text

Documents often comprise factual data, such as announcements of corporate takeovers or bankruptcies. In these cases, actually rather structured facts are conveyed via unstructured natural language as text documents are overwhelmingly written by human authors for primary consumption by human readers. As introduced in Subsection 2.1.3, information extraction techniques aim at filling pre-defined, scenario-specific templates (e.g., ‘acquisition announcement’) from texts. After extraction, factual data can be stored in a relational database to facilitate, for example, information integration and queries using the Structured Query Language (SQL). The IE task of filling templates by identifying instances of events and relationships in natural language text is synonymously referred to as extracting relational tuples<sup>1</sup> from text. For example, the schema `AcquisitionAnnouncement(AcquiringCompany: string, AcquiredCompany: string, MonetaryDealValue: string)` defines the relation `AcquisitionAnnouncement`  $\subseteq$  `string`

<sup>1</sup>Notation: Given  $n \in \mathbb{N}$  arbitrary, not necessarily distinct, and atomic domains  $D_i$  ( $i = 1, 2, \dots, n$ ), relation  $R \subseteq D_1 \times \dots \times D_n$  is defined as the Cartesian product of  $n$  domains or as a subset thereof (cf. Codd, 1970, 1990). Each domain  $D_i$  is associated with a descriptive attribute name  $a_i \in \{a_1, \dots, a_n\}$ . The schema of relation  $R$  is denoted by  $R(a_1: D_1, \dots, a_n: D_n)$ . An  $n$ -ary element  $(d_1, \dots, d_n) \in R$  with attribute values  $d_i \in D_i$  is referred to as a tuple of relation  $R$ .

$\times$  string  $\times$  string. Tuples of this relation, like ("Giant Foo", "Innovative Boo", "2000000.00 USD"), can be extracted from business news stories.

Searching for tuples of pre-defined relations in text documents is aimed at identifying occurrences of related semantic concepts, as defined in Section 1.2. Hence, appropriately chosen and content-descriptive names of relations, such as `AcquisitionAnnouncement`, correspond to semantic concept labels. Analogously, attribute values of identified relational tuples may constitute named entities, whose types are characterized by attribute names (e.g., `AcquiringCompany`). Text documents may thus be transformed into semantically annotated XML documents. Unlike semantic XML markup in our framework, however, tuples do define a semantic relationship between their attribute values.

Research into extracting tuples from documents is only relevant if the proposed solutions explicitly address the question of processing plain, completely unstructured natural language text. Consequently, research on creating source-specific information extraction procedures, so-called wrappers, is deliberately excluded from the following discussion if the techniques are only applicable to semi-structured documents, such as HTML files published on the World Wide Web (e.g., Chang et al., 2006; Zhao and Betz, 2007). Abolhassani et al. (2003) distinguished the knowledge engineering approach from the machine learning approach to information extraction. The former relies on extraction rules that are manually created and updated by knowledge engineers. In contrast, the machine learning approach infers extraction rules from annotated training documents. Additionally, the fully automated approach to information extraction aims at both discovering relations that exist in text corpora and extracting tuples thereof.

**Knowledge Engineering Approach** According to Jackson and Moulinier (2002, p. 81), the `FINITE STATE AUTOMATON TEXT UNDERSTANDING SYSTEM (FASTUS)` (Appelt et al., 1995; Hobbs et al., 1996) is a representative of information extraction systems that were employed in the series of Message Understanding Conferences. FASTUS represents information extraction rules as a cascade of finite state automata that match incoming text against expressions of regular languages handcrafted by knowledge engineers. Each finite state automaton maps output data from its predecessor onto a more structured representation, which in turn constitutes input for the next information extraction phase.

Earlier stages of the FASTUS extraction process recognize smaller and domain-independent linguistic objects. After tokenizing character sequences into tokens, multi-token words and proper names are identified. Sentences are decomposed into noun groups and verbs groups by means of finite state grammars to avoid computationally expensive full syntactic parsing (Grishman, 1997, pp. 21–23). Unlike other partial parsing approaches (e.g., Neumann and Piskorski, 2002), FASTUS does not employ part-of-speech tagging algorithms for reasons of scalability. Later IE process stages take these linguistic objects as input and match them against finite state rules to fill slots of domain-dependent templates. By employing elaborate rules, extracted relational tuples from distinct text passages are merged into a composite template if they provide information about the same entity or event. Finally, the system outputs relational tuples in the form of filled, domain-dependent

templates. FASTUS performed well in MUC-6 by achieving 76% precision at 74% recall in the template-filling task.

**Machine Learning Approach** Soderland (1999) introduced the WHISK system capable of learning information extraction rules from sentences of unstructured text. These IE rules are system-specific regular expressions, whose matches correspond to tuples of pre-defined relations. In a pre-processing phase, sentences are syntactically parsed. In addition, terms may be mapped onto semantic classes that denote groups of synonymously used terms. Both syntactic and semantic metadata are valid elements of regular expressions to be learned. WHISK incorporates a supervised learning algorithm to induce IE rules and thus requires hand-tagged training examples. However, human efforts are reduced by iteratively asking the knowledge engineer to annotate only texts likely to improve the performance of the learning algorithm. WHISK attained 69% precision at 46% recall when applied to extract tuples of the management succession relation `SuccessionEvent(Organization, Post, PersonOut, PersonIn)` from news stories.

Soderland (1999) adopted a top-down approach to rule induction, which starts with an ‘empty’ information extraction rule matching all training sentences. Subsequently, this rule is iteratively extended by adding discriminative terms that serve as appropriate boundaries for the values of relational tuples to be extracted. In contrast to top-down rule induction, Califf and Mooney (2003) presented the bottom-up relational learning algorithm RAPIER, which incorporates techniques from inductive logic programming systems to induce IE rules from hand-tagged training documents. This ‘specific to general’ approach generalizes an initial set of rules, each of which represents a filled template occurring in a training text. Attaining 86% precision at 60% recall, RAPIER scored slightly better than WHISK in extracting 17-ary tuples from textual job postings.

Unlike the above mentioned machine learning approaches to information extraction, the SNOWBALL system requires only a few user-supplied tuples of the relation, whose tuples occur in plain texts (Agichtein and Gravano, 2000; Yu and Agichtein, 2003). The user-provided exemplary tuples, like (“United Nations”, “New York”), of the target relation, like `Headquarter(Organization, Location)`, are input to an algorithm that infers rules for extracting them from the text collection. Applying these induced rules to the same documents typically results in new tuples of the target relation. This process is iteratively repeated whereby only sufficiently reliable tuples and patterns are employed in the next iteration to ensure a high information extraction performance. Agichtein and Gravano (2000) reported 76% precision at 45% recall when employing SNOWBALL to identify tuples of the relation `Headquarter` in a large collection of news stories.

Etzioni et al. (2004, 2005) introduced the KNOWITALL system that extracts facts (i.e., tuples of relations) from textual Web pages in an unsupervised, domain-independent, and scalable manner by extensively leveraging existing Web search engines. The generate-and-test architecture resembles the iterative SNOWBALL approach, but it further reduces human efforts by inducing the required seed tuples from generic, domain-independent extraction patterns. Inspired by Hearst (1992), KNOWITALL initially utilizes extraction

patterns that exploit part-of-speech tags to generate candidate facts. Subsequently, the plausibility of candidate facts is tested by using pointwise mutual information statistics and treating the Web as a massive corpus. The system is capable of instantiating  $n$ -ary relations with arbitrary, user-supplied relation names and multiple, user-given predicate arguments, yet the authors experimentally focused on unary relations.

Suchanek et al. (2006) presented another approach that combines linguistic and statistical analysis to extract tuples of a priori specified, binary relations from textual Web pages. Instead of limiting the linguistic feature space to part-of-speech information (e.g., cf. SNOWBALL and KNOWITALL), the authors proposed utilizing deep syntactic analysis features as input to a supervised machine learning technique. The prototype system LEILA requires a definition of each target relation in the form of a specifically programmed “function that decides into which of” four “categories a pair of words falls” (Suchanek et al., 2006, p. 713). To set up the semantic search engine NAGA (cf. Kasneci et al., 2007), the prototype system was employed to extend the knowledge base YAGO, which was initially established from structured data sources by carefully, semi-automatically combining rule-based and heuristic methods (cf. Suchanek et al., 2007).

**Fully Automated Approach** To reduce the large number of search engine queries required by KNOWITALL (cf. Etzioni et al., 2005) and to eliminate the need to specify relation names and predicates, Banko et al. (2007) and Etzioni et al. (2007) introduced the OPEN INFORMATION EXTRACTION paradigm. It obviates relation specificity by automatically discovering possible relations of interest in one pass over the corpus without requiring any human input. The prototype system TEXTRUNNER comprises a self-supervised learner, a single-pass extractor, and a quality assessor module. Based on a small corpus sample annotated by a deep linguistic parser, the learning module automatically identifies and labels trustworthy and untrustworthy candidate relations between noun phrases. This labeled data set, whose features can be evaluated at extraction time without a parser, is input to training a Naive Bayes classifier used by the extractor module. Relation extraction and assessment is performed akin to KNOWITALL. Banko et al. (2007, p. 2674) extracted over one million high-quality tuples of binary relations, which are “potentially useful for information extraction,” from approx. nine million Web pages.

In contrast to the OPEN INFORMATION EXTRACTION paradigm that aims at identifying as many distinct relations as possible in large text corpora, Sekine (2006) introduced the notion of on-demand information extraction to discover relations in documents returned by an information retrieval system in response to a user’s query. Analogous to TEXTRUNNER, the proposed system strongly relies on natural language processing by performing part-of-speech tagging, dependency analysis, rule-based named entity extraction, and paraphrase discovery. Instead of training a classifier, however, an unsupervised machine learning technique is employed to identify groups of patterns that correspond to  $n$ -ary, unlabeled relations of named entities occurring in the same semantic context.

**Assessment** Due to the research conducted in the series of Message Understanding Conferences, extracting relational tuples from unstructured text is a partly solved research issue. Machine learning and fully automated techniques significantly reduced the necessary efforts of knowledge engineers to establish and maintain extensive repositories of information extraction rules. For the purpose of semantic XML tagging of domain-specific text documents, the originally posed research question on how to extract domain-specific named entities from text units is thus considered solved by related work. Consequently, we do not address named entity extraction in the remainder of this work in detail, but rather utilize the techniques conceived by this research community.

Extracting tuples from text focuses either on populating a small number of relations of major importance or finding all occurring relations independent of their importance. Instead of filling a few pre-defined templates or merely enumerating relational tuples, our framework is designed to find frequently occurring, important semantic concepts and associated named entities in text units. Discovered concepts may be interpreted as relation names, and extracted named entities might correspond to attribute values of these relations. However, we do not aim at establishing a semantic relation between frequently occurring named entity types in text units, which are marked up with a common semantic concept. Hence, semantic XML annotation and extraction of relational tuples constitute complimentary research questions. After converting texts into semantically tagged XML documents, information extraction techniques can, for example, be utilized to further structure the content of text units featuring important semantic concepts.

### 2.2.3 Learning Taxonomies, Thesauri, and Ontologies

The tasks of conceptually indexing and retrieving text documents are largely facilitated by an existing, typically domain-specific controlled vocabulary (Baeza-Yates and Ribeiro-Neto, 1999, pp. 170–171). A controlled vocabulary is a pre-defined list of accepted index terms or concepts (Moens, 2000, pp. 51–53). It usually incorporates some form of semantic structure, such as definitions of hierarchical and associative relations between terms. Controlled vocabularies normalize indexing concepts and therefore allow for the identification of indexing concepts with a well-defined semantic meaning. After employing a word sense disambiguation algorithm (cf. Manning and Schütze, 1999, pp. 229–261), disambiguated terms can be automatically mapped onto appropriate concepts by exploiting the semantic structure of a controlled vocabulary. For example, the piece of text ‘rate of interest’, whose third token is disambiguated as ‘money charged for borrowing money’, is mapped onto the semantic indexing concept ‘interest rate’.

Mapping terms occurring in text documents onto unambiguous semantic concepts is slightly related to semantic XML tagging. Our framework for semantic XML markup, however, assigns meaningful concept descriptors to entire text units instead of focusing on the meaning of single terms. Resembling topic discovery reviewed in Subsection 2.2.1, each text unit may be content-descriptively characterized by the set of descriptors that comprises all extracted semantic concepts. Clearly, an existing and semantically enriched

controlled vocabulary is useful for extracting topics as well. For that reason, we give an overview on techniques for (semi-) automatically inducing an initial, semantically enriched controlled vocabulary from a collection of text documents. In particular, approaches to learning taxonomies, thesauri, and ontologies from free form text are introduced in the remainder of this subsection. We again focus on techniques designed to process unstructured textual data and thus deliberately exclude approaches that exploit characteristics of structured or semi-structured data (e.g., Suchanek et al., 2007).

**Taxonomy Learning** Taxonomies are classification systems that organize concepts represented by words or phrases (Saeed, 2003, p. 68). Typically, taxonomies have a hierarchical structure and are synonymously referred to as topic hierarchies, concept trees, or concept hierarchies. In particular, more general concepts (e.g., ‘stockholders’ equity’) are included at upper hierarchy levels of taxonomies whereas specific concepts (e.g., ‘preferred stock’) are placed at lower taxonomic levels (Chung et al., 2002, p. 608). Thereby, the hyponymy, or inclusion, relation between concepts is made explicit (Saeed, 2003, p. 68). A hyponym includes the meaning of a more general concept and thus ‘is a kind of’ the more general concept. For example, ‘preferred stock’ and ‘common stock’ are hyponyms of ‘stockholders’ equity’. In contrast, a more general concept, such as ‘stockholders’ equity’, is referred to as the hypernym of more specific concepts.

Hearst (1992, 1998a) described a light-weight method for acquiring hyponyms from text by searching for occurrences of hyponymy-specific lexico-syntactic patterns. For example, the pattern ‘NP<sub>1</sub> such that NP<sub>2</sub> or NP<sub>3</sub>’ matches three semantically related noun phrases NP<sub>*i*</sub>, *i* ∈ {1, 2, 3}. According to the semantics of the English language, occurrences of this lexico-syntactic pattern (e.g., ‘stockholders’ equity such as common stock or preferred stock’) imply that NP<sub>2</sub> and NP<sub>3</sub> are hyponyms of NP<sub>1</sub>. Given a few initial lexico-syntactic patterns, new patterns are automatically and iteratively discovered in a text corpus. After clustering nouns using a hierarchical agglomerative algorithm, Caraballo (1999) employed the Hearst approach to find appropriate hypernyms for labeling each cluster in the hierarchy of related nouns. Another simple, but effective, statistical approach to building a concept hierarchy was pursued by Sanderson and Lawrie (2000). Based on term co-occurrence frequencies, the method assumes that term *t*<sub>1</sub> is a hypernym of term *t*<sub>2</sub> if the documents containing *t*<sub>2</sub> are a subset of those comprising *t*<sub>1</sub>.

Subsequent to clustering text documents, Glover et al. (2002) analyzed term frequency statistics in the entire collection and in clusters to infer hierarchical term relationships. Mao et al. (2003a) also employed hierarchical agglomerative clustering to establish a collection-specific taxonomy. The authors proposed disambiguating the sense of salient words in each cluster and looking up related synonyms, hypernyms, and hyponyms in a lexical reference system. Each cluster of text documents is assigned a meaningful label derived from dominant, semantically enriched concepts in documents that are assigned to itself and its children clusters. Snow et al. (2006) proposed an algorithm for inducing taxonomies that incorporates evidence from multiple classifiers over heterogeneous taxonomic relationships and considers lexical ambiguity in one probabilistic framework.

**Thesaurus Learning** Having Greek and Latin origins, the term thesaurus has been used to denote a ‘treasury’ of words and phrases for centuries (Foskett, 1997, pp. 111–115). Most thesauri comprise domain-specific, as opposed to general, entries in the form of single words, composite words, or phrases. Typically, thesauri (i) provide an alphabetical list of entries, (ii) contain internal references from entries to related terms, and (iii) group entries into conceptual categories. Although thesauri are a useful means of enhancing one’s style, they nowadays primarily serve as indexing languages for information retrieval to provide a standardized vocabulary and ensure consistent indexing (cf. ISO 2788, 1986).

Usually, thesaurus entries are associated with concepts as basic semantic units for conveying ideas (Baeza-Yates and Ribeiro-Neto, 1999, p. 171). Since a semantic concept can often be expressed by various synonymous words, only one thesaurus entry (i.e., the preferred term) represents a given semantic concept for indexing purposes (ISO 2788, 1986). All non-preferred terms reference their respective preferred thesaurus entry. Furthermore, the concretely referred to sense of homonyms is disambiguated by a qualifying word or phrase, such as ‘bank (river)’. Moreover, thesauri make explicit different kinds of hierarchical (e.g., hyponymy) and associative relationships between entries.

Grefenstette (1994) introduced the SEXTANT approach to extracting corpus-specific semantics that is based on domain-independent syntax parsing techniques. Given a corpus of unstructured text, SEXTANT creates a corpus-specific list of similar words for each term and thereby groups terms into conceptual categories. Assuming that similar terms are used in a similar way throughout the corpus, the similarity between terms depends on their local context words and can thus be computed by applying standard similarity measures. Grefenstette demonstrated that term similarities extracted by this method correspond to human similarity judgments. The resulting context-based term similarities are exploited to automatically enrich an existing thesaurus or to establish an initial, preliminary thesaurus. In the latter case, SEXTANT only extracts semantically related terms instead of inferring complex hierarchical or associative term relationships.

Chen et al. (1997), Houston et al. (2000), and Chen et al. (2003) advocated the statistical concept space approach to automatically constructing domain-specific thesauri. A concept space is a network of terms and weighted term associations that make explicit semantic concepts and their relationships occurring in the underlying information space (i.e., documents in an archive). Subsequent to domain-specific text pre-processing and automatic indexing to identify relevant terms, an asymmetric similarity measure is used to compute pairwise similarities of index terms in the co-occurrence analysis step. Based on the resulting network-like thesaurus, an associative retrieval component employs a standard neural network algorithm to determine terms that are semantically related to a user-supplied query. The concept space approach is also adopted by LEXIMANCER to map texts onto a set of thesaurus-like concepts (Smith, 2000).

**Ontology Learning** In the context of formal knowledge representation in artificial intelligence systems, Gruber (1993, p. 199) defined an ontology as an “explicit specification of a conceptualization.” According to Gruber, the term conceptualization denotes a simplified

and abstract view of the focal real-world domain, or a part thereof. A conceptualization reflects the relevant concepts (e.g., entities and attributes), their definitions, and their inter-relationships (cf. Uschold and Gruninger, 1996). We adopt the definition proposed by Fensel (2004, p. 3) that additionally incorporates the aspect of knowledge sharing and the focus on formal, machine-processable domain models: “An ontology is a formal, explicit specification of a shared conceptualization.” Formally specified ontologies associate concepts and their relations with (i) representational names, (ii) human-readable definitions describing the meaning of concepts, as well as (iii) formal axioms constraining their interpretation and enabling automated reasoning (Gruber, 1993, p. 199). In particular, the required formal specification and the possibility to define arbitrary logical relationships between concepts distinguish ontologies from typical, informally defined taxonomies and thesauri (cf. McGuinness, 2003, pp. 173–185).

Maedche and Staab (2000) as well as Maedche (2002) introduced the ontology learning framework *TEXT-TO-ONTO*, whose implementation is embedded into the ontology engineering workbench *ONTOEDIT*. The multi-disciplinary framework supports the semi-automated extraction of ontologies from unstructured texts and their subsequent maintenance. More specifically, Maedche (2002) presented several ontology learning algorithms that have been adapted from existing work in, for example, hierarchical clustering, association rule discovery, and pattern matching. The framework comprises algorithms to extract lexical entries referring to concepts and their relations, to organize concepts in a taxonomy, to identify non-taxonomic relations between concepts, as well as to prune and refine initially inferred ontologies. Emphasis is placed on the aspect of importing and pre-processing natural language text using the *SMES* system (cf. Neumann et al., 1997) because linguistic knowledge improves the quality of ontology learning. Cimiano (2006) continued Maedche’s work on ontology learning and population by presenting various algorithms within the *ONTOLOGY LEARNING LAYER CAKE* framework. Cimiano focused on inducing concept hierarchies, learning attributes of binary relations between concepts, as well as extracting instances of concepts to populate an ontology from texts.

Missikoff et al. (2002a,b) pursued an iterative approach to learning, validating, and managing ontologies, which includes the *ONTOLEARN* module for automatic concept discovery from texts. Analogously to Maedche (2002), the authors strongly relied on linguistic pre-processing before employing statistical and machine learning algorithms. The system extracts relevant domain-specific terminology, disambiguates term senses, assigns unambiguous concept names to terms, as well as identifies taxonomic and similarity relations among concepts. After deploying *ONTOLEARN* in a project, Missikoff et al. (2002a, pp. 56–57) reported a remarkable increase in ontology creation productivity.

Buitelaar et al. (2004) described the *ONTOLT* plug-in for the ontology editor *PROTÉGÉ* (cf. Gennari et al., 2003), which connects linguistic analysis and ontology learning. *ONTOLT* applies manually or semi-automatically created rules to extract concepts and their attributes from linguistically annotated text corpora. Before finally constructing or extending an ontology, statistical pre-processing is applied to filter relevant, domain-specific concepts from selected linguistic entities (e.g., nouns and their modifiers).

**Assessment** Various research communities proposed, implemented, and successfully evaluated techniques to assist knowledge engineers in the laborious task of establishing domain-specific controlled vocabularies. In general, however, human beings still play a decisive role in assessing, pruning, and extending an algorithmically established controlled vocabulary. Due to the complexity of natural language, the development of fully automated techniques is an open research challenge (cf. Zhou, 2007, pp. 247–249).

Our framework for semantic XML tagging does not aim at identifying semantic concepts at the term level. Its objective is rather to annotate structural text units with content-descriptive tags. Typical semantic tags consist of several atomic concepts, or basic units of thought, whose specific combination conveys the meaning of marked-up text. For example, the XML tag `<AcquisitionAnnouncement>` comprises the atomic concepts ‘acquisition’ and ‘announcement’, both of which are supposed to occur in marked-up text units. Nevertheless, our framework does incorporate a domain-specific controlled vocabulary to ensure a high quality of semantic text annotation, as discussed in Subsection 4.2.4. Consequently, research into learning taxonomies, thesauri, and ontologies constitutes another complimentary research area to the research questions pursued in this work. Without doubt, these techniques can potentially reduce the human efforts of establishing and maintaining a controlled vocabulary prior to semantic XML tagging.

## 2.3 Semantic Annotation of Text Documents

After reviewing implicitly related work on concept identification in the preceding section, we now survey explicitly relevant research into semantic text annotation. Colloquially, the term annotation refers to a note by which explanation or comment is added to a text document (Bechhofer and Goble, 2003, pp. 196–197). The authors differentiated between three major annotation types: textual annotation, link annotation, and semantic annotation. Textual annotations are created by adding personal notes and comments to documents whereas link annotations denote embedded references to thematically related resources. In principle, these two annotation types primarily address human readers and authors. Semantic annotations convey explicit metadata about the meaning of annotated content. Besides explicitly providing semantic clues for human consumption, semantic annotations are mostly inserted into documents to facilitate automated document processing by software agents. To that end, the meaning of semantic annotations must be explicitly defined in a formal ontology (cf. Fensel, 2004, pp. 3–10).

In accordance with our research questions raised in Section 1.4, we restrict the review to research into explicit semantic annotation of unstructured textual data. Consequently, we refrain from discussing work on textual annotation (e.g., ANNOTEA, Kahan et al., 2002) and research into link annotation (e.g., COHSE, Bechhofer and Goble, 2003). By intentionally excluding work on textual annotation, we in particular refrain from surveying so-called Web 2.0 tagging systems (e.g., cf. Marlow et al., 2006; Ames and Naaman, 2007). Tagging systems allow users to intuitively attach personal keywords to Internet resources

(e.g., blog entries, links, photos or videos) for subsequent personal and community use without relying on any form of controlled vocabulary at all.

Due to the interest in Semantic Web applications (cf. Berners-Lee et al., 2001), researchers proposed various techniques to facilitate the semantic annotation of semi-structured Web pages. In contrast, our work focuses on semantic markup of unstructured texts. Hence, frameworks and tools are deliberately excluded from the discussion if their application is restricted to semi-structured HTML or XML files. Analogously, specific techniques for other semi-structured documents (e.g., BibTex entries) and highly structured, often ungrammatical text files (e.g., seminar announcements and classifieds) are not reviewed either. Finally, linguistic and natural language processing research into word sense disambiguation and semantic annotation at the term level is not considered relevant either as our framework is designed to annotate structural text units.

In the remainder of this section, we outline characteristic research projects aimed at semantically tagging entire text documents or individual text passages. Thereby, we discuss techniques for both informal and formal semantic annotation. Our framework for semantic text annotation generates explicit and informal semantic XML markup. To bootstrap the Semantic Web, however, a large body of related research has been directed at ontology-based, formal, and thus machine-understandable semantic markup of text documents. In this section, related research is primarily categorized with respect to the degree of apparently required human involvement. We distinguish between manual, semi-automated, and automated approaches to semantic text annotation.

### 2.3.1 Manual Semantic Text Annotation

Undoubtedly, the semantic enrichment of documents by hand is a laborious task. Nevertheless, some applications rely on manually crafted annotations because they require high quality, intellectually challenging, and/or fine-grained semantic markup. Following, we describe representative research projects that resort to manual annotation for these reasons. Thereafter, a characteristic framework and several tools are reviewed that support authors and domain experts in the process of semantically annotating texts. Finally, work on manual semantic annotation is assessed with respect to our research.

**Use Cases for Manual Semantic Markup** Launched in 1987, the Text Encoding Initiative (TEI) is an internationally accepted guideline for encoding literary and linguistic texts for scholarly research (Ide and Sperberg-McQueen, 1995, pp. 5–11). Nowadays, this guideline serves as the de facto standard for encoding and exchanging, for example, dictionaries, literary prose, and recorded spoken language. This guideline consists of several extensively documented SGML document type definitions for various types of scholarly text (Sperberg-McQueen and Burnard, 1995, pp. 22–37). SGML tags represent conventional document metadata, structural metadata (e.g., indicating paragraphs), or content-descriptive metadata (e.g., indicating stage directions in drama). Content-descriptive TEI tags are explicit and informal semantic annotations. The TEI Consortium also released

an XML-based guideline (cf. Sperberg-McQueen and Burnard, 2002).

Due to the potential benefits induced by the TEI guideline, scholars have an incentive to encode texts accordingly. To generate high quality markup, human domain experts are frequently employed to manually assign appropriate content-descriptive tags to text passages. Unlike these semantic tags, structural markup can be reliably created in an automated manner. For example, the ORLANDO project (Butler et al., 2000) purposefully extended TEI document type definitions to encode a literary history of British women writing. Team members compiled an archive of structurally and thematically annotated SGML documents and thereby enabled sophisticated document post-processing. Instead of marking up existing documents, a team of highly qualified researchers and graduate students authored textual content and corresponding semantic annotations in parallel. After agreeing upon a document type definition comprising 238 unique tags, team members inserted more than 577,000 annotations into approx. 2,500 texts, like biographies or historical reports. Despite taking social and technical precautions, the observed tagging consistency among the large team of authors did not meet the high expectations and thus proved to be a challenging research issue. Interestingly, this result confirmed previous empirical studies on inter-tagger consistency (Butler et al., 2000, pp. 124–125).

Bayerl et al. (2003b) emphasized the importance of manual annotations in the compilation of corpora and linguistic research material. In this context, Bayerl et al. (2003a) presented a corpus-based method to analyze the semantics of structural markup in XML-encoded scientific articles. Due to the absence of appropriately tagged archives, 158 scientific articles were annotated at the structural level and two linguistically motivated semantic levels. Analogously to the above mentioned ORLANDO project, domain specialists manually added semantic markup to already structurally tagged articles. During the annotation phase, a systematic methodology was adopted to ensure the consistent usage of semantic tags by all team members (Bayerl et al., 2003b).

Finally, Weiss-Lijn et al. (2002, 2003) described the GRIDVIS system for interactive exploration of semantically enriched document collections. Resembling passage retrieval systems (cf. Salton et al., 1993; Callan, 1994), GRIDVIS facilitates the goal-driven search for relevant information at the paragraph level by leveraging semantic metadata. Document collections are visualized by a matrix, whose rows represent a hierarchy of semantic concepts occurring at the paragraph level. The occurrence of semantic concepts in certain paragraphs, which are represented by matrix columns, can easily be visualized by coloring the respective matrix cells. Weiss-Lijn et al. (2003, p. 117) underlined that high-quality semantic metadata is essential to implement and evaluate the novel visualization technique: Unlike low-quality metadata that can be produced with relatively little human involvement, the required high-quality semantic metadata was “painstakingly hand crafted.” In particular, Weiss-Lijn et al. established and iteratively refined a taxonomy of domain-specific concepts. Subsequently, all paragraphs of two corporate text archives were manually annotated by making explicit the concepts occurring therein.

**Facilitating Manual Semantic Markup** Handschuh and Staab (2003b,c) introduced the

annotation framework CREAM, whose acronym stands for CREATING METADATA FOR THE SEMANTIC WEB. This conceptual framework facilitates the semantically accurate and ontology-based annotation of resources published on the Web. Handschuh and Staab aimed at overcoming the ‘annotation bottleneck,’ namely, the hardly existing willingness to provide the required explicit and formal metadata. This phenomenon poses an enormous challenge in the process of transforming the envisioned Semantic Web (cf. Berners-Lee et al., 2001) into omnipresent reality. CREAM specifies components for (i) manual annotation of existing resources, (ii) authoring of semantically enriched resources, and (iii) semi-automated markup of resources (cf. Subsection 2.3.2). The reference implementation ONTOMAT-ANNOTIZER is a client application of the freely available KAON tool suite (cf. Maedche and Staab, 2003, p. 29). An extended version is part of the commercial ONTOPRISE TOOL SUITE (cf. Fensel, 2004, pp. 83–84). The free reference implementation only supports the markup of semi-structured resources, such as HTML files. Nevertheless, the semantic annotation of unstructured texts is conceptually encompassed by this framework as well (Handschuh and Staab, 2003c, p. 29).

The conceptual framework CREAM is designed to satisfy requirements for successful semantic annotation, which reflect the authors’ experience gained in several knowledge acquisition projects (Handschuh and Staab, 2003c, pp. 26–28). Ensuring consistent markup, using proper references, and avoiding redundant markup are the top three out of nine prerequisites for successfully creating formal and semantically accurate metadata. Markup consistency requires annotators to adhere to a given ontology because only well-defined formal annotations allow for effective knowledge sharing and automated reasoning. In addition, real-world entities, like human beings, must be referenced by unique identifiers within the entire knowledge base. Otherwise, captured knowledge about particular entities cannot be completely retrieved from the knowledge base. To make the reuse of captured knowledge possible, collaborative annotation systems should prevent users from unknowingly providing redundant markup of already tagged resources.

Tallis et al. (2001, 2002) argued that benefits of semantic markup mostly accrue to agent-assisted consumers of documents published on the Semantic Web. In contrast, content producers typically lack the motivation to put in extra annotation effort. To simplify the costly production of hand-crafted semantic metadata, Tallis et al. designed and implemented the BRIEFING ASSOCIATE plug-in that augments Microsoft’s widely used POWERPOINT application. This plug-in enables authors of POWERPOINT presentations to simultaneously compose the actual content and the corresponding formal, ontology-based metadata. Unlike the reference implementation of CREAM, BRIEFING ASSOCIATE takes advantage of a popular commercial off-the-shelf application and thereby reduces the required initial training of users. Tallis (2003) and Eriksson (2007) introduced plug-ins that support semantic annotation in WORD and PDF documents, respectively. BRIEFING ASSOCIATE and SEMANTICWORD are representatives of highly specialized annotation tools. By contrast, GATE is a flexible, component-based architecture for natural language processing (cf. Cunningham, 2002). Specifically, the GATE MANUAL ANNOTATION TOOL supports theory-neutral, format-independent text annotation and thus enables authors to

insert arbitrary semantic markup into texts as well.

To persuade content producers into semantically tagging documents, Tallis et al. (2002) suggested offering them value-adding services that exploit semantic metadata and thus demonstrate their usefulness. This strategy resembles the instant gratification approach pursued in the MANGROVE project to entice non-technical people to semantically annotate their resources (McDowell et al., 2003). Besides offering a robust and easy-to-use annotation tool, providing instant gratification to content authors is a key concept in MANGROVE. Concretely, newly marked-up content is immediately published and consumed by semantics-based Web services that show the utility of annotated resources. Analogously, Motta et al. (2000) emphasized the necessity of identifying concrete use scenarios and added value prior to beginning ontology-driven document enrichment.

**Assessment** Despite the costs involved, semantic text annotation by hand is the preferred technique in a variety of domains. Substituting costly manual with (semi-) automated markup techniques, which provide semantic markup of comparably high quality, is therefore an open research challenge. Our framework for semantic tagging of large, domain-specific text archives comprises a semi-automated knowledge discovery phase followed by a fully automated application phase. If the DIASDEM framework is capable of maintaining equivalently high markup quality, it can easily complement applications that currently rely on hand-crafted metadata, such as GRIDVIS (cf. Weiss-Lijn et al., 2003) introduced above. Nevertheless, our framework should only be applied in a well-defined and value-adding (i.e., benefits outweigh costs) scenario because experienced domain and KDT specialists are involved in the initial knowledge discovery phase.

Unlike the framework and tools reviewed above, our explicit and informal approach to semantic tagging does not incorporate a priori established, formal ontologies as input. This characteristic feature is discussed after reviewing research into semi-automated, mostly formal text annotation in the next subsection.

### 2.3.2 Semi-Automated Semantic Text Annotation

Unlike truly automated techniques, semi-automated text annotation tools purposefully incorporate knowledge supplied by human experts into the tagging process. Instead of having to manually mark up entire text archives, domain and KDT specialists typically annotate training documents as input for machine learning algorithms, supervise a knowledge discovery process, or assess the markup quality to adjust parameters of annotation algorithms. In this subsection, we consider approaches to semantic text annotation as being semi-automated if they require considerable human intervention in any phase of the proposed process. We classify representative approaches to semi-automatic semantic text annotation into (i) methods chiefly based on information extraction algorithms and (ii) methods mainly based on natural language processing techniques. Furthermore, Reeve and Han (2005), Uren et al. (2006), and Han et al. (2007, pp. 443–449) provided comprehensive surveys of existing semantic annotation platforms.

**Focus on Information Extraction** AMILCARE is an adaptive information extraction system based on supervised rule induction (Ciravegna et al., 2003; Ciravegna and Wilks, 2003). This system is embedded within the MELITA annotation tool for human-centered semantic markup of textual resources. In particular, Ciravegna et al. introduced a methodology of seamless interaction between human annotators, the annotation interface, and the information extraction algorithm that aims at providing non-intrusive and timely support for human annotators. Consequently, annotators do not directly interact with an IE system, but they merely accept, correct, or decline automatically created annotation suggestions within their usual markup environment. The IE-enhanced annotation process is split into two main phases. While the user annotates named entities in the initial training phase, AMILCARE induces information extraction rules in the background. These rules are utilized to make markup suggestions in the second phase, which is denoted ‘active annotation with revision.’ In this phase, users are encouraged to actively correct erroneous markup suggestions. User feedback augments the training documents and triggers a re-execution of the rule induction algorithm.

Given appropriate training documents, AMILCARE is capable of extracting arbitrary named entities, such as names of persons, locations, dates, or phone numbers (Ciravegna and Wilks, 2003, pp. 114–117). Hence, this information extraction system offers a flexible support for identifying and annotating these named entities in text documents. Ciravegna et al. (2003, p. 159) extracted named entities from rather structured seminar announcements and concluded that AMILCARE and MELITA substantially contributed towards reducing the “burden of manual annotation.” Furthermore, S-CREAM (Handschuh et al., 2002) and MNM (Vargas-Vera et al., 2002) successfully integrated the AMILCARE information extraction system to semi-automatically markup named entities. Users of both S-CREAM and MNM approve identified named entities before the tools generate explicit and formal semantic markup adhering to a pre-defined ontology.

The SEMANTIC CONTENT ORGANIZATION AND RETRIEVAL ENGINE (SCORE) is a comprehensive and integrated tool set designed to facilitate the development of analysis-oriented Semantic Web applications (Sheth et al., 2002). SCORE provides facilities to define ontological components, as well as to extract, normalize, store, integrate, and query ontology-based metadata. In particular, SCORE exploits semantic metadata to associate semantically related, but not directly connected, content. Establishing semantic links between heterogeneous resources requires a tight integration of document management, knowledge management, and semantic technology. In this context, the SEMANTIC ENHANCEMENT ENGINE (SEM) pre-processes heterogeneous content, employs supervised learning techniques to determine the topicality of documents, and extracts contextually relevant semantic metadata (Hammond et al., 2002). Semantic text annotations are explicitly and informally stored in XML files conforming to specific XML document type definitions. Other formal ontology languages like the W3C-proposed Web Ontology Language OWL (cf. Fensel, 2004, pp. 44–46) can be supported in principle.

Prior to extracting metadata, automated text classification (cf. Sebastiani, 1999) is employed in SEM to assign retrieved documents into pre-defined thematic categories

(Hammond et al., 2002). For each topic category, contextually relevant semantic metadata are defined in a topic-specific ontology. SEM executes information extraction algorithms to semantically enhance documents by making explicit topic-related entities and their relationships. Since text classification requires human-generated rules or manually tagged training documents, we consider SCORE to be semi-automated annotation system. In addition, topic-specific IE techniques typically necessitate hand-coded regular expressions. However, Sheth et al. (2002, pp. 85–86) reported that only three full-time knowledge engineers implement and maintain “a few hundred Web-based extractors.”

Abolhassani et al. (2003) distinguished between macro- and micro-level text markup. The former corresponds to structural markup, as defined in Section 1.2, whereas the latter denotes semantic markup of single words or word groups. Within the VASARI project, Abolhassani et al. focused on micro-level markup and applied information extraction techniques to semantically annotate plain texts published in encyclopedias of art. After processing, for example, introductory articles about artists, informal semantic XML tags make explicit names of artists, places of their birth, and detailed descriptions of their achievements. The VASARI project adopts the knowledge engineering approach to information extraction and is thus considered to support semi-automatic text annotation. After having interactively and iteratively defined domain-specific extraction rules, however, thematically related texts can be automatically annotated.

**Focus on Natural Language Processing** Although Web pages are generally considered to be semi-structured, they often contain passages of unstructured text without embedded, semi-structuring HTML markup. Buitelaar and Declerck (2003) thus stressed the need for language technologies to establish the Semantic Web (cf. Berners-Lee et al., 2001) on a large scale. By determining the inherent semantic structure, natural language processing transforms text into a representation suitable for subsequent information extraction and content-descriptive markup, respectively. According to Buitelaar and Declerck, natural language processing techniques (e.g., morphological analysis, part-of-speech tagging, chunking for phrase identification, and word sense tagging) are important pre-processing steps to generate explicit and formal text markup for Semantic Web applications. In fact, all IE-focused techniques for semantic annotation surveyed above do perform linguistic pre-processing to some limited extent because the actual information extraction algorithms cannot be applied to plain natural language text.

Volk et al. (2002) as well as Buitelaar and Declerck (2003) integrated linguistic and semantic annotation to associate textual content with domain-specific, ontology-based knowledge. The MUCHMORE framework facilitates conceptual, cross-language information retrieval in a corpus of medical abstracts. For each abstract, multiple layers of in-depth linguistic and semantic annotations are stored in a single XML file conforming to a project-specific XML document type definition (cf. Vintar et al., 2002). Thereby, explicit and informal semantic markup is generated in MUCHMORE. More specifically, semantic tagging corresponds to mapping terms onto concepts that are defined in domain-specific, informal or formal ontologies (e.g., thesauri, semantic lexicons, or semantic networks). In

addition, ontology-defined relations between concepts are made explicit. The approach pursued by Buitelaar and Declerck is considered to be semi-automated because linguistic algorithms must be manually parameterized for each new domain prior to automatically annotating domain-specific texts. After tagging German abstracts, Volk et al. concluded that linguistic processing, in particular lemmatization and compound analysis, is a prerequisite for creating high-quality semantic markup.

The XDOC DOCUMENT SUITE is a tool collection for linguistic text processing (Rösner and Kunze, 2003; Kunze, 2006). XDOC comprises several modules for text pre-processing, detection of linguistic units (e.g., sentences and titles), part-of-speech tagging, syntactic parsing, and semantic analysis. Semantic tagging in XDOC corresponds to mapping terms onto concepts defined in a domain-specific semantic lexicon. In particular, case frame analysis is conducted to resolve ambiguous term senses and to map disambiguated words onto concepts. Furthermore, relations between concepts are identified by exploiting syntactic characteristics of domain-specific sublanguage. Since linguistic text annotations are stored in tool-specific XML documents, this application generates explicit and informal semantic markup. Rösner et al. (2004) outlined a use case for XDOC in the context of transforming and enriching textual content for Semantic Web applications. The XDOC DOCUMENT SUITE is a semi-automated text annotation tool because it requires domain-specific parameterization.

Analogous to the XDOC DOCUMENT SUITE, Li et al. (2001) exploited syntactic characteristics of domain-specific sublanguage to semantically annotate sentences. The authors focused on sublanguage texts with limited vocabulary, patterns in vocabulary usage, and rare semantic ambiguities. Essentially, Li et al. proposed to syntactically parse sentences and subsequently employ supervised learning algorithms to infer a mapping between the syntactical sentence structure and RDF (cf. Fensel, 2004, pp. 19–21) statements, which constitute explicit and formal markup. However, this approach requires a considerable human effort to create a training corpus by manually labeling syntactically parsed sentences with RDF statements. The authors demonstrated the feasibility of this approach by annotating 500 sentences from approx. 300 clothes descriptions.

Besides analyzing company descriptions and technical documentation, Rösner and Kunze (2003) employed the XDOC DOCUMENT SUITE to semantically annotate German autopsy protocols. In this medical sublanguage, a very telegrammatic style is dominant (cf. Rösner and Kunze, 2003, p. 123). Focusing on a related medical domain, Moore and Berman (2001) presented a technique to convert textual pathology reports into semantically marked-up XML documents. Moore and Berman employed natural language processing techniques and a medical thesaurus to map single terms and noun groups onto medical concepts. Subsequently, medical concepts serve as XML tags that semantically annotate the corresponding terms in an explicit and informal way.

**Assessment** Analogous to the research reviewed above, our framework pursues a semi-automated approach to semantic text annotation. The initial, interactive knowledge discovery phase incorporates human domain knowledge to ensure a consistently high markup

quality. In the second, batch-oriented phase, domain-specific text documents are automatically transformed into semantically tagged XML documents. Like the surveyed research on semi-automated text annotation, our framework is also focused on domain-specific text archives to exploit characteristics of the respective sublanguage.

As defined in Section 1.2, our framework derives archive-specific XML tags that concisely summarize the content of structural text units. Unlike research into purely IE-based text annotation, the DIASDEM framework is hence not restricted to marking up certain named entities only. The identified named entities rather augment content-descriptive XML tags. Thereby, extracted named entities are placed in a semantic context. However, we do not aim at inferring the specific roles of named entities within annotated text units. As the primarily linguistic research into semantic markup, our approach maps structural text units comprising multiple terms onto high-level semantic concepts. In contrast to annotation tools focusing on natural language processing, the DIASDEM framework additionally incorporates conventional information extraction techniques. Thereby, our approach combines the information extraction thread of research into text annotation and the linguistic aspect of mapping textual content onto semantic concepts. We advocate basic linguistic text pre-processing (e.g., tokenization and lemmatization), but we refrain from employing sophisticated NLP techniques, such as morphological analysis or syntactic parsing, to reduce complexity.

A large proportion of research into semi-automated semantic tagging is targeted at bootstrapping the Semantic Web. In this context, the creation of explicit and formal semantic markup is imperative since machine-understandable markup is a prerequisite for automated reasoning. The Semantic Web approach to text annotation hence necessitates pre-defined formal ontologies. In contrast, our framework transforms text documents into explicitly and informally annotated XML documents and does not require a formally defined ontology. Furthermore, the DIASDEM framework incorporates an explorative knowledge discovery approach to reduce human intervention. Unlike the representative IE-based and NLP-focused work introduced above, our framework neither requires semantically tagged training documents nor relies on existing sophisticated semantic lexica. Instead, explorative knowledge discovery techniques (i.e., clustering algorithms) are utilized to discover domain-specific, content-descriptive XML tags, which are subsequently aggregated into a concept-based XML document type definition.

### **2.3.3 Automated Semantic Text Annotation**

This subsection introduces representative research into automated semantic text markup. In contrast to both manual and semi-automated approaches, we subsequently focus on techniques that do not incorporate human knowledge into the tagging process. Although mostly relying on an existing and mainly semi-automatically set up knowledge base, domain-independent techniques are considered to support automatic markup. Unlike their semi-automated counterparts, these techniques can be straightforwardly applied to text archives comprising general content without the need for re-parameterization. Existing

automatic markup tools mainly extract and semantically annotate named entities.

**Focus on Information Extraction** AERODAML utilizes information extraction techniques to alleviate the human effort required to semantically mark up textual resources published on the Web (Kogut and Holmes, 2001; Kogut and Heflin, 2003). Given a URI, the corresponding resource is downloaded and automatically marked up by explicit and formal annotations in the DAML (cf. Fensel, 2004, p. 44) knowledge representation language. AERODAML annotates proper nouns, common nouns, and other named entities that instantiate concepts and relationships defined in a DAML ontology. In addition to mapping typical named entities (e.g., persons) onto concepts, common nouns (e.g., ‘gun’) are identified as instantiations of the respective concepts (e.g., ‘weapon’). Furthermore, pre-defined relationships, such as ‘person works for company’, can be instantiated as well. AERODAML is built on top of a proprietary text mining system, utilizes domain-independent rules for proper noun extraction, and incorporates the lexical database WORDNET (cf. Fellbaum, 1998) into its domain-independent knowledge base. Inspired by computer-assisted translation, Kogut and Holmes proposed to automatically mark up textual resources by default. Only if necessary, domain experts may add new or correct generated annotations via a markup tool of their choice. AERODAML can also be customized to create domain-specific markup in a semi-automated way.

The semantic text annotation platform KIM is based on GATE components (cf. Cunningham, 2002) for text pre-processing and information extraction (Popov et al., 2003; Kiryakov et al., 2003; Popov et al., 2004). KIM automatically maps named entities onto concepts that are defined in the domain-independent KIM ONTOLOGY. The KIM ONTOLOGY is encoded in the formal ontology language RDFS (cf. Fensel, 2004, pp. 37–39) and comprises a taxonomy of approx. 250 concepts (e.g., ‘entity’, ‘object’, and ‘product’) along with their attributes and relationships among them. In addition, information extraction in KIM takes advantage of a massive knowledge base, which includes lexical resources and approx. 80,000 important named entities as instantiations of the KIM ONTOLOGY. Due to the existing KIM ONTOLOGY and the extensive knowledge base, KIM is capable of automatically inserting explicit and formal semantic annotations into general text documents. The system achieved 86% average precision at 82% average recall in annotating six general-purpose named entities (i.e., date, person, organization, location, percent, and money) in 100 news articles. After extending the knowledge base appropriately, KIM is also capable of marking up domain-specific text documents. However, domain-specific text annotation is considered to be a semi-automated technique because it requires a considerable amount of human involvement.

Dill et al. (2003) introduced SEEKER, a component-based framework for large-scale text analytics. SEMTAG is built on top of this framework and allows for an automated semantic annotation of pre-defined named entities in large text corpora. Dill et al. aimed at bootstrapping the Semantic Web (cf. Berners-Lee et al., 2001) by automatically annotating textual content with a set of domain-independent, widely usable named entities. SEMTAG strongly relies on the semi-automatically compiled TAP KNOWLEDGE BASE

(cf. Guha and McCool, 2003) that comprises lexical and taxonomic information about approx. 72,000 important entities, such as ‘musicians’, ‘movies’, and ‘athletes’. Nevertheless, SEMTAG is considered to be an automated annotation tool because this application marked up 434 million named entities in 264 million Web-based resources and achieved a tagging accuracy of around 82%. This Web-scale experiment required negligible human involvement (i.e., approx. 700 yes/no decisions) because Dill et al. automatically resolved ambiguities by executing a taxonomy-based disambiguation algorithm. After identifying named entities listed in the TAP KNOWLEDGE BASE, ambiguous strings (e.g., ‘Michael Jordan’ may be a basketball player or a statistician) are disambiguated on the basis of context terms. All generated annotations are represented as RDFS (cf. Fensel, 2004, pp. 37–39) statements. These annotations are stored separately from marked-up content and thus facilitate flexible semantic Web services.

Gruhl et al. (2004) gave an overview on WEBFOUNTAIN, an architecture for very large-scale text analytics that is based on SEEKER and SEMTAG. The platform allows access to different sources, supports the scalable deployment of document-level text enrichment and corpus-level analysis, as well as the creation of Web services that serve end-user applications. Alba et al. (2006) discussed challenges and lessons learned when building this Web-scale system that collects, analyzes, stores, and serves billions of documents.

**Assessment** Ideally, human involvement in generating semantically enriched content should be completely eliminated to facilitate large-scale semantic services. As illustrated by the research reviewed above, however, fully automated text annotation techniques are currently capable of identifying and marking up domain-independent named entities only. These techniques take advantage of the vast literature on extracting named entities and relational tuples from unstructured text (cf. Subsection 2.2.2). In contrast, our semi-automated framework for semantic XML tagging pursues different objectives in two important aspects. Firstly, we employ explorative knowledge discovery techniques to find content-descriptive XML tags that briefly summarize the textual content of text units. Thereby, textual content is mapped onto high-level semantic concepts, such as ‘AcquisitionAnnouncement’, and extracted named entities are placed in the context of conceptual tags. Secondly, our framework is designed to transform domain-specific, as opposed to general, text archives into XML documents conforming to an archive-specific XML document type definition. In contrast, all reviewed automatic markup tools assume the existence of domain-independent knowledge in a lexical database or generic ontology, which was manually or semi-automatically codified or acquired by a third party. The existence of appropriate knowledge bases can hardly be assumed for a variety of specific domains. Hence, the development of fully automated markup techniques for domain-specific texts remains an open research challenge (cf. Reeve and Han, 2005, p. 1434).

## 2.4 Schema Discovery in Marked-Up Text Documents

The DIASDEM framework transforms domain-specific texts into semantically annotated XML documents and semi-structured data, respectively. As introduced in Section 1.1, ‘self-describing’ semi-structured data do not adhere to a rigid and explicitly defined schema, but rather exhibit some structure and may implicitly conform to a loose schema (Buneman, 1997; Wang and Liu, 2000, p. 353). In our framework, however, each collection of semantically tagged XML documents conforms to a concept-based XML document type definition. This automatically derived XML DTD serves as a context-free grammar for all XML documents and thus constitutes a loose schema that specifies the internal structure of annotated XML documents (cf. Abiteboul et al., 2000, pp. 38–45).

Albeit imposing only loose structural constraints, XML document type definitions offer substantial advantages in storing, processing, and querying the respective collections of XML documents (Garofalakis et al., 2003, p. 24). In principle, these benefits accrue from the fact that a document type definition specifies a regular expression pattern for each DTD element, which must be matched by all sequences of DTD elements occurring therein. Unfortunately, XML documents do not have to conform to a document type definition (cf. World Wide Web Consortium, 2000). Consequently, the question of deriving an XML document type definition from a given collection of XML documents constitutes a research issue of practical importance. The discovery of a typically loose schema for marked-up textual data is an issue relevant to our research questions although we limit ourselves to establishing an XML document type definition that enumerates frequently occurring thematic concepts and associated named entity types.

Since XML documents are instances of semi-structured data in many respects (cf. Garofalakis et al., 2003, p. 24), we firstly survey representative approaches to schema discovery in conventional semi-structured data. Subsequently, techniques for inferring a document type definition from a collection of SGML or XML documents are reviewed. Due to the missing relevance to our research, we do not discuss approaches that require an existing DTD as input. This criterion is, for example, met by research into inferring a reduced DTD from a complex DTD and a sample of marked-up documents (e.g., Bia et al., 2001) and by techniques for aggregating several document-specific source DTDs into one common DTD (e.g., Sengupta and Puro, 2000). Finally, this subsection is concluded by assessing the reviewed work with respect to our pursued research questions.

**Schema Discovery in Semi-Structured Data** Collections of semi-structured documents are usually modeled as graphs (cf. Buneman, 1997; Abiteboul et al., 2000, pp. 11–13). For example, the OBJECT EXCHANGE MODEL (OEM; Papakonstantinou et al., 1995) can be conveniently represented by a labeled, directed graph. Vertices correspond to uniquely identifiable OEM objects in OEM graphs, and edges convey semantic information about the relationship between connected objects via descriptive textual labels (e.g., `has_company`). Labeled vertices without outgoing edges represent typed atomic values (e.g., the string "Giant Foo Corp."). The remaining vertices represent complex objects

comprising a set of subordinate OEM objects.

Goldman and Widom (1997) introduced an algorithm to derive concise and accurate structural summaries (i.e., *DATAGUIDES*) for collections of OEM encoded semi-structured documents. Serving as collection-specific schemata, *DATAGUIDES* enable users to grasp an understanding of the collection structure prior to formulating queries. Analogously to data dictionaries in relational databases, *DATAGUIDES* facilitate optimized query processing by providing structural metadata. Unlike data dictionaries, *DATAGUIDES* are dynamically derived from the data themselves and do not enforce any constraints on the document collection. *DATAGUIDES* are OEM graphs that reflect structural features (i.e., labeled edges that connect placeholder vertices) occurring at least in one document. In the worst case, constructing a *DATAGUIDE* for arbitrary OEM graphs is exponential in time and space with respect to the graph size. When applied to OEM trees, however, the algorithm has a linear complexity.

Experimental results indicate that *DATAGUIDES* are significantly smaller than typical OEM source collections (Goldman and Widom, 1997, p. 438). Nevertheless, the size of accurate schemata, like *DATAGUIDES*, may be quite large unless the described documents share a very similar structure. Complex schemata tend to exacerbate interactive structure exploration as well as automated query optimization. To balance the need for compact schemata against schema accuracy requirements, Goldman and Widom (1999) as well as Nestorov et al. (1998) presented distinct approaches to extracting an approximate, yet sufficiently detailed, schema from semi-structured documents.

Domain-specific semi-structured documents are often similarly, though not identically, structured. To reveal frequently occurring structural elements, Wang and Liu (2000) proposed an algorithm that discovers typical graph structures in a collection of OEM encoded documents. The authors extended the *OBJECT EXCHANGE MODEL* (Papakonstantinou et al., 1995) such that vertices of the directed OEM graph may represent complex objects that comprise either an ordered list or an unordered set of subordinate OEM objects. Furthermore, Wang and Liu generalized the problem of discovering association rules supported by a minimum number of transactions (cf. Agrawal et al., 1993; Agrawal and Srikant, 1994). Consequently, the minimum frequency of interesting structural elements must be specified by the user. Subsequently, the algorithm searches partial graph structures (i.e., so-called tree expressions) whose frequency of occurrence in all source OEM graphs exceeds the user-supplied threshold.

**Schema Discovery in SGML/XML Documents** Since an archive of SGML documents may lack a document type definition, Ahonen (1995, 1996) studied the problem of automatically establishing a DTD for a given collection of SGML documents. Ahonen conceptualized and validated a generic method for generating a context-free grammar from a set of marked-up documents. Initially, the structure of input documents is completely captured by creating a finite-state automaton for each structural element. Thereafter, machine learning techniques are employed to generalize these automata and subsequently convert them into regular expressions. Finally, a grammar is constructed from the result-

ing regular expressions that makes explicit the internal document structure. Specifically, this method derives archive-specific SGML document type definitions, which (unlike trivial ones) are neither too generalized nor too restrictive.

Shafer (1996) introduced a method for automated inference of SGML DTDs, which resembles the approach pursued by Ahonen (1996). Shafer employed the proprietary `GRAMMAR BUILDER ENGINE` to infer a grammar from tagged documents. Firstly, structural rules representing SGML tags and their nestings are extracted from input documents. Subsequently, sophisticated heuristics are applied to combine, generalize, and reduce these structural rules before the resulting grammar is finally output as an SGML DTD. Both Shafer and Ahonen approached the problem of generating a DTD from a set of SGML documents as an application of deterministic grammar inference. In contrast, Young-Lai and Tompa (2000) advocated a technique based on stochastic grammar inference, which takes frequency information from the training documents directly into consideration. According to Young-Lai and Tompa, stochastic grammar inference scales better to large collections and creates models with richer semantics.

DTD-MINER automatically extracts a document type definition from XML documents in a three-step process (Moh et al., 2000b,a). Firstly, the hierarchical internal structure of each XML document is transformed into an ordered  $n$ -ary tree whereby tags are modeled as labeled vertices. Since XML documents may be structurally dissimilar even in the same archive, Moh et al. (2000a) suggested utilizing a graph-based clustering algorithm to identify structurally homogeneous sub-collections. Secondly, all document trees are merged into one ordered, directed, acyclic graph that represents the overall structural information about XML tags, their hierarchical relationships, and attributes. Finally, heuristic rules are applied to modify this spanning graph before generating an XML DTD. In particular, heuristics distinguish optional from mandatory DTD elements, identify repeatable DTD elements, and find appropriate groups of repeatable DTD elements.

Garofalakis et al. (2000, 2003) described the XTRACT system, which comprises algorithms to infer an XML document type definition from XML documents. The authors specifically focused on exploiting the full expressive power of regular expressions supported by the DTD syntax to automatically establish both concise and precise document type definitions. To that end, generalization and factorization algorithms create several candidate DTDs by employing heuristics and logic optimization techniques. However, the most important step involves choosing the ‘best’ document type definition from all candidate DTDs by applying the minimum description length principle rooted in information theory. Thereby, the tradeoff between ‘conciseness’ and ‘preciseness’ of DTDs is taken into consideration by balancing these two desired characteristics of XML document type definitions. Min et al. (2003) introduced a simplified and heuristic approach to DTD extraction and reported that this technique derived highly concise and accurate real-world document type definitions up to 20 times faster than XTRACT.

Ling et al. (2005) introduced the semantically rich `OBJECT RELATIONSHIP ATTRIBUTE DATA MODEL FOR SEMI-STRUCTURED DATA (ORA-SS)` to overcome limitations of existing data models, such as `OEM` or `DATAGUIDES`. According to Ling et al. (2005,

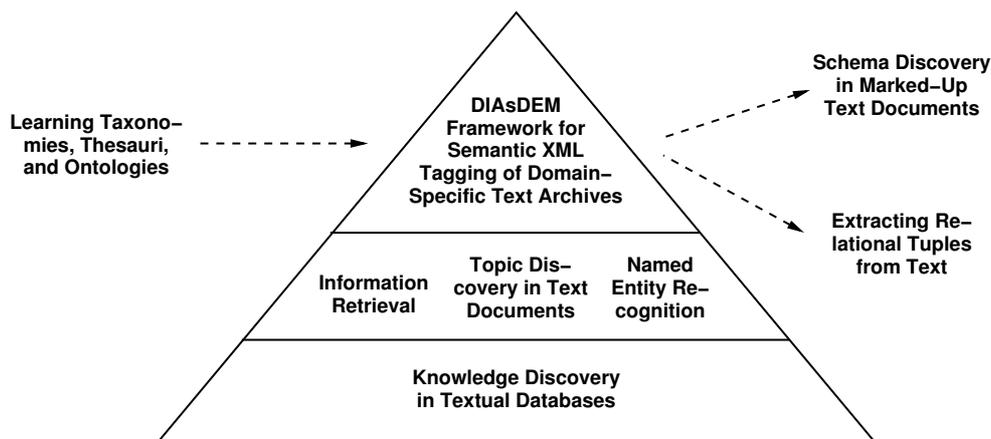
pp. 57–58), ORA-SS combines concepts central to semi-structured data models (e.g., references, ordering of object classes and attributes, and document instances) with traditional data modeling concepts, like binary and  $n$ -ary ( $n > 2$ ) relationships, participation constraints on object classes participating in relationships, as well as inheritance hierarchies between object classes. This data model consists of a schema diagram, an instance diagram, a functional dependency diagram, and an inheritance hierarchy diagram. Besides defining ORA-SS, Ling et al. (2005, pp. 56–75) proposed an algorithm designed to extract an ORA-SS schema from a semi-structured data instance, such as an XML document. The authors suggested employing a semi-automated, rule-based algorithm that is highly focused on data-centric, as opposed to text-centric, XML documents. Subsequent to generating an initial schema in the first extraction step, user input is required to verify schema properties derived from the data in the second, schema-refining step.

**Assessment** The rapid emergence of SGML and XML for encoding various kinds of data sparked a large body of research into schema discovery in marked-up documents. Since neither SGML nor XML documents have to be accompanied by a document type definition, automatically inferring a DTD for a given collection of marked-up documents is an efficient means of enjoying the considerable DTD-induced benefits.

As illustrated by the representative work reviewed above, schema discovery in marked-up documents is mostly a solved research issue. A variety of experimentally validated methods is available to researchers and practitioners. These techniques often rely on graph-based representations and sophisticated heuristics to infer an intuitively usable, both concise and accurate DTD from marked-up documents. These methods are complementary to the research questions addressed in this work since we primarily aim at marking up unstructured text units with domain-specific, conceptual XML tags discovered beforehand. Nevertheless, semantic XML tags are aggregated into a concept-based XML document type definition. Establishing an enumerative XML document type definition is sufficient because our framework does not encompass the creation of nested markup. Once the insertion of nested XML tags is fully supported in the future, however, an existing method for automatic inference of non-trivial XML document type definitions can easily be incorporated into our framework.

## 2.5 Summary

In this chapter, we have reviewed work related to semantic XML tagging, as defined in the introductory Section 1.2. We have given an overview of fundamental text processing research areas, as well as specific research into topic discovery, semantic text annotation, and schema discovery in marked-up documents. However, the assessments of individually surveyed research areas remain to be integrated. To that end, we delineate our approach from directly related work on semantic text annotation and highlight its multi-disciplinarity.



**Figure 2.2:** Fundamental and Complementary Research Areas of the DIAsDEM Framework for Semantic XML Tagging of Domain-Specific Text Archives

**Semantic XML Tagging** Unlike other approaches to semantic text annotation, we investigate the content-descriptive markup of fine-grained structural text units, such as sentences or paragraphs. In principle, our semi-automated framework is capable of enhancing applications that, for quality reasons, currently rely on manual text annotation. Current techniques for automated semantic markup are restricted to the annotation of generic named entities and often strongly rely on existing knowledge bases. After going through an initial knowledge discovery phase, our framework instead allows for an automated conversion of domain-specific texts into conceptually marked-up XML documents.

For each new domain, an interactive knowledge discovery process must be accomplished once. This process is designed to discover domain-specific, concept-based XML tags for a given text collection and to derive a matching concept-based XML document type definition. Unlike most research into semi-automated markup, we neither solely concentrate on named entity extraction nor restrict ourselves to mapping textual content onto predefined semantic concepts. Instead, our framework combines research into topic discovery (i.e., identifying semantic XML tags) and named entity recognition (i.e., augmenting XML tags). Subsection 2.2.2 has revealed that the originally posed research question related to the extraction of domain-specific named entities from text units can be considered solved by the information extraction research community for the scope of this work. Furthermore, our approach does not require a formal ontology as input because we aim for informally annotating text documents. Instead, we suggest incorporating domain knowledge in the form of a controlled vocabulary, which can usually be established with less effort. Subsequent to the knowledge discovery phase, documents from the same domain are automatically marked up without any human intervention.

**Multi-Disciplinary Approach** Due to the complexity of inferring semantics from text documents, we adopt an inherently multi-disciplinary approach to semantic XML tagging. As the pyramid in Figure 2.2 illustrates, knowledge discovery in textual databases is the fundamental discipline. To attain our objectives, we selectively utilize methods from information retrieval (e.g., text pre-processing), topic discovery (e.g., employing unsupervised learning techniques), and named entity recognition. Furthermore, dashed arrows indicate three major complementary research areas. For example, complementary research into automatic learning of controlled vocabularies may be utilized to further reduce human efforts in the knowledge discovery phase. In addition, complementary work on schema discovery in marked-up documents and extraction of relational tuples from text might be exploited to post-process semantically tagged XML documents.

**Intermediate Conclusion** The semi-automatic DIASDEM framework occupies an important and clearly-defined niche in the continuum of requirements for semantic XML tagging of textual data. Specifically, our framework enables explicit, informal, and high-quality semantic markup of large, domain-specific text archives whose documents share a common sublanguage. Distinctive characteristics of our framework are the application of explorative knowledge discovery techniques to identify thematic concepts at the text unit level as well as the combination of content-descriptive text annotation and named entity extraction. Our framework is introduced in detail in the next chapter.



## 3 DIAsDEM Framework

Chapter 3 outlines the DIAsDEM framework for semantic XML tagging of domain-specific text archives. This conceptual framework provides a high-level solution to the research questions raised in the introduction. In the next section, we clarify the most important terminology for the scope of this work. Section 3.2 presents the principal objectives of our framework and gives a concise overview thereof. The subsequent Sections 3.3 and 3.4 outline the semi-automated knowledge discovery phase and the fully automated knowledge application phase, respectively.

### 3.1 Terminology

According to Definition 1, the central term semantic XML markup denotes XML tags whose names explicitly convey informal metadata about the meaning of marked-up text units. Semantic XML tags concisely describe concepts that domain experts typically associate with marked-up text units. Optionally, attributes of semantic tags make explicit named entities occurring in annotated text units. The syntax of semantic XML markup is defined in a concept-based XML document type definition whereas the meaning of semantic XML markup is informally specified in the accompanying DTD documentation.

In this subsection, we define fundamental terms, such as text document, text unit, and semantically marked-up text document. Each fundamental, framework-specific term is accompanied by a corresponding abstract data type specified in Appendix B. An abstract data type (ADT) is a class “of objects whose logical behavior is defined by a set of values and a set of operations” (Dale and Walker, 1996, p. 3). Although ADTs are precisely defined, they are independent of any particular implementation in a programming language (Guttag and Horning, 1978, p. 27). The set of values encapsulated by an abstract data type can only be accessed and modified via the set of valid operations, which is referred to as the interface. This principle of hiding implementation details by providing a well-defined interface is known as encapsulation (cf. Loeckx et al., 1996, p. 11). We define and subsequently utilize<sup>1</sup> abstract data types related to semantic XML tagging of text archives because they allow for a high-level description of required data structures, as well as characteristic properties and operations.

---

<sup>1</sup>Notation: Let  $x: T$  denote a variable  $x$  of type  $T$  that can be either a primitive data type described in Appendix B.1 (i.e., Boolean, Integer, Real, Char, and arrays) or an abstract data type (e.g., String) specified in the remainder of Appendix B. If  $T$  is an abstract data type and  $o(\cdot)$  is an operation thereof,  $x.o(\cdot)$  denotes the execution of operation  $o(\cdot)$  on the instantiation  $x$  of abstract data type  $T$ .

Computer-accessible text documents may conform to a variety of internal format specifications. The DIAsDEM framework, however, abstracts from this diversity in formats and only considers archives that comprise plain text files. Since our framework is deliberately restricted to alphabet-based languages whose characters are interpreted from left to right, text documents<sup>2</sup> are modeled as sequences<sup>3</sup> of characters:

**Definition 2 (Text Document)** *A text document  $\check{t}$ : String is a sequence of characters.*

In particular, we abstract from specific character encoding schemes, but assume the existence of characters, such as the blank space (i.e., " "), letters (e.g., "A", "a", or "ä"), digits (e.g., "0"), punctuation marks (e.g., "."), and special characters (e.g., "\$", line feed, or tab stop). So-called whitespace characters, like the blank space or the tab stop, potentially separate natural language words (cf. Manning and Schütze, 1999, p. 125). The abstract data type TextDocument (see Appendix B.4 on page 220) encapsulates one text document. Given the exemplary text  $\check{t}_1$ : TextDocument and  $\check{t}_1.create("New York: The contract was signed on Dec. 30, 2006.")$ ,  $\check{t}_1.length() = 51$ ,  $\check{t}_1.char(9) = ":",$   $\check{t}_1.char(\check{t}_1.length()) = ".",$  and  $\check{t}_1.chars(38, 50) = "Dec. 30, 2006"$ .

The ADT TextArchive (see Appendix B.5 on page 221) encapsulates one text archive. Due to the possibility of duplicate documents, text archives are modeled as follows:

**Definition 3 (Text Archive)** *A text archive  $\check{a} := \langle \check{t}_1, \check{t}_2, \dots, \check{t}_{|\check{a}|} \rangle$  is a sequence of not necessarily distinct text documents such that  $\check{t}_i$ : TextDocument, where  $i = 1, 2, \dots, |\check{a}|$ .*

Instead of disambiguating the sense of single terms (cf. Manning and Schütze, 1999, pp. 229–263), our framework enables the semantic markup of structural text units:

**Definition 4 (Text Unit)** *Given the text document  $\check{t}$ : TextDocument, a text unit  $\check{u}$ : String is a sequence of contiguous characters in  $\check{t}$  such that  $\check{u} = \check{t}.chars(i, j)$ , where  $i = 1, 2, \dots, \check{t}.length()$  and  $j = i, i + 1, \dots, \check{t}.length()$ . A text unit represents a structural text component that consists of more than one natural language word.*

Structural text units can be fine-grained (e.g., sentences), coarse-grained (e.g., paragraphs), or even comprise entire text documents. The ADT TextUnit (see Appendix B.6 on page 221) encapsulates one text unit. Let  $\check{u}_1$ : TextUnit denote the text unit that is instantiated by  $\check{u}_1.create(\check{t}_1, 11, 51)$  to encapsulate the sentence  $\check{t}_1.chars(11, 51) = "The contract was signed on Dec. 30, 2006."$  Thus,  $\check{u}_1.length() = 41$ ,  $\check{u}_1.startIndex() = 11$ , and  $\check{u}_1.endIndex() = 51$ . As part of the document pre-processing prior to semantic tagging, text documents are decomposed into text units of the required granularity. The resulting decomposition is henceforth referred to as a text unit layer:

<sup>2</sup>Hereafter, text documents are synonymously referred to as textual data and text, respectively.

<sup>3</sup>Notation: Let the set  $X$  denote an arbitrary domain. A non-empty sequence  $x := \langle x_1, x_2, \dots, x_i \rangle$  is a tuple comprising  $i \in \mathbb{N}$  ordered elements  $x_1, x_2, \dots, x_i$  such that  $x_j \in X$ , where  $j = 1, 2, \dots, i$ . The number of elements  $|x| \in \mathbb{N}$  in sequence  $x$  (i.e., its length) is not necessarily constant. The  $j$ th element in sequence  $x$  is denoted by  $x[j]$ , where  $j = 1, 2, \dots, |x|$ . An empty sequence is denoted by  $\varepsilon$  ( $|\varepsilon| = 0$ ).

**Definition 5 (Text Unit Layer)** *Given the text document  $\check{t}$ : TextDocument, a text unit layer is a decomposition of text document  $\check{t}$  into structural text units. Text unit layer  $\check{r} := \langle \check{u}_1, \check{u}_2, \dots, \check{u}_{|\check{r}|} \rangle$ , where  $\check{u}_i$ : TextUnit and  $i = 1, 2, \dots, |\check{r}|$ , is a sequence of ordered and not necessarily contiguous, but non-overlapping, text units in text document  $\check{t}$ .*

One text unit layer is encapsulated by the abstract data type TextUnitLayer (see Appendix B.7 on page 222). In principle, text documents may be decomposed into multiple text unit layers that have different structural characteristics. For instance, the text unit layer  $\check{r}_1$ : TextUnitLayer decomposes text document  $\check{t}_1$  into dateline and news item, namely, "New York" and "The contract was signed on Dec. 30, 2006." Alternatively, text unit layer  $\check{r}_2$ : TextUnitLayer splits the same document into grammatical sentences and thus only comprises the text unit corresponding to the second string. To further conceptual clarity, our framework is deliberately restricted to process exactly one text unit layer per text document. Future extensions, however, are straightforward.

We now proceed by formalizing our notion of semantic XML markup. As informally introduced in Section 1.2, semantic markup associates text units with content-descriptive concepts and named entities occurring therein. To achieve conceptual clarity, however, we abstract from syntax-related issues of outputting marked-up textual data as XML documents in this subsection. Before defining semantically marked-up text units, the notions of concepts and named entities introduced in Section 1.2 are clarified as follows:

**Definition 6 (Concept)** *Given the domain-specific text archive  $\check{a}$ : TextArchive, a concept is a mental reference or thought that domain experts typically associate with a group of semantically similar text units occurring in text documents of  $\check{a}$ . A concept is represented by a symbol  $o \in O$  from the domain-specific set of concepts  $O := \{o_1, o_2, \dots, o_{|O|}\}$ .*

The abstract data types Concept (see Appendix B.8 on page 222) and SetOfConcepts (see Appendix B.9 on page 223) encapsulate one concept and one set of concepts, respectively. As emphasized in Section 1.2, there is a clear semiotic distinction between symbols on the one side and mental references, thoughts, or concepts on the other. Unless otherwise stated, however, we henceforth directly refer to symbol  $o \in O$  as the thematic concept  $o$  to simplify our notation. Alphanumeric labels of concepts (e.g., ConclusionOfContract associated with the concept conclusionOfContract  $\in O_1$ ) ultimately serve as names of semantic XML tags. Named entities are modeled as 2-tuples<sup>4</sup>:

**Definition 7 (Named Entity)** *Let  $P := \{p_1, p_2, \dots, p_{|P|}\}$  denote a set of domain-specific named entity types that represent abstract classes and generic numerical expressions. Given the text unit  $\check{u}$ : TextUnit, a named entity  $e := (p: P, \check{e}: \text{String})$  identified in*

<sup>4</sup>Notation: A  $k$ -tuple  $x := (x_1, x_2, \dots, x_k)$  comprises  $k \in \mathbb{N}$  ordered elements  $x_1, x_2, \dots, x_k$ . If the sets  $X_i$ , where  $i = 1, 2, \dots, k$ , denote  $k$  arbitrary domains, then  $x := (x_1: X_1, x_2: X_2, \dots, x_k: X_k)$  defines a  $k$ -tuple such that  $x_i \in X_i$ . The number of elements  $|x| = k$  in  $k$ -tuple  $x$  (i.e., its arity) is constant. Elements  $x_i$  of  $k$ -tuple  $x$  may take the null value (i.e.,  $x_i = \text{null}$ ) if  $x_i$  is unknown, inappropriate, or non-existent. The  $i$ th element in  $k$ -tuple  $x$  is denoted by  $x[i]$ .

text unit  $\check{u}$  is a 2-tuple comprising its named entity type  $p$  and the canonical form  $\check{e}$  of the instantiated named entity type  $p$  in text unit  $\check{u}$ . A set of distinct named entities is denoted by  $E := \{e_1, e_2, \dots, e_{|E|}\}$ .

Given the set of named entity types  $P_1 := \{\text{company, person, date}\}$ , the named entities  $e_1 := (\text{date, "2006-12-30"})$  and  $e_2 := (\text{date, "Day=30; Month=Dec; Year=2006"})$  can, for example, be extracted from the text unit "New York: The contract was signed on Dec. 30, 2006." Although named entities  $e_1$  and  $e_2$  refer to the same instantiation of named entity type `date` in this text unit, they have distinct canonical forms. Alphanumeric labels of named entity types, like `Date`, and canonical named entity forms, like "2006-12-30", ultimately serve as names and values, respectively, of attributes that represent the respective named entities within semantic XML tags. The extent to which named entities are normalized is application-dependent. The abstract data types `NamedEntityType` (see Appendix B.10 on page 223) and `SetOfNamedEntityTypes` (see Appendix B.11 on page 224) encapsulate one named entity type and one set of named entity types, respectively. Additionally, one named entity and one set of named entities is encapsulated by the abstract data types `NamedEntity` (see Appendix B.12 on page 224) and `SetOfNamedEntities` (see Appendix B.13 on page 225), respectively.

**Definition 8 (Semantically Marked-Up Text Unit)** *Given the parent text document  $\check{t}$ : `TextDocument`, a semantically marked-up text unit  $\hat{u} := (\check{u}: \text{TextUnit}, o: \text{Concept}, E: \text{SetOfNamedEntities})$  is a 3-tuple comprising the original text unit  $\check{u}$ , the domain-specific concept  $o$  that domain experts typically associate with text units similar to text unit  $\check{u}$ , and the set of named entities  $E$  extracted from text unit  $\check{u}$ .*

A semantically marked-up text unit is encapsulated by the abstract data type `SmuTextUnit` (see Appendix B.14 on page 225). For example, the semantically marked-up text unit  $\hat{u}_1: \text{SmuTextUnit}$  associates the original text unit  $\check{u}_1$ , which represents the sentence "The contract was signed on Dec. 30, 2006.", with the concept `conclusion-OfContract` and a set of named entities that merely includes the identified named entity  $e_1 = (\text{date, "2006-12-30"})$ . To proceed, we extend our notions of text unit layers, text documents, and text archives by defining their marked-up counterparts:

**Definition 9 (Semantically Marked-Up Text Unit Layer)** *Given the text unit layer  $\check{r}$ : `TextUnitLayer`, a semantically marked-up text unit layer  $\hat{r} := \langle \hat{u}_1, \hat{u}_2, \dots, \hat{u}_{|\hat{r}|} \rangle$  is a sequence of semantically marked-up text units such that  $\hat{u}_i: \text{SmuTextUnit}$  and  $\hat{u}_i.\text{textUnit}() = \check{r}.\text{textUnit}(i)$ , where  $i = 1, 2, \dots, \check{r}.\text{size}()$ .*

**Definition 10 (Semantically Marked-Up Text Document)** *Given the text document  $\check{t}$ : `TextDocument`, a semantically marked-up text document  $\hat{t} := (\check{t}: \text{TextDocument}, \hat{r}: \text{SmuTextUnitLayer})$  is a 2-tuple comprising the original text document  $\check{t}$  and the semantically marked-up text unit layer  $\hat{r}$  that represents a decomposition of text document  $\check{t}$  into semantically marked-up text units.*

**Definition 11 (Semantically Marked-Up Text Archive)** *Given the text archive  $\hat{a}$ : TextArchive, a semantically marked-up text archive  $\hat{a} := \langle \hat{t}_1, \hat{t}_2, \dots, \hat{t}_{|\hat{a}|} \rangle$  is a sequence of not necessarily distinct, semantically marked-up text documents such that  $\hat{t}_i$ : SmuTextDocument and  $\hat{t}_i.\text{textDocument}() = \hat{a}.\text{textDocument}(i)$ , where  $i = 1, 2, \dots, \hat{a}.\text{size}()$ .*

Consequently, the abstract data types SmuTextUnitLayer (see Appendix B.15 on page 226), SmuTextDocument (see Appendix B.16 on page 226), and SmuTextArchive (see Appendix B.17 on page 227) encapsulate the respective framework-specific objects.

Besides semantically tagging text archives, we aim at inferring a concept-based XML document type definition. Again disregarding syntax issues of creating XML DTDs, we introduce the notion of conceptual document structures. A conceptual document structure encapsulates the necessary information for generating a concept-based XML DTD. This DTD reflects the thematic structure of marked-up documents on the text unit level by enumerating existing concepts and associated named entity types.

**Definition 12 (Conceptual Document Structure)** *Given the semantically marked-up text archive  $\hat{a}$ : SmuTextArchive, the set  $\hat{A} = \{\hat{u} \mid \hat{u}: \text{SmuTextUnit} := \hat{a}.\text{smuTextUnit}(j, k_j) \forall j = 1, 2, \dots, \hat{a}.\text{size()} \forall k_j = 1, 2, \dots, \hat{a}.\text{smuTextUnitLayerSize}(j)\}$  contains all distinct, semantically marked-up text units in  $\hat{a}$ . The archive-specific set  $O_{\hat{a}} := \{o \mid o: \text{Concept} := \hat{u}.\text{concept()} \forall \hat{u}: \text{SmuTextUnit} \in \hat{A}\}$  contains all concepts occurring in  $\hat{a}$ . A conceptual document structure  $\hat{s} := (\hat{l}, o, p)$  is a 3-tuple comprising an alphanumeric label  $\hat{l}$ : String that concisely describes the common theme of text documents covered by  $\hat{s}$ , the sequence of concepts  $o := \langle o_1, o_2, \dots, o_{|O_{\hat{a}}|} \rangle$ , where  $o_i \in O_{\hat{a}}$  for  $i = 1, 2, \dots, |O_{\hat{a}}|$ , the sequence of possibly empty sets  $p := \langle P_1, P_2, \dots, P_{|O_{\hat{a}}|} \rangle$ , where  $P_i := \{p \mid p: \text{NamedEntityType} := \hat{u}.\text{distinctNamedEntityTypes}()[l] \forall \hat{u}: \text{SmuTextUnit} \in \hat{A} \text{ s.t. } \hat{u}.\text{concept}() = o_i \forall l = 1, 2, \dots, \hat{u}.\text{distinctNamedEntityTypes}().\text{size}\}$  represents the types of named entities extracted from text units associated with concept  $o_i$ . The semantically marked-up text archive  $\hat{a}$  conforms to conceptual document structure  $\hat{s}$ .*

The ADT ConceptualDocumentStructure (see Appendix B.18 on page 227) encapsulates one conceptual document structure. Although we abstract from syntactical issues related to generating XML document type definitions from conceptual document structures as well as transforming semantically marked-up text documents into XML documents, this ADT nevertheless defines the corresponding operations xmlDtd() and xmlDocuments( $\hat{a}$ : SmuTextArchive) for the purposes of making explicit the objectives of our framework and providing an overview thereof in the next section.

The DIASDEM framework adopts a knowledge discovery approach to semantic text annotation. In Section 2.1.1, we have emphasized that knowledge discovery is an inherently process-oriented activity. We have described the generic process of discovering knowledge in textual databases. Since our framework involves a specific KDT process comprising multiple algorithms, we finally introduce our notion of KDT process flows:

**Definition 13 (KDT Process Flow)** Let  $g(\cdot)$  denote an arbitrary, parameterizable KDT algorithm for selecting, pre-processing, or transforming textual data, or for discovering, interpreting, or evaluating patterns. A specific, parameterizable KDT process flow  $f := \langle g_1(\cdot), g_2(\cdot), \dots, g_{|f|}(\cdot) \rangle$  is a sequence of KDT algorithms that must be parameterized and executed in the order of their occurrence in  $f$  to accomplish the objectives of the respective KDT process. Let  $\tilde{g}(\cdot)$  denote an arbitrary, parameterized KDT algorithm. A parameterized KDT process flow is denoted by  $\tilde{f} := \langle \tilde{g}_1(\cdot), \tilde{g}_2(\cdot), \dots, \tilde{g}_{|\tilde{f}|}(\cdot) \rangle$ .

In our knowledge discovery context, the KDT process flow  $f$  is modeled as a sequence of algorithms because algorithm  $g_{i+1}(\cdot)$ , where  $i = 1, 2, \dots, |f| - 1$ , typically requires the output generated by its predecessor  $g_i(\cdot)$  as input. Considering possibilities for a potential parallel execution of several KDT algorithms is beyond the scope of this work.

The abstract data types `KdtAlgorithm` (see Appendix B.19 on page 228) and `KdtProcessFlow` (see Appendix B.20 on page 229) encapsulate one KDT algorithm and one KDT process flow, respectively. These abstract data types represent both parameterizable and fully parameterized algorithms and process flows, respectively. In the former case, KDT algorithms do not yet contain parameter values.

## 3.2 Objectives and Overview

Providing a compelling solution to the research questions stated in Section 1.4 constitutes the main objective of the DIAsDEM framework. The conceptual framework, along with its prototype implementation, addresses the following primary research question:

*Can techniques for knowledge discovery in textual databases be employed to convert large archives comprising domain-specific text documents of homogeneous content into semantically marked-up XML documents?*

In Section 1.2, we have informally introduced the framework-specific notion of semantically marked-up XML documents. As part of the literature review, the research discipline of knowledge discovery in textual databases has been concisely introduced in Subsection 2.1.1. The key terms text document, text archive, and text archive, along with their semantically marked-up counterparts, have been defined in the preceding section. We intentionally restrict the applicability of our framework to text archives that exhibit the following characteristics:

- All input text documents are composed of characters from an arbitrary, finite alphabet whose characters are interpreted from left to right.
- Furthermore, the text archive contains domain-specific, as opposed to general, documents of relatively homogeneous content. The text documents to be marked up do not comprise narrative or literary text, but rather consist of expository text that explicitly explains, teaches, or states something (cf. Hearst, 1997, p. 35).

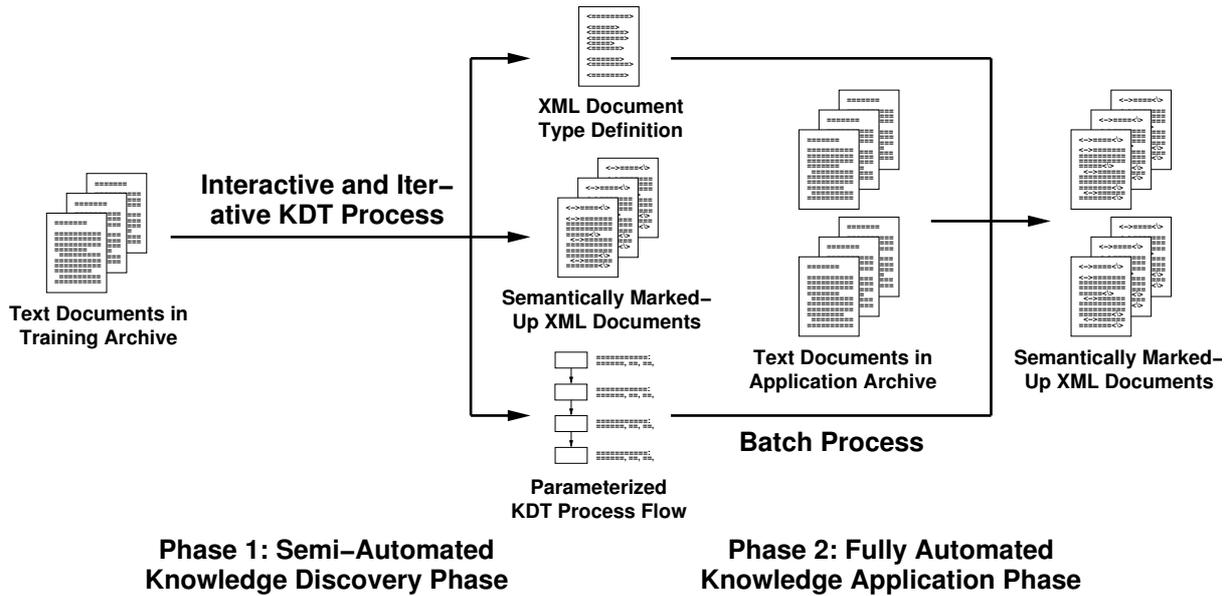
- Moreover, a natural clustering tendency (cf. Jain and Dubes, 1988, pp. 201–202) is observed at the selected text unit level of the archive. Trying to identify frequently occurring concepts at the text unit level only makes sense if text units exhibit a predisposition to cluster into natural groups of semantically similar content.
- Finally, the input text archive is both large and potentially valuable enough to outweigh the costs of employing human domain and KDT experts, which incur during the initial, inherently semi-automated knowledge discovery phase.

To sum up, our framework is focused on semantically annotating large, domain-specific text archives whose documents comprise rather homogeneous content and exhibit a natural clustering tendency at the text unit level. These prerequisites are satisfied by various kinds of important text archives, such as public announcements of courts and administrative authorities, quarterly and annual reports to shareholders, collections of company and industry news, textual patient records in health care applications, as well as product and service descriptions published on electronic marketplaces.

As an overall objective, the DIASDEM framework adopts a knowledge discovery approach (i) to convert the domain-specific text archive  $\check{a}$ : TextArchive into the semantically marked-up text archive  $\hat{a}$ : SmuTextArchive, (ii) to establish the accompanying conceptual document structure  $\hat{s}$ : ConceptualDocumentStructure, (iii) to output the domain-specific and concept-based XML document type definition  $\hat{s}.xmlDtd()$ , and (iv) to create the collection of semantically marked-up XML documents  $\hat{s}.xmlDocuments(\hat{a})$ .

Unlike text classification techniques (cf. Sebastiani, 1999), our framework neither requires pre-defined thematic concepts nor relies on the existence of manually tagged training XML documents. Instead, we employ unsupervised learning, or clustering techniques, to discover classification knowledge in unstructured text documents. Specifically, we initially acquire knowledge about frequently recurring thematic concepts at the text unit level. Subsequently, this knowledge is exploited to classify text units by assigning them to content-descriptively labeled concepts. To that end, the entire text archive is split into the training archive  $\check{a}_T$ : TextArchive and the typically larger, or even continuously growing, application archive  $\check{a}_A$ : TextArchive. The overall objective of our framework is thus attained by pursuing the following two, more specific, and complementary goals:

1. Knowledge discovery: Given the domain-specific training text archive  $\check{a}_T$ , domain and KDT experts shall be supported in interactively parameterizing the generic, framework-specific KDT process flow for semantic XML tagging  $f$ : KdtProcessFlow. Thereby, training text archive  $\check{a}_T$  is interactively transformed into semantically marked-up text archive  $\hat{a}_T$ : SmuTextArchive, and the accompanying conceptual document structure  $\hat{s}$ : ConceptualDocumentStructure is established. To enable a quality assessment of semantic markup by domain experts, the XML document type definition  $\hat{s}.xmlDtd()$  and the semantically marked-up XML documents  $\hat{s}.xmlDocuments(\hat{a}_T)$  shall be created.



**Figure 3.1:** Outline of the Two-Phase DIAsDEM Framework

2. Knowledge application: Given the application text archive  $\check{a}_A$ , which comprises thematically similar text documents as the training text archive  $\check{a}_T$ , and the fully parameterized KDT process flow  $\hat{f}$ :  $KdtProcessFlow$  for semantic XML tagging, application text archive  $\check{a}_A$  shall be automatically transformed into the semantically marked-up text archive  $\hat{a}_A$ :  $SmuTextArchive$ , which conforms to the conceptual document structure  $\hat{s}$ . In addition, the collection of XML documents  $\hat{s}.xmlDocuments(\hat{a}_A)$  shall be output. Domain experts shall be supported in monitoring the quality of semantic XML markup.

Figure 3.1 illustrates the two-phase DIAsDEM framework for semantic XML tagging of large, domain-specific text archives. For each new domain, the semi-automated knowledge discovery phase must be completed only once. This explorative phase, which is outlined in the next section, comprises several subtasks to attain the knowledge discovery objectives stated above. Besides transforming the training text archive into a semantically marked-up text archive, the interactive and iterative knowledge discovery process outputs an archive-specific XML DTD and a fully parameterized KDT process flow. This process flow encapsulates a sequence of algorithms and their respective parameter settings, which must be executed to semantically mark up new text archives in the knowledge application phase. As outlined in Section 3.4, huge amounts of new texts are automatically converted into semantically annotated XML documents to achieve the knowledge application objectives in this productive, repeatable phase.

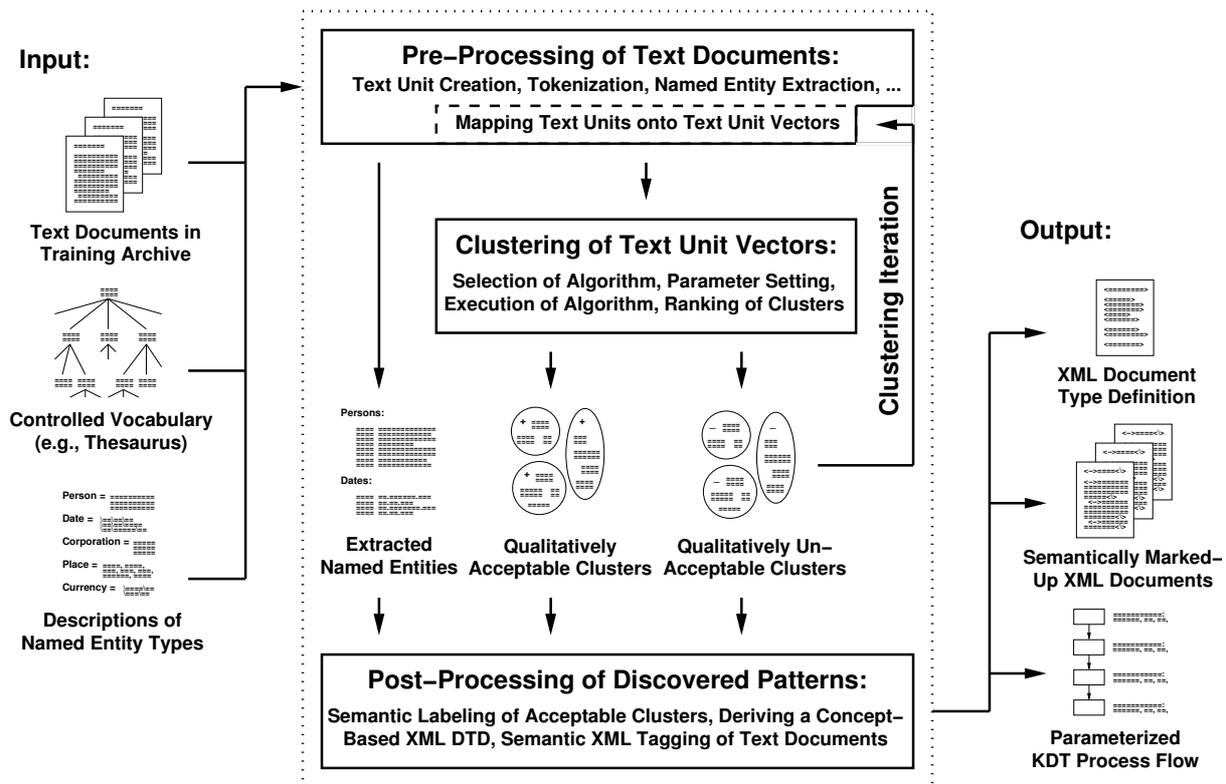


Figure 3.2: Knowledge Discovery Process of the DIASDEM Framework

### 3.3 Knowledge Discovery Phase

In this section, the knowledge discovery phase of the DIASDEM framework is informally outlined whereas it is described in full detail in Chapter 4. Illustrated in Figure 3.2, the interactive and iterative KDT process accomplishes the knowledge discovery goals stated in the preceding section. Finding groups of thematically similar text units featuring coherent topics is the major challenge. To meet this challenge, we employ an unsupervised learning technique (i.e., clustering) in the pattern discovery step to identify semantically homogeneous subgroups of text units.

Besides the training archive of text documents, a controlled vocabulary representing domain-specific terminology and descriptions of relevant named entity types constitute the main input to the knowledge discovery process. After purposefully choosing the level of text unit granularity, basic text document pre-processing (e.g., text unit creation, tokenization, lemmatization, and optional word sense disambiguation) is performed in the KDT pre-processing step. Additionally, named entities are extracted from text units. Instead of removing meaningless stopwords to reduce the dimensionality, we establish a drastically reduced feature space by selecting a limited set of terms (i.e., text unit descriptors) from the controlled vocabulary. Text unit descriptors are chosen by the domain

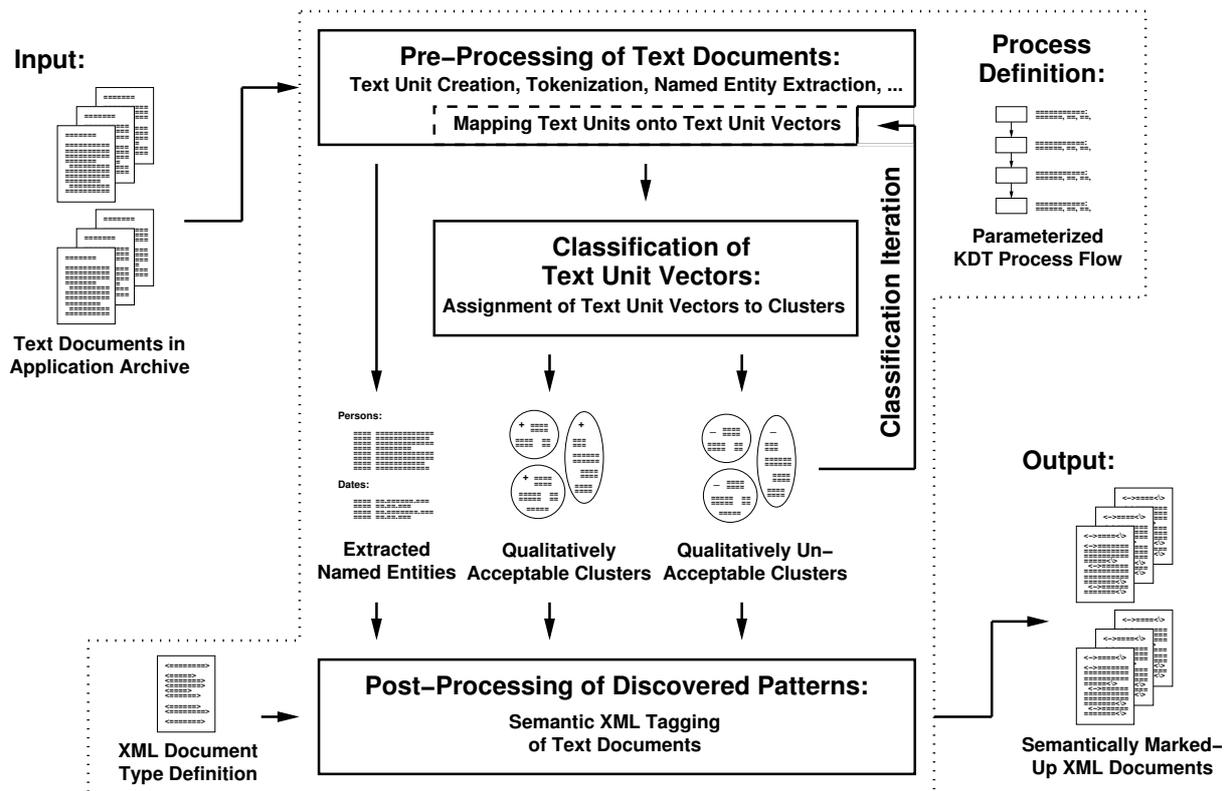
expert because they have to reflect important terminology of the domain. Subsequently, all text units are mapped onto numerical text unit vectors by utilizing the vector-space model introduced by Salton (1968, pp. 236–243; cf. Subsection 2.1.2).

In the pattern discovery step, text unit vectors, each representing one text unit, are clustered based on similarity. The objective is to discover dense and semantically homogeneous clusters of text unit vectors and text units, respectively. Our KDT process is termed iterative because a clustering algorithm is repeatedly invoked with a varying input data set and possibly different parameter settings to identify as many groups of thematically similar text units as possible. Our notion of iterative clustering, by repeatedly executing a clustering algorithm, is different from the multiple passes over the input data that are internally performed by some clustering algorithms.

In Subsection 4.3.4, we suggest algorithms that are particularly appropriate for clustering text unit vectors. Each clustering iteration outputs a set of clusters, which is automatically partitioned into acceptable and unacceptable ones according to framework-specific cluster quality criteria. A cluster of text unit vectors is qualitatively acceptable if (i) its cardinality is sufficiently large, (ii) the corresponding text units are homogeneous, and (iii) the text units can be content-descriptively characterized by a small number of text unit descriptors. Text units that correspond to members of acceptable clusters approved by the domain expert are removed from the data set for subsequent labeling. In contrast, text units that correspond to the remaining text unit vectors assigned to unacceptable clusters are again mapped onto numerical vectors. These text unit vectors constitute the input data to the clustering algorithm in the next iteration. In each iteration, parameters of the clustering algorithm may be adjusted such that acceptable clusters become progressively less specific in content. The clustering iteration stops if acceptable clusters cannot be found or if the expert prefers to stop pattern discovery.

In the post-processing step, qualitatively acceptable clusters are semi-automatically assigned a content-descriptive label. Although cluster labels are ultimately determined by the domain expert, cluster label suggestions are automatically formulated to assist the expert in choosing meaningful semantic cluster labels. All default cluster labels are derived from feature space dimensions (i.e., text unit descriptors) that are prevailing in the respective cluster. The finally chosen cluster labels correspond to the names of semantic XML tags, which are subsequently used to annotate the respective cluster members. Furthermore, XML tags are enhanced by attributes that represent previously extracted named entities. Moreover, an archive-specific, concept-based XML document definition is automatically derived and all original text documents are transformed into semantically marked-up XML documents that conform to this XML DTD. Finally, a fully parameterized KDT process flow is created that enables an automated transformation of text documents from the same domain into semantically annotated XML documents.

To evaluate the markup quality in absence of pre-tagged test documents, a random sample of marked-up text units is drawn and manually evaluated by domain specialists. Analogous to supervised learning, the quality of XML tag names is in principle assessed with respect to two error types. Firstly, a text unit may be mistakenly annotated with a



**Figure 3.3:** Knowledge Application Process of the DIASDEM Framework

semantic XML tag that does not properly reflect the content of the marked-up text unit. Secondly, a text unit may not be annotated at all although the concept-based XML DTD comprises a semantic XML tag that appropriately reflects the content of the untagged text unit. Additionally, the quality of XML tag attributes and extracted named entities, respectively, is assessed in accordance with the concept-based XML DTD.

The entire KDT process is intentionally designed to be interactive, or semi-automated, to create semantic markup of high quality. Besides evaluating the tagging quality, domain and KDT experts supervise the whole knowledge discovery process. In particular, human experts establish a new, or refine an existing, domain-specific controlled vocabulary (cf. Moens, 2000, pp. 51–53) in the pre-processing step. They also approve, modify, or reject automatically generated cluster label suggestions in the post-processing step.

### 3.4 Knowledge Application Phase

Having successfully completed the initial knowledge discovery process for a domain-specific text archive, the second phase of the DIASDEM framework enables an automated annotation of text documents from the same domain. This batch-oriented and productive

process is illustrated in Figure 3.3 and informally summarized in this section. Details regarding the second phase of our framework are presented in Subsection 4.3.4, in Section 4.5, and in the descriptions of two real-world case studies in Chapter 6.

A fully parameterized KDT process flow can be executed without human intervention and thus plays a central role in applying discovered classification knowledge to documents that are thematically similar to the training documents. Informally, the KDT process flow is a sequence of parameterized algorithms that are executed one after the other to annotate domain-specific text documents. In the knowledge application phase, exactly the same pre-processing steps are performed as in the knowledge discovery phase. Instead of iteratively clustering text unit vectors, however, vectors are iteratively classified by assigning them to clusters and thematic concepts, respectively, identified in phase one.

In the knowledge application phase, the number of classification iterations equals the number of clustering iterations performed in phase one. Each clustering iteration of the KDT process is associated with a set of qualitatively acceptable, hence semantically labeled clusters and a set of qualitatively unacceptable clusters. These iteration-specific clusters of text unit vectors constitute the discovered classification knowledge. In the first classification iteration, all text unit vectors are assigned to one of the clusters discovered in the first clustering iteration. The concrete assignment procedure depends on the characteristics of the clustering algorithm employed beforehand. If clusters are represented by centroid vectors, for example, an input text unit vector is assigned to the cluster whose centroid vector is closest to itself, according to the utilized proximity measure. At the end of each iteration, text units that correspond to vectors assigned to qualitatively acceptable clusters are removed from the data set since their content-descriptive labels correspond to the respective cluster labels. By contrast, text units whose vectors are assigned to unacceptable clusters are again mapped onto numerical text unit vectors. They constitute the input data for the next classification iteration.

The post-processing step is reduced to semantically marking up original text documents. All created XML documents adhere to the concept-based XML document type definition established in phase one. Unlike the interactive knowledge discovery phase, no human interventions are thus necessary to transform text documents into semantically marked-up XML documents. Continuously monitoring the quality of automatically created markup, however, still requires domain experts.

## 3.5 Summary

In this chapter, we have defined fundamental terminology for the scope of this work. Thereafter, our two-phase DIASDEM framework for semantic XML tagging of large, domain-specific text archives has been introduced. For each new domain, a semi-automated knowledge discovery phase must be completed once. We have outlined an interactive KDT process that iteratively discovers classification knowledge required for semantic markup, derives a conceptual XML DTD, and creates a parameterized KDT process flow.

By executing this process flow in the second knowledge application phase, thematically similar text documents are automatically transformed into semantically marked-up XML documents.



## 4 DIAsDEM Knowledge Discovery Process

As an integral component of our framework for semantic XML tagging, the DIAsDEM knowledge discovery process has been coarsely outlined in Section 3.3. In this chapter, we introduce the interactive and iterative knowledge discovery process in full detail. To that end, Section 4.1 extends the framework-specific terminology by defining terms relevant for all subsequent sections whereas more specific terminology is defined throughout this chapter. As illustrated in Figure 3.2 on page 65, the KDT process consists of three main steps, which are henceforth covered by three individual sections. Firstly, Section 4.2 describes the necessary pre-processing of text documents. Secondly, the discovery of patterns by means of clustering text unit vectors is discussed in Section 4.3. Thirdly, Section 4.4 explains our approach to post-processing discovered patterns to accomplish the knowledge discovery objectives of the DIAsDEM framework. The parameterized KDT process flow specified in Section 4.5 bridges the semi-automatic discovery and the automated application of knowledge. Furthermore, the current extent and possible reductions of human involvement in the KDT process are discussed in Section 4.6. Finally, this chapter is summarized in Section 4.7.

To illustrate concepts and algorithms introduced in this chapter, we utilize the exemplary texts listed in Table 4.1. These five news items used for illustrative purposes are contained in Volume 1 of the Reuters Corpus (cf. Rose et al., 2002). Each news item comprises information concerning the definitive or likely agreement between a divesting and an acquiring firm about the sale of a business unit. Divestitures, or so-called sell-offs, are one form of corporate restructuring (cf. Gaughan, 2002, pp. 395–417). The item identifier listed in the second column of Table 4.1 is the primary key of news items in the corpus. Henceforth, these news items are referred to by means of their identifiers  $\check{t}_{E1}$ : TextDocument through  $\check{t}_{E5}$ : TextDocument, which are listed in column three.

### 4.1 Terminology

Although fundamental, framework-specific terms have already been defined in Section 3.1, additional terminology is required in the context of our knowledge discovery process. In the preceding chapter, the number of introduced terms has been intentionally kept to a minimum to enhance conceptual clarity. We have focused on the input side (i.e., an archive of text documents and their components) and on the output side (i.e., a conceptual

**Table 4.1:** Five Reuters News Items Used in Examples (Source: Reuters Corpus, Volume 1, English Language, 1996-08-20 to 1997-08-19; cf. Rose et al., 2002)

Reuters News Item	Item	Text
USA: VASCO says to sell consulting unit. VASCO Corp said it agreed to sell its consulting and technical organization, VASCO Performance System, to Wizdom Systems Inc. Terms were not disclosed. –Chicago Newsdesk 312-408-8787	16106	$\check{t}_{E1}$
USA: Helmerich to sell unit to Occidental. Helmerich & Payne Inc said Tuesday it entered into a definitive agreement under which Occidental Petroleum Corp will acquire its Baytown, Texas-based Natural Gas Odorizing Inc subsidiary. The total acquisition value will be about \$48 million, payable in Occidental common stock. Closing is expected on August 30. – Chicago newsdesk 312 408-8787	17109	$\check{t}_{E2}$
USA: Weyerhaeuser may sell subsidiary. Timber giant Weyerhaeuser Company said Monday it may sell its mortgage loan subsidiary, the Weyerhaeuser Mortgage Company. The company said it has retained the New York investment banking firm of Goldman, Sachs & Co. to explore its strategic options with the 1,500-employee unit. The mortgage unit, which is headquartered in Woodland Hills, Calif., originated \$2.1 billion in residential first mortgages in the eight months that ended Aug. 31. The company did not comment on the timing of its announcement or on the sale price of the subsidiary. In early trading on the New York Stock Exchange, the stock was up 25 cents to \$46.87.	71177	$\check{t}_{E3}$
USA: Miller to sell unit to Modtech. Miller Building Systems Inc said it agreed to sell its Miller Structures Inc unit in California to Modtech Inc, a maker of modular classrooms. Final settlement is scheduled for October 21, Miller said. Terms were not disclosed. –Chicago Newsdesk 312-408-8787	78899	$\check{t}_{E4}$
USA: Zurn sells unit to Constellation Capital. Zurn Industries Inc said on Tuesday that it will sell its Zurn Mechanical Power Transmission Group to Constellation Capital Partners LLC, for an undisclosed amount. The companies indicated that the sale is expected to be completed by the end of November. In a statement, Philip Knisely, president of Constellation Capital, said: “Mechanical Power Transmission (MPT) represents an excellent platform for growth both internally and through complementary additional acquisitions.” – New York Newsdesk +1 212 859 1610	115995	$\check{t}_{E5}$

document structure along with an archive of semantically marked-up text documents and their components) of the framework. The knowledge discovery algorithms described in this chapter, however, require additional intermediate data structures to ultimately transform input text documents into semantically marked-up XML documents.

Analogous to Vintar et al. (2002), we create tokenized text in the pre-processing step. To that end, our notion of tokens and tokenized text units is made explicit as follows:

**Definition 14 (Token)** *A token  $\check{w}$ : String is a sequence of characters that represents one natural language word, one punctuation mark, one numeral, or any other atomic and semantic-carrying character sequence. A set of distinct tokens is denoted by  $\check{W} := \{\check{w}_1, \check{w}_2, \dots, \check{w}_{|\check{W}|}\}$ .*

The abstract data type `Token` (see Appendix B.21 on page 229) encapsulates one token. For example, the tokens  $\check{w}_1: \text{Token} := \check{w}_1.\text{create}(\text{"New York"})$ ,  $\check{w}_2: \text{Token} := \check{w}_2.\text{create}(\text{"."})$ ,  $\check{w}_3: \text{Token} := \check{w}_3.\text{create}(\text{"The"})$ , and  $\check{w}_9: \text{Token} := \check{w}_9.\text{create}(\text{"30"})$  are contained in text document  $\check{t}_1: \text{TextDocument}$ , which encapsulates the string `"New York: The contract was signed on Dec. 30, 2006."` Although whitespace characters often separate tokens from each other, whitespace characters are preserved if they are integral parts of natural language words, like the token `"New York"`.

**Definition 15 (Tokenized Text Unit)** *Given the text unit  $\check{u}: \text{TextUnit}$ , a tokenized text unit  $\check{u} := \langle \check{w}_1, \check{w}_2, \dots, \check{w}_{|\check{u}|} \rangle$  is a sequence of tokens identified in text unit  $\check{u}$  such that  $\check{w}_i: \text{Token}$ , where  $i = 1, 2, \dots, |\check{u}|$ . The set  $W_{\check{u}} := \{ \check{w} \mid \check{w}: \text{Token} := \check{u}[i], i = 1, 2, \dots, |\check{u}| \}$  contains all distinct tokens identified in tokenized text unit  $\check{u}$ .*

The concrete number of tokens identified in a text unit depends on domain-specific requirements and the specific tokenization algorithm. Issues in tokenization are discussed in Subsection 4.2.1. By tokenizing a text unit, we intentionally ignore whitespace characters that separate single tokens in the original text unit. One tokenized text unit is encapsulated by the ADT `TokenizedTextUnit` (see Appendix B.22 on page 229). For instance, let the tokenized text unit  $\check{u}_1: \text{TokenizedTextUnit}$  comprise the sequence of strings `<"The", "contract", "was", "signed", "on", "Dec.", "30", "", "2006", ".">` that corresponds to text unit  $\check{u}_1$ . The operations provided by this abstract data type can be illustrated as follows:  $\check{u}_1.\text{size}() = 10$ ,  $\check{u}_1.\text{token}(2)$  represents the token `"contract"`, and  $\check{u}_1.\text{tokens}(6, 9)$  corresponds to the token sequence `<"Dec.", "30", "", "2006">`.

In Definition 7 on page 59, we have introduced our notion of named entities. To actually extract them from tokenized text units, intermediate named entities are required:

**Definition 16 (Intermediate Named Entity)** *Let  $P := \{p_1, p_2, \dots, p_{|P|}\}$  denote a set of domain-specific named entity types that represent abstract classes and generic numerical expressions. Given the tokenized text unit  $\check{u}: \text{TokenizedTextUnit}$ , an intermediate named entity  $\bar{e} := (p: P, \check{e}: \text{String}, i: \text{Integer}, j: \text{Integer}, k: \text{Integer})$  is a 5-tuple comprising named entity type  $p$ , the canonical form  $\check{e}$  of instantiated named entity type  $p$ , its unique identifier  $i$ , as well as the start token index  $j$  and the end token index  $k$ , where  $j = 1, 2, \dots, \check{u}.\text{size}()$  and  $k = j, j + 1, \dots, \check{u}.\text{size}()$ , such that  $\check{u}.\text{tokens}(j, k)$  corresponds to the tokens that instantiate named entity type  $p$  in tokenized text unit  $\check{u}$ . A set of distinct intermediate named entities is denoted by  $\bar{E} := \{\bar{e}_1, \bar{e}_2, \dots, \bar{e}_{|\bar{E}|}\}$ .*

Given the exemplary set of named entity types  $P_1 := \{\text{company}, \text{person}, \text{date}\}$ , the intermediate named entity  $\bar{e}_1 := (\text{date}, \text{"2006-12-30"}, 2, 6, 9)$  may, for example, be extracted from the tokenized text unit  $\check{u}_1$  introduced above. This intermediate named entity is instantiated by tokens 6 through 9 that constitute the token sequence `<"Dec.", "30", "", "2006">`. The abstract data types `IntNamedEntity` (see Appendix B.23 on page 230) and `SetOfIntNamedEntities` (see Appendix B.24 on page 231) encapsulate one intermediate named entity type and one set thereof, respectively.

As we adopt the IR vector-space model outlined in Subsection 2.1.2, pre-processed text units are mapped onto numerical text unit vectors to enable pattern discovery.

**Definition 17 (Text Unit Vector, Preliminary Definition)** *A text unit vector  $\mathbf{u} \in \mathbb{R}^n$ , where  $n \in \mathbb{N}$ , is an  $n$ -dimensional vector.*

Characteristic properties of text unit vectors, such as the semantics of vector dimensions and components, are defined in Subsection 4.2.5. One text unit vector is encapsulated by the abstract data type TextUnitVector (see Appendix B.25 on page 231).

During the process of transforming text units into semantically marked-up text units, each text unit is represented by a corresponding intermediate text unit that comprises the original text unit along with both intermediate (e.g., the text unit vector of the current clustering iteration) and final (e.g., the identified concept) processing results.

**Definition 18 (Intermediate Text Unit)** *An intermediate text unit  $\bar{u} := (\check{u}: \text{TextUnit}, \ddot{u}: \text{TokenizedTextUnit}, \vec{u}: \text{TextUnitVector}, i: \text{Integer}, j: \text{Integer}, o: \text{Concept}, \bar{E}: \text{SetOfIntNamedEntities})$  is a 7-tuple comprising the original text unit  $\check{u}$ , the processed tokenized text unit  $\ddot{u}$ , the text unit vector  $\vec{u}$ , the clustering iteration identifier  $i$ , the cluster identifier  $j$ , the domain-specific concept  $o$  that domain experts typically associate with text unit  $\check{u}$ , and the set of intermediate named entities  $\bar{E}$  identified in text unit  $\check{u}$ .*

For instance, let the tokenized text unit  $\ddot{u}_1: \text{TokenizedTextUnit}$  encapsulate the token sequence  $\langle \text{"contract"}, \text{"sign"} \rangle$  obtained by pre-processing the original text unit  $\check{u}_1: \text{TextUnit}$ , which encapsulates the sentence "The contract was signed on Dec. 30, 2004." Additionally, assume that  $\vec{u}_1: \text{TextUnitVector}$  represents the pre-processed text unit  $\check{u}_1$  in the vector-space model by means of the vector  $[0, 0, 0, 0.12, 0, 0, 0, 0.78, 0, 0]^T$ . Furthermore, the semantic concept  $\text{conclusionOfContract} \in O_1$  is encapsulated by  $o_1: \text{Concept}$  and  $\bar{E}_1: \text{SetOfIntNamedEntities}$  includes intermediate named entities extracted from text unit  $\check{u}_1$ . In this illustrative case,  $\bar{u}_1 := (\check{u}_1, \ddot{u}_1, \vec{u}_1, 1, 5, o_1, \bar{E}_1)$  is an intermediate text unit whose text unit vector  $\vec{u}_1$  is assigned to cluster 5 in the first iteration. The notion of iterative clustering is discussed in detail in Subsection 4.3.4. One intermediate text unit is encapsulated by the ADT IntTextUnit (see Appendix B.26 on page 232).

Finally, we extend our notions of text unit layers, text documents, and text archives by defining their intermediate counterparts. The abstract data types IntTextUnitLayer (see Appendix B.27 on page 233), IntTextDocument (see Appendix B.28 on page 234), and IntTextArchive (see Appendix B.29 on page 234) encapsulate the respective objects.

**Definition 19 (Intermediate Text Unit Layer)** *Given text unit layer  $\check{r}: \text{TextUnitLayer}$ , an intermediate text unit layer  $\bar{r} := \langle \bar{u}_1, \bar{u}_2, \dots, \bar{u}_{|\bar{r}|} \rangle$  is a sequence of intermediate text units such that  $\bar{u}_i: \text{IntTextUnit}$  and  $\bar{u}_i.\text{textUnit}() = \check{r}.\text{textUnit}(i)$ ,  $i = 1, 2, \dots, \check{r}.\text{size}()$ .*

**Definition 20 (Intermediate Text Document)** *Given the text document  $\check{t}: \text{TextDocument}$ , an intermediate text document  $\bar{t} := (\check{t}: \text{TextDocument}, \bar{r}: \text{IntTextUnitLayer})$  is a 2-tuple comprising the text document  $\check{t}$  and the intermediate text unit layer  $\bar{r}$  that represents a decomposition of text document  $\check{t}$  into intermediate text units.*

**Definition 21 (Intermediate Text Archive)** *Given the text archive  $\check{a}: \text{TextArchive}$ , an intermediate text archive  $\bar{a} := \langle \bar{t}_1, \bar{t}_2, \dots, \bar{t}_{|\bar{a}|} \rangle$  is a sequence of not necessarily distinct intermediate text documents such that  $\bar{t}_i: \text{IntTextDocument}$  and  $\bar{t}_i.\text{textDocument}() = \check{a}.\text{textDocument}(i)$ , where  $i = 1, 2, \dots, \check{a}.\text{size}()$ .*

## 4.2 Pre-Processing of Text Documents

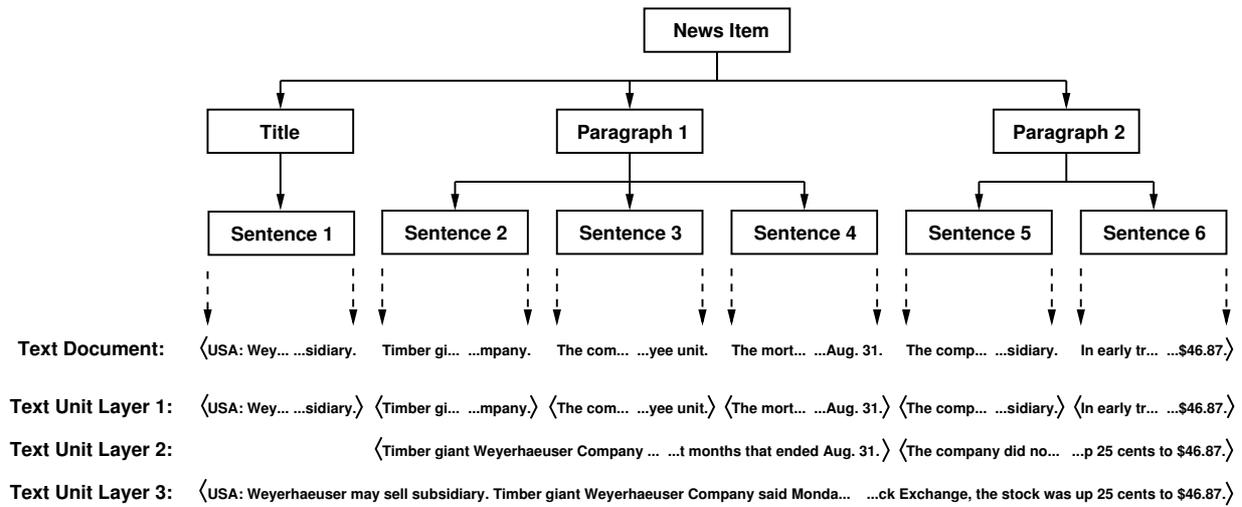
Transforming text encoded as a character sequence into a numerical representation suitable for pattern discovery is the main objective of the pre-processing and transformation steps of the generic KDT process introduced in Subsection 2.1.1. Because each specific knowledge discovery process requires an individual and goal-oriented approach to pre-processing input textual data, the concrete pre-processing steps of the DIASDEM knowledge discovery process are described in this section. The pre-processing tasks that are introduced in the following subsections (i) create and tokenize text, (ii) extract named entities, (iii) lemmatize words and disambiguate their senses if necessary, (iv) establish a controlled vocabulary, and finally (v) map pre-processed text units onto numerical text unit vectors.

### 4.2.1 Creating and Tokenizing Text Units

As explained in Section 3.1, we focus on raw text stored in plain text files. Due to the plethora of existing file conversion programs (e.g., the Linux tool `ps2ascii`), this restriction has only minor practical consequences. Initially, raw text documents are decomposed into tokenized text units of the required granularity level.

**Text Unit Creation** Since our framework aims at semantically marking up structural text units, reliably identifying them in input text documents is of paramount importance. According to Definition 4 on page 58, text units are strings extracted from text documents. They represent structural text components comprising more than one word. Input documents are decomposed into text unit layers (cf. Definition 5 on page 59) in the first pre-processing step such that each document is associated with exactly one text unit layer. Each text unit layer is a sequence of ordered, not necessarily contiguous, structural text units that originate from one document. Decomposed text documents are neither deleted nor replaced by their respective text unit layers because original documents are an integral part of semantically marked-up text documents.

Our concept of text unit layers is illustrated in Figure 4.1, which depicts three conceivable decompositions of example text  $\check{t}_{E3}$  into text units. Adopting the notion of text as an ordered hierarchy of content objects (cf. DeRose et al., 1990, pp. 3–6), we assume that news item  $\check{t}_{E3}$  consists of six sentences, each of which is either assigned to the title (i.e., the first sentence “USA: Weyerhaeuser may sell subsidiary.”) or to one of two paragraphs. Figure 4.1 illustrates the hierarchical structure of document  $\check{t}_{E3}$  and indicates sentence



**Figure 4.1:** Text Document  $\check{t}_{E3}$  Decomposed into Three Distinct Text Unit Layers

**Table 4.2:** Text Document  $\check{t}_{E1}$  Decomposed into Text Unit Layer  $\check{r}_{E1}$ :  $\text{TextUnitLayer} := \check{r}_{E1}.\text{create}(\check{t}_{E1})$

---

$\check{r}_{E1} = \langle \langle \text{"USA: VASCO says to sell consulting unit."}, \langle \text{"VASCO Corp said it agreed to sell its consulting and technical organization, VASCO Performance System, to Wizdom Systems Inc."}, \langle \text{"Terms were not disclosed."}, \langle \text{"-Chicago Newsdesk 312-408-8787"} \rangle \rangle \rangle$

---

boundaries by dashed arrows. Text unit layer 1 represents all six sentences as individual text units whereas the second text unit layer models paragraphs as text units. Finally, text unit layer 3 represents the entire document as a single text unit.

Semantically marked-up text documents comprise exactly one marked-up text unit layer. Therefore, domain experts have to determine the required level of text unit granularity in the pre-processing phase. When making this important decision, experts have to consider the required natural clustering tendency of text units (cf. Section 3.2). In addition, selecting the granularity of text units simultaneously determines the granularity of semantic markup (cf. Section 1.2). In both case studies described in Chapter 6, for example, we created fine-grained semantic markup at the sentence level because sentences of the respective text documents clearly exhibited a natural clustering tendency. Depending on the concrete application domain, however, structural text units may even correspond to main and subordinate clauses, specific sections, or entire chapters. In general, a finer text unit granularity entails more specific semantic tags and vice versa.

The complexity of identifying text unit boundaries in documents is influenced by the application domain, the characteristics of input text documents (e.g., original file format), and the granularity level chosen by the expert. Ideally, existing metadata about the logical

document structure can be exploited to reliably identify chapters, headings, sections, and paragraphs of input documents. If metadata that make explicit structural document components are not available, a variety of specific algorithms for document structure analysis (cf. Mao et al., 2003b) can be employed to infer the missing structural metadata. Reviewing this specific thread of research, however, is beyond the scope of this work.

Instead of striving towards detecting arbitrary document structures, the prototype DIASDEM WORKBENCH is highly focused on identifying sentences in text documents for two reasons. On the one hand, sentences often convey important, self-contained facts that are worth being semantically marked up. On the other hand, many algorithms for heuristic sentence boundary detection are available and tend to work well in domain-specific texts (Manning and Schütze, 1999, pp. 134–136). Based on supervised machine learning techniques, Mikheev (1998), for instance, developed a domain-independent method to detect sentence boundaries and achieved an accuracy rate of 99.25% in a large-scale evaluation. The heuristics applied by DIASDEM WORKBENCH to identify sentence boundaries are outlined in Subsection 5.2.1.

In principle, any method for identifying text units can be utilized in the pre-processing step of our framework. Hence, the abstract data type `TextUnitLayer` merely defines the constructor `create(̃tinit: TextDocument)` that constructs a new text unit layer by decomposing the input text document  $\check{t}_{\text{init}}$  into  $i \in \mathbb{N}$  ordered, non-overlapping, structural text units  $\langle \check{u}_1, \check{u}_2, \dots, \check{u}_i \rangle$  of the required granularity level (see Appendix B.7 on page 222). For illustrative purposes, Table 4.2 depicts the result of decomposing example text document  $\check{t}_{\text{E1}}$  into the sentence-based text unit layer  $\check{r}_{\text{E1}}: \text{TextUnitLayer} := \check{r}_{\text{E1}}.\text{create}(\check{t}_{\text{E1}})$ .

**Tokenization** Text units (i.e., character sequences) are divided into units called tokens in the second pre-processing step, referred to as tokenization. Each token “is either a word or something else like a number or a punctuation mark” (Manning and Schütze, 1999, p. 124). According to Fox (1992, p. 102), “tokens are groups of characters with collective significance.” In Definition 14 on page 72, we have analogously defined tokens as sequences of contiguous characters that represent one natural language word, one punctuation mark, one numeral, or any other atomic and semantic-carrying character sequence. Baeza-Yates and Ribeiro-Neto (1999, pp. 165–167) preferred the term lexical analysis to denote the process of converting a stream of characters into a stream of words. Taking the information retrieval perspective, lexical analysis and tokenization, respectively, is the first stage in both automatic indexing and query processing. In our KDT process, splitting text units into tokens is a fundamental pre-processing technique.

Intuitively, identifying tokens seems to be as easy as splitting a character sequence at the occurrences of one or many contiguous whitespace characters that usually separate natural language words from each other. Of course, the occurrence of whitespace serves as the main clue for tokenizing English or German texts. Nevertheless, tokenization involves several lexical issues that must be carefully considered. Tokens do not solely consist of letters, and tokens are not always surrounded by whitespace characters. Under the felicitous heading “Tokenization: What is a word?”, Manning and Schütze (1999, pp. 124–

131) discussed several problems that typically arise during tokenization:

- Punctuation marks: The substrings "disclosed. VASCO" and "organization, VASCO" of text  $\check{t}_{E1}$  illustrate that punctuation marks, due to layout conventions, are often directly attached to words. During tokenization, words are separated from attached punctuation marks, like  $\langle$ "disclosed", ".", "VASCO" $\rangle$ . Periods, commas, and semicolons, however, are often parts of tokens and do not serve as punctuation marks. Periods may mark the end of sentences or indicate abbreviations. In the latter case, periods are inseparable parts of tokens (e.g., "Inc." in text  $\check{t}_{E1}$  or "Aug." in text  $\check{t}_{E3}$ ). Furthermore, punctuation marks are often integral parts of tokens that represent numerals (e.g., "1,000.5") or date expressions (e.g., "30.12.2006").
- Hyphenation: Character sequences containing a hyphen, like "Texas-based" in text  $\check{t}_{E2}$ , require careful attention. One difficulty involves detecting and removing line-breaking hyphens that are inserted to typographically improve the justification of text. As the string "1,500-employee" in  $\check{t}_{E3}$  illustrates, hyphens at the end of lines cannot always be removed. Other hyphenated character sequences include words like "so-called", which are typically regarded as one token, and words like "Texas-based", which are mostly split into several tokens. Inconsistent spelling (e.g., "cooperate" vs. "co-operate") within one archive adds complexity as well.
- Whitespace not indicating a word break: Unlike the above mentioned issues of not splitting certain character sequences, the opposite problem of regarding a character sequence comprising whitespace as one token arises as well. For instance, multi-part words like "Stock Exchange" in text  $\check{t}_{E3}$ , well-known proper nouns, such as "New York" in text  $\check{t}_{E5}$ , or other semantic-carrying character sequences, like phone numbers, are typically considered as single tokens. However, named entities (e.g., the company "VASCO Corp" in text  $\check{t}_{E1}$ ) are not identified during tokenization. Tokens that include whitespace are henceforth referred to as multi-part tokens.

In the context of automatic indexing, Fox (1992, p. 103) emphasized the importance of deciding what counts as a token in the respective indexing scheme. The author proposed entirely removing character sub-sequences comprising, for instance, digits and punctuation marks if they "do not make good index terms." Since the exact case of letters is usually irrelevant for retrieving information, Fox also advocated converting all characters into either upper or lower case unless case distinction is indeed necessary. In contrast, the tokenization step within our knowledge discovery process neither removes non-whitespace characters nor converts the case of characters for one reason: Named entity extraction described in the next subsection analyses and takes advantage of existing digits, special characters (e.g., "\$", "/", and "+"), and the case of letters. In particular, upper-case characters facilitate the automated recognition of named entities in English texts (e.g., "Philip Knisely" and "Constellation Capital" in text  $\check{t}_{E5}$ ).

Baeza-Yates and Ribeiro-Neto (1999, p. 167) pointed out that all lexical analysis operations can be implemented without difficulty. However, "careful thought should be given

**Table 4.3:** Tokenized Text Units after Tokenizing the Elements of Text Unit Layer  $\check{r}_{E1}$ 


---

$\check{u}_{E1,1}$ :	TokenizedTextUnit = $\langle$ "USA", ":", "VASCO", "says", "to", "sell", "consulting", "unit", ". $\rangle$
$\check{u}_{E1,2}$ :	TokenizedTextUnit = $\langle$ "VASCO", "Corp", "said", "it", "agreed", "to", "sell", "its", "consulting", "and", "technical", "organization", ",", "VASCO", "Performance", "System", ",", "to", "Wisdom", "Systems", "Inc." $\rangle$
$\check{u}_{E1,3}$ :	TokenizedTextUnit = $\langle$ "Terms", "were", "not", "disclosed", ". $\rangle$
$\check{u}_{E1,4}$ :	TokenizedTextUnit = $\langle$ "-", "Chicago", "Newsdesk", "312", "-", "408", "-", "8787" $\rangle$

---

**Algorithm 4.1** DecomposeAndTokenizeTextDocuments**Input:** ( $\check{a}$ : TextArchive)**Output:** ( $\bar{a}$ : IntTextArchive)

```

1:  $\bar{a}$ : IntTextArchive :=  $\bar{a}$ .create() // initialize output int. text archive
2: for all  $i$ : Integer := 1, 2, ...,  $\check{a}$ .size() do // iterate through text documents
3:    $\check{r}$ : TextUnitLayer :=  $\check{r}$ .create( $\check{a}$ .textDocument( $i$ )) // decompose text document
4:    $\bar{r}$ : IntTextUnitLayer :=  $\bar{r}$ .create() // initialize int. text unit layer
5:   for all  $j$ : Integer := 1, 2, ...,  $\check{r}$ .size() do // iterate through text units
6:      $\check{u}$ : TextUnit :=  $\check{r}$ .textUnit( $j$ ) // extract  $j$ th text unit
7:      $\check{u}$ : TokenizedTextUnit :=  $\check{u}$ .create( $\check{u}$ ) // tokenize  $j$ th text unit
8:      $\bar{u}$ : IntTextUnit :=  $\bar{u}$ .create( $\check{u}$ ,  $\check{u}$ , null, null, null, null, null) // create int. text unit
9:      $\bar{r}$ .appendIntTextUnit( $\bar{u}$ ) // append int. text unit
10:  end for
11:   $\bar{t}$ : IntTextDocument :=  $\bar{t}$ .create( $\check{a}$ .textDocument( $i$ ),  $\bar{r}$ ) // create int. text document
12:   $\bar{a}$ .appendIntTextDocument( $\bar{t}$ ) // append int. text document
13: end for

```

---

to each one of them” since lexical analysis operations may have a profound impact. For example, Fox (1992) presented dedicated algorithms and data structures for lexical analysis based on finite state machines. In our framework, the ADT TokenizedTextUnits therefore provides the constructor create( $\check{u}_{init}$ : TextUnit) that takes a text unit  $\check{u}_{init}$  as input and divides it into a sequence of tokens (see Appendix B.22 on page 229). The prototypical implementation of this tokenization algorithm within DIASDEM WORKBENCH is outlined in Subsection 5.2.1. To illustrate our notion of tokenization, Table 4.3 depicts four tokenized text units created by tokenizing the elements of text unit layer  $\check{r}_{E1}$ : TextUnitLayer.

**Effect on KDT Process Flow** Algorithm 4.1 (DecomposeAndTokenizeTextDocuments) encapsulates the initial pre-processing tasks of decomposing text documents and tokenizing the resulting text units. Additionally, this algorithm transforms the input text archive into an intermediate text archive required by subsequent document pre-processing tasks. Consequently, Algorithm 4.1 is the first algorithm of the KDT process flow to be parameterized during the interactive knowledge discovery.

**Table 4.4:** Excerpt from the Extended Named Entity Hierarchy (Source: Sekine et al., 2002) and Exemplary Named Entities

Named Entity Type	Exemplary Named Entities
(top)	
name	
person	"Philip Knisely"
organization	
company	"VASC0", "VASC0 Corp", "Wizdom Systems Inc.", "Helmerich"
market	"New York Stock Exchange"
location	"USA", "Chicago", "Baytown, Texas", "Woodland Hills, Calif."
product	"modular classrooms"
time_top	
timex	"Monday", "end of November"
date	"August 30", "Aug. 31", "October 21"
periodx	"8 months"
numex	"1,500-employee"
money	"\$48 million", "\$2.1 billion", "25 cents", "\$46.87"

## 4.2.2 Extracting Named Entities

Following text decomposition and tokenization, recognizing domain-specific named entities is the second main pre-processing step. Informally introduced in Section 1.2 and defined in Section 3.1, named entities are instantiations (e.g., "Wizdom Systems Inc.") of named entity types (e.g., company). More precisely, named entity types are abstract classes and generic numerical expressions whereas named entities are 2-tuples comprising the respective named entity type and a character sequence that represents a canonical form of the instantiated named entity type (cf. Definition 7 on page 59). Given the set  $P_{E1} := \{\text{person, company, location, money}\}$  of domain-specific named entity types, the excerpt "System, to Wizdom Systems Inc. Terms were" from text document  $\check{t}_{E1}$ , for example, contains the named entity  $e_{E1,4} := (\text{company, "Wizdom Systems Inc."})$ .

Named entity extraction is highly domain-dependent due to the variety of existing and possibly interesting named entities. Sekine et al. (2002), for instance, compiled a hierarchy of approx. 150 named entity types. Table 4.4 depicts a small excerpt from this hierarchy and exemplifies named entity types through named entities occurring in texts  $\check{t}_{E1}$  through  $\check{t}_{E5}$ . Besides time expressions (i.e., timeex), period expressions (i.e., periodx), and numerical expressions (i.e., numex), these five texts primarily contain various names of companies and locations. The latter named entity type is in fact the parent concept of more specific locations, such as city, country, or postal\_address. In addition to 150 named entity types listed in the named entity hierarchy proposed by Sekine et al., tagging documents from a narrow or exceptional domain might necessitate the instantiation of domain-specific named entity types, such as pharmaceutical\_substance, international\_security\_identification\_number, or statutory\_source.

Initially, domain experts thus have to carefully select relevant and domain-specific named entity types. Since extracted named entities ultimately serve as attributes of semantic XML tags, significant importance must be attached to instantiations of elements in the set of relevant named entity types. Furthermore, named entities to be identified should occur sufficiently frequently in the input text archive to justify their extraction. Being able to reliably identify certain named entities is by no means a reason to actually extract them. Extracting phone numbers from the five example texts, which only occur in the context of the publishing newsroom, adds no value if these news items are semantically annotated on behalf of Reuters itself. In contrast, identifying, for instance, names of cities in news items is an important prerequisite to distinguish between different branch offices of the same company based on their locations. Undoubtedly, instances of fundamentally important named entity types within one application domain may constitute meaningless tokens in another domain.

As the literature review in Subsection 2.1.3 has revealed, recognizing names of persons, companies, and locations in general English news items “was more or less a solved problem” in 1998 (Jackson and Moulinier, 2002, p. 77). In addition, both freely available (e.g., cf. Cunningham, 2002) and commercial (e.g., cf. Hammond et al., 2002; Ferrucci and Lally, 2004) tools nowadays allow for the extraction of domain-independent and domain-specific named entities. Due to the inherent domain dependency of named entity extraction, we refrain from presenting a specific algorithm. Instead, we subsequently make use of the operation `identifyNamedEntities(P: SetOfNamedEntityType)` defined by the abstract data type `IntTextUnit` (see Appendix B.26 on page 232). This operation takes a set of domain-specific named entity types as input and identifies their instantiations in the tokenized text unit associated with the respective intermediate text unit. By utilizing an abstract algorithm for named entity recognition, we again emphasize the necessity of domain experts to supply detailed descriptions of named entity types as an important input to our knowledge discovery process (cf. Figure 3.2 on page 65). These descriptions must be sufficiently precise to enable the re-use of adequate algorithms or to allow for the implementation of new, dedicated named entity extractors.

In our framework, neither the canonical forms of identified named entities (e.g., "2006-12-30") nor the respective tokens themselves, such as `<"Dec.", "30", ", ", "2006">`, constitute features in the pattern discovery step because they convey too fine-grained metadata. Unlike values of named entities, however, the mere fact that instances of certain named entity types occur in a text unit is indeed an auspicious feature. To find groups of semantically similar text units, measuring the frequency of occurrence of selected named entity types tends to provide more abstract insights into the semantic document structure than treating each named entity as an individual feature. As tokens that correspond to identified named entities are thus useless, we advocate replacing them with specific tokens, so-called named entity placeholders that are defined as follows:

**Definition 22 (Named Entity Placeholder)** *Given the intermediate named entity  $\bar{e}$ : `IntNamedEntity` identified in tokenized text unit  $\bar{u}$ : `TokenizedTextUnit`, a named entity placeholder is a token that uniquely identifies  $\bar{e}$  within the scope of its intermediate text*

**Table 4.5:** Processed Text Units of Intermediate Text Unit Layer  $\bar{r}_{E1}$  (cf. Table 4.3 on Page 79) after Extracting Named Entities

---

$\ddot{u}_{E1,1} = \langle \text{"USA"}, \text{":"}, \bar{e}_{E1,1}.\text{placeholder}(), \text{"says"}, \text{"to"}, \text{"sell"}, \text{"consulting"}, \text{"unit"}, \text{"."} \rangle$
$\ddot{u}_{E1,2} = \langle \bar{e}_{E1,2}.\text{placeholder}(), \text{"said"}, \text{"it"}, \text{"agreed"}, \text{"to"}, \text{"sell"}, \text{"its"}, \text{"consulting"}, \text{"and"}, \text{"technical"}, \text{"organization"}, \text{","}, \bar{e}_{E1,3}.\text{placeholder}(), \text{","}, \text{"to"}, \bar{e}_{E1,4}.\text{placeholder}() \rangle$
$\ddot{u}_{E1,3} = \langle \text{"Terms"}, \text{"were"}, \text{"not"}, \text{"disclosed"}, \text{"."} \rangle$
$\ddot{u}_{E1,4} = \langle \text{"-"}, \bar{e}_{E1,5}.\text{placeholder}(), \text{"Newsdesk"}, \text{"312"}, \text{"-"}, \text{"408"}, \text{"-"}, \text{"8787"} \rangle$

---

*document. A named entity placeholder substitutes all tokens that instantiate intermediate named entity  $\bar{e}$  in tokenized text unit  $\ddot{u}$ .*

For any intermediate named entity, the specific placeholder token is created by executing the operation `placeholder(): Token`, which is provided by the abstract data type `IntNamedEntity` (see Appendix B.23 on page 230). Subsequent to named entity identification, all tokens that correspond to named entities are replaced by their respective placeholders. Thereby, we considerably reduce the dimensionality of textual data in terms of the number of distinct tokens occurring in a text archive. To that end, the abstract data type `IntTextUnit` (see Appendix B.26 on page 232) defines an operation `replaceNamedEntities()`. This operation can be implemented straightforwardly since it only replaces all tokens that instantiate an intermediate named entity identified in the respective tokenized text unit with appropriate named entity placeholders.

To illustrate our notion of named entity extraction, Table 4.5 depicts the processed, tokenized text units  $\ddot{u}_{E1,1}$ : `TokenizedTextUnit` through  $\ddot{u}_{E1,4}$ : `TokenizedTextUnit`, which are contained in the intermediate text unit layer  $\bar{r}_{E1}$ : `IntTextUnitLayer` (cf. Table 4.3 on page 79), after both identification and replacement of named entities. Furthermore, the identified intermediate named entities along with their placeholders are listed in Table 4.6. For example, the tokenized text unit  $\ddot{u}_{E1,2}$  contains placeholders for three named entities. More specifically, the expression  `$\bar{e}_{E1,2}.\text{placeholder}()$`  stands for the intermediate named entity "VASC0 Corp" of type `company` that corresponds to tokens 1 and 2 of its parent tokenized text unit  $\ddot{u}_{E1,2} = \bar{r}_{E1}.\text{processedTextUnit}(2)$ .

**Effect on KDT Process Flow** Algorithm 4.2 (`ExtractNamedEntities`) performs the pre-processing tasks of identifying named entities in tokenized text units and replacing them with named entity placeholders. As input, this algorithm takes a set of domain-specific named entity types and an intermediate text archive. Subsequent to their extraction from text units, named entities are stored in the set of intermediate named entities of the respective intermediate text units. Following `DecomposeAndTokenizeTextDocuments`, `ExtractNamedEntities` is the second algorithm in the KDT process flow that has to be parameterized during the knowledge discovery process. Named entity identification is

**Table 4.6:** Intermediate Named Entities Identified in Tokenized Text Units of Intermediate Text Unit Layer  $\bar{t}_{E1}$  (cf. Table 4.3 on Page 79)

Intermediate Named Entity	Named Entity Placeholder
$\bar{e}_{E1,1}$ : IntNamedEntity = (company, "VASC0", 1, 3, 3)	$\bar{e}_{E1,1}$ .placeholder()
$\bar{e}_{E1,2}$ : IntNamedEntity = (company, "VASC0 Corp", 2, 1, 2)	$\bar{e}_{E1,2}$ .placeholder()
$\bar{e}_{E1,3}$ : IntNamedEntity = (company, "VASC0 Performance System", 3, 13, 15)	$\bar{e}_{E1,3}$ .placeholder()
$\bar{e}_{E1,4}$ : IntNamedEntity = (company, "Wizdom Systems Inc.", 4, 18, 20)	$\bar{e}_{E1,4}$ .placeholder()
$\bar{e}_{E1,5}$ : IntNamedEntity = (place, "Chicago", 5, 2, 2)	$\bar{e}_{E1,5}$ .placeholder()

**Algorithm 4.2** ExtractNamedEntities**Input:** ( $\bar{a}$ : IntTextArchive, P: SetOfNamedEntityType)**Output:** ( $\bar{a}$ : IntTextArchive)

```

1: for all  $i$ : Integer := 1, 2, ...,  $\bar{a}$ .size() do // iterate through int. text documents
2:   for all  $j$ : Integer := 1, 2, ...,  $\bar{a}$ .intTextUnitLayerSize( $i$ ) do // iterate through int. text units
3:      $\bar{a}$ .intTextUnit( $i, j$ ).identifyNamedEntities(P) // identify named entities
4:      $\bar{a}$ .intTextUnit( $i, j$ ).replaceNamedEntities() // replace named entities with placeholders
5:   end for
6: end for

```

deliberately executed before lemmatization, which is described in the next subsection, because replacing words with their grammatical root forms may trigger negative side effects on named entity extraction.

### 4.2.3 Lemmatizing Words and Word Sense Disambiguation

According to the pre-processing phases outlined in Subsection 2.1.2, lexical analysis, or tokenization, is followed by the removal of meaningless, though frequently occurring, stopwords (Baeza-Yates and Ribeiro-Neto, 1999, p. 167). In an IR context, stopwords are removed because they have a very low discriminatory power for retrieval purposes. Besides, their elimination reduces the index size considerably. In our framework, removing stopwords is not compulsory since we do not build a persistent document index. The straightforward (cf. Fox, 1992) procedure of stopword removal is optional and hence deliberately omitted from discussion in this section on text pre-processing. Having tokenized text units and extracted named entities, the third main document pre-processing step in our KDT process is lemmatization and the optional word sense disambiguation.

**Lemmatization** Due to the more or less complex syntax rules of natural languages, text documents mostly contain many different word variants. The singular and plural noun forms (e.g., "acquisition" in text document  $\check{t}_{E2}$  and "acquisitions" in text  $\check{t}_{E5}$ ) or conjugated verb forms (e.g., "sells" and "sell" in text  $\check{t}_{E5}$ ) are examples of syntactical

word variations that do not regularly affect the word meaning. During document pre-processing, these word variants are automatically mapped onto their grammatical root forms (e.g., "acquisition" and "sell"). Thereby, the lemmatization step facilitates the use of a controlled vocabulary. The benefits of lemmatization become apparent when processing text documents composed of a highly inflective language, like German.

Lemmatization is the activity of determining lemma forms of natural language words. The term lemma denotes a lexical entry that represents all inflected forms of one word in a dictionary (cf. Bowker and Pearson, 2002, pp. 166–167). For example, the infinitive is used as the lemma of conjugated verbs whereas nouns are typically represented by their singular forms (Saeed, 2003, p. 56). To that end, lemmatization algorithms utilize the part of speech information (e.g., noun vs. verb) of inflected word forms. Unlike the notion of lemma forms, the term stem denotes the portion of a word that is left after removing its affixes, such as prefixes and suffixes (Baeza-Yates and Ribeiro-Neto, 1999, pp. 168–169). Stems do not necessarily correspond to lexical entries. For instance, the token "connect" is the stem of the inflected verb forms "connected" and "connecting", as well as the noun forms "connection" and "connections". This example illustrates that stemming does not utilize the part-of-speech information of words to be stemmed.

We advocate employing a sophisticated lemmatization algorithm instead of simply stemming words because lemma forms convey more semantic information than word stems. Manning and Schütze (1999, p. 132), for instance, noticed that stemming “often costs you a lot of information.” Since canonical dictionary entries depend on the part-of-speech information of words, their meaning can be more easily recognized by domain experts while establishing a domain-specific controlled vocabulary (cf. next subsection). Unlike stems, lexical units can be looked up directly in a conventional dictionary or in a lexical database (e.g., WORDNET; cf. Fellbaum, 1998). In addition, content-descriptive labels for qualitatively acceptable text unit clusters are automatically suggested on the basis of frequently occurring features and text unit descriptors, respectively (cf. Subsection 4.4.1). Again, the semantic expressiveness of word stems, such as "comput", is rather limited when compared with lemma forms (e.g., "computer", "computational", "compute", or "precompute"). Especially for less inflective languages like English, however, the coarser results of stemming words can be considered an approximation of lemmatizing them.

Frakes (1992) surveyed four different approaches to stemming English words. According to Baeza-Yates and Ribeiro-Neto (1999, pp. 168–169), the seminal suffix removal algorithm proposed by Porter (1980a) is the most popular one due to its simplicity. This algorithm stems English words by applying a series of rules for suffix stripping. In the challenging domain of automated lemmatization, Schmid (1994, 1999), for example, introduced a language-independent, fast, high-quality part-of-speech tagger based on the supervised learning paradigm. Besides determining the parts of speech, TREETAGGER outputs the lemma form of words as well. The system achieved an accuracy of between 96.05% and 97.53% in experiments with a German newspaper corpus.

Consequently, we do not suggest one specific lemmatization algorithm, but utilize the abstract operation `lemmatizeWords()` defined by the abstract data type `IntTextUnit` (see

Appendix B.26 on page 232). When executed on a specific intermediate text unit, this operation modifies the processed, tokenized text unit contained in the intermediate text unit. Concretely, `lemmatizeWords()` replaces all tokens that are natural language words in the tokenized text unit with their grammatical root forms.

**Word Sense Disambiguation** Natural language expressions can be highly ambiguous. In particular, lexical ambiguity arises when one word has many senses and meanings. Homonyms are unrelated senses of the same phonological word. Homonyms can be further subdivided into homographs, namely, unrelated senses of the same written word, and homophones that are unrelated senses of the same spoken word (Saeed, 2003, pp. 63–64). In contrast to homonyms, polysemes are different, yet somehow related, senses of one phonological word. Since our framework is not rooted in the discipline of lexicology, we differentiate neither between homonymy and polysemy nor between the two forms of homonymy although only homographs are relevant to our framework. Henceforth, multiple senses of the same written word are thus referred to as homonyms.

Word sense disambiguation is the activity of assigning words in a text to their appropriate senses listed in a lexicon (cf. Leacock and Chodorow, 1998, p. 265). This task determines the sense of an ambiguous word, which is invoked in a concrete use of this word (Manning and Schütze, 1999, p. 229). For example, the lemmatized token "term" occurring in the text unit "Terms were not disclosed." is highly ambiguous unless the context in which it appears is taken into consideration. The English lexical database WORDNET (cf. Fellbaum, 1998), for example, lists seven different senses, or homonyms, for the noun "term". In the third text unit of news item  $\check{t}_{E1}$ , this noun is used in the sense 'statement of what is required as part of an agreement'.

Manning and Schütze (1999, p. 132) emphasized that lemmatization implies disambiguation because the token "lying" may, for example, represent either the lexical entry 'making a statement one knows to be untrue' or 'having one's body in a flat position.' In the pre-processing step of our framework, nevertheless, we separate the mandatory lemmatization step, which converts words into their lemma forms, from the optional word sense disambiguation step, which maps ambiguous lemmas onto their appropriate senses, for three reasons. Firstly, existing lemmatization systems (e.g., TREETAGGER, cf. Schmid, 1999) do not necessarily disambiguate word senses as a byproduct. Secondly, word sense disambiguation may not be necessary for text documents originating from a very narrow or exceptional domain and which are thus less ambiguous. Since automated word sense disambiguation is error-prone as well, its application may thirdly be restricted to a few important words (e.g., "interest" in business news items) that are both frequently used and associated with multiple domain-specific senses.

A large number of algorithms for automated word sense disambiguation are available (e.g., Leacock and Chodorow, 1998; Manning and Schütze, 1999, p. 235–256). Therefore, our framework only exploits the operation `disambiguateWordSenses` provided by the ADT `IntTextUnit` (see Appendix B.26 on page 232). This abstract operation modifies the tokenized text unit associated with an intermediate text unit by appending a sense tag,

**Algorithm 4.3** LemmatizeAndDisambiguateWords**Input:** ( $\bar{a}$ : IntTextArchive)**Output:** ( $\bar{a}$ : IntTextArchive)

---

```

1: for all  $i$ : Integer := 1, 2, ...,  $\bar{a}.size()$  do // iterate through int. text documents
2:   for all  $j$ : Integer := 1, 2, ...,  $\bar{a}.intTextUnitLayerSize(i)$  do // iterate through int. text units
3:      $\bar{a}.intTextUnit(i, j).lemmatizeWords()$  // lemmatize natural language words
4:      $\bar{a}.intTextUnit(i, j).disambiguateWordSenses()$  // disambiguate word senses
5:   end for
6: end for

```

---

**Table 4.7:** Processed Text Units of Intermediate Text Unit Layer  $\bar{r}_{E1}$  (cf. Table 4.3 on Page 79) after Lemmatization and Word Sense Disambiguation

---

```

 $\bar{u}_{E1,1} = \langle \text{"USA", ":", } \bar{e}_{E1,1}.placeholder(), \text{"say", "to", "sell", "consulting", "unit", "."} \rangle$ 
 $\bar{u}_{E1,2} = \langle \bar{e}_{E1,2}.placeholder(), \text{"say", "it", "agree", "to", "sell", "its", "consulting", "and", "technical", "organization", ",", } \bar{e}_{E1,3}.placeholder(), \text{",", "to", } \bar{e}_{E1,4}.placeholder() \rangle$ 
 $\bar{u}_{E1,3} = \langle \text{"term/s:agreement", "be", "not", "disclose", "."} \rangle$ 
 $\bar{u}_{E1,4} = \langle \text{"-", } \bar{e}_{E1,5}.placeholder(), \text{"Newsdesk", "312", "-", "408", "-", "8787"} \rangle$ 

```

---

which indicates the concrete word sense, to all tokens that are homonymous natural language words. Ultimately, the extent of sense disambiguation depends on linguistic characteristics and the semantic diversity of the domain.

**Effect on KDT Process Flow** Given an intermediate text archive, Algorithm 4.3 (LemmatizeAndDisambiguateWords) lemmatizes and disambiguates word senses in intermediate text units. Following DecomposeAndTokenizeTextDocuments and ExtractNamedEntities, LemmatizeAndDisambiguateWords is the third algorithm to be executed in the pre-processing phase of our interactive knowledge discovery process. The implementation of these two pre-processing tasks in DIAsDEM WORKBENCH is outlined in Subsection 5.2.1.

Table 4.7 illustrates the effect of applying the algorithm LemmatizeAndDisambiguateWords to the processed text units of intermediate text unit layer  $\bar{r}_{E1}$ . For example, the inflected word forms "says" and "said" are replaced with their infinitive "say". The capitalized plural noun "Terms" is substituted by its canonical form "term". In addition, the appropriate sense tag "/s:agreement" is appended to the lemmatized token "term", which results in the lemmatized and disambiguated token "term/s:agreement".

#### 4.2.4 Establishing a Controlled Vocabulary

As explained in Subsection 2.1.2, selecting index terms and constructing a term categorization structure (e.g., thesaurus) are the fourth and fifth, respectively, text operation

of document pre-processing in information retrieval. In the pre-processing phase of our knowledge discovery process, these highly interdependent steps are combined into the semi-automated task of establishing a domain-specific controlled vocabulary. The result provides the basis for transforming pre-processed text units into a numerical representation suitable for pattern discovery, as discussed in Subsection 4.2.5. In this subsection, we first introduce our approach to feature<sup>1</sup> selection and explain the rationale behind it. Secondly, we define our notion of a controlled vocabulary and related terminology for the scope of this work. Thirdly, the usage of domain-specific thesauri as controlled vocabularies is discussed. Finally, we outline supportive techniques that facilitate the laborious, but nonetheless crucial, task of establishing a controlled vocabulary.

**Feature Selection** After intensive pre-processing, text units are mapped onto numerical text unit vectors that in turn constitute the structured input data to a conventional clustering algorithm in the pattern discovery step. However, what numerical features shall represent pre-processed, though nevertheless unstructured, text units? This fundamental question is addressed by feature selection<sup>2</sup> as part of the transformation step in the generic KDT process, which is depicted in Figure 2.1 on page 22.

The higher the dimensionality (i.e., the number of features) of a fixed-size input data set, the worse the performance of many knowledge discovery algorithms tends to be in terms of result quality and required resources (e.g., cf. Hand et al., 2001, pp. 193–196). This phenomenon is well known as the curse of dimensionality, “a malediction that has plagued the scientist from earliest days” (Bellman, 1961, p. 94). In particular, the performance of clustering algorithms (Aggarwal and Yu, 2000, pp. 70–71) and classification algorithms (Sebastiani, 1999, p. 6) tends to decline drastically as the dimensionality of the input data increases. According to Hand et al. (2001, p. 194), feature selection is a fairly general, but nevertheless sensible, strategy when analyzing high-dimensional data. Selecting appropriate features refers to choosing a subset of the original variables by eliminating redundant and uninformative ones (Kim et al., 2000, p. 365). Feature selection may also involve constructing new features from the original ones (Guyon and Elisseeff, 2003, p. 84). Clearly, finding useful features ultimately depends on the objectives of the knowledge discovery task at hand (Fayyad et al., 1996b, p. 10). Since clusters of semantically similar text units shall be identified in the pattern discovery phase of our KDT process, features of text units must properly reflect their content.

Feature selection techniques are broadly divided into supervised methods for classification tasks and unsupervised algorithms for clustering tasks (cf. Yang and Pedersen, 1997; Guyon and Elisseeff, 2003). In pursuing our knowledge discovery objectives, we focus on the latter techniques since unsupervised learning techniques (i.e., clustering algorithms) are employed in our pattern discovery phase. Unlike training data sets prepared for text classification tasks, text units in training archives are not assigned to a priori known

---

<sup>1</sup>The terms variable, attribute, and measurement are synonyms of the term feature in this work.

<sup>2</sup>Variable selection and dimensionality reduction are synonyms of feature selection in this work.

semantic concepts because our framework aims at inferring them. In principle, three unsupervised approaches to selecting features of text units can be distinguished:

1. Choosing all available features without performing any selection constitutes the naive approach. Unless the number of remaining distinct terms after pre-processing the input archive is relatively small, their meanings are reliably disambiguated, and synonymous terms do not occur, this simplistic approach is disadvantageous due to the effects related to the curse of dimensionality. In contrast to dimension-sensitive KDT applications, however, full text indexing is commonly employed by Web-scale information retrieval systems (cf. Baeza-Yates and Ribeiro-Neto, 1999, p. 171).
2. Manual, semi-automated, or automated feature selection involves identifying a limited number of semantic-carrying content descriptors (cf. Subsection 2.1.2) that are capable of representing the topicality, or the subject, of documents. Research in information retrieval has produced a variety of techniques to select appropriate content identifiers and index terms, respectively (cf. Baeza-Yates and Ribeiro-Neto, 1999, pp. 169–170). For example, representative methods for topic identification in text have been reviewed in Subsection 2.2.1. Once content descriptors are manually, or at best automatically, chosen (e.g., cf. Moens, 2000, pp. 77–102), they form a reduced feature space and correspond to a domain-specific controlled vocabulary or, synonymously, an indexing language. Each text unit is henceforth represented by the text descriptors that occur therein (cf. Korfhage, 1997, pp. 105–110).
3. Automated feature creation methods map text unit vectors from a high-dimensional input feature space onto a significantly lower-dimensional feature space. These techniques compute novel numerical variables that make hidden semantic relationships between different terms as explicit as possible. For example, latent semantic indexing exploits term co-occurrences within documents and drastically reduces the feature space by computing the singular value decomposition of the term-by-document matrix (Deerwester et al., 1990; Manning and Schütze, 1999, pp. 554–564). Other techniques, such as concept decomposition (Dhillon and Modha, 2001) and concept indexing (Karypis and Han, 2000b), employ clustering algorithms to first identify  $k$  document clusters and subsequently use their centroid vectors to form a  $k$ -dimensional feature space onto which document vectors are finally mapped.

The DIASDEM framework adopts the second approach to feature selection. We favor establishing a domain-specific controlled vocabulary over the alternatives for three reasons. Firstly, text units have to be represented such that their semantic content is retained as much as possible (cf. Hand et al., 2001, p. 457). Since computational problems of clustering algorithms induced by a high-dimensional feature space must be avoided simultaneously, treating each distinct term as an individual feature is ruled out. The remaining two reasons in favor of creating a controlled vocabulary are related to the objective of our framework, namely, deriving content-descriptive, human-readable cluster labels and names of semantic

XML tags, respectively. Secondly, we consequently refrain from utilizing methods that compute new numerical features because techniques like latent semantic indexing make the interpretation of clustering results more difficult (Liu et al., 2003, p. 488). Guyon and Elisseeff (2003, p. 1159) emphasized the importance of utilizing domain knowledge during feature selection. Assuming the existence of domain knowledge (e.g., common synonyms of terms) on the part of the human expert, a controlled vocabulary thirdly allows for its seamless integration into the KDT process.

Undoubtedly, creating and maintaining a domain-specific controlled vocabulary requires costly human efforts. Even methods for the automatic establishment of controlled vocabularies have to be supervised to ensure a high quality of the resulting term collections. Nevertheless, we strongly advocate constructing a dedicated controlled vocabulary during feature selection as it forms an important basis for generating high-quality semantic markup. We briefly survey techniques that support domain experts in selecting appropriate content identifiers after introducing additional, framework-specific terminology and discussing the use of thesauri as specific controlled vocabularies.

**Terminology** Korfhage (1997, p. 317) defined the term controlled vocabulary as a “restricted set of words and phrases that are used to describe documents” in a given text archive. Colomb (2002, p. 87) characterized a controlled vocabulary as an information structure that is a “collection of terms from which descriptors are drawn.” This set of text descriptors<sup>3</sup> is used to concisely represent or summarize the content of associated full text documents (Colomb, 2002, p. 31). Prior to clarifying our terminology related to controlled vocabularies, the auxiliary notion of atomic concepts shall be introduced:

**Definition 23 (Atomic Concept)** *An atomic concept is one unit of thought whose semantic content can be re-expressed by a combination of other and different concepts.*

The above definition is deliberately analogous to the standardized notion of concepts in the context of subject indexing (cf. ISO 5963, 1985, p. 1). In our framework, however, atomic concepts are intentionally distinguished from concepts (cf. Definition 6 on page 59). The latter, higher-level notion of concepts denotes a thought that domain experts typically associate with a group of semantically similar text units, such as sentences or paragraphs. Unlike atomic concepts (e.g., **Acquisition** or **Announcement**), concepts concisely summarize text units. More specifically, concepts are mostly combinations of several atomic concepts (e.g., **AcquisitionAnnouncement**). Our framework-specific distinction between atomic concepts and higher-level, typically composite concepts is inspired by the theory of conceptual atomism, as discussed by Laurence and Margolis (1999, pp. 59–71). This cognitive science theory stipulates that lexical concepts, which we refer to as atomic concepts, are primitive and have no structure. Relevant atomic concepts are either represented by text unit descriptors or text unit non-descriptors.

---

<sup>3</sup>The terms content identifier, keyword, and index term are considered as synonyms of text descriptor.

**Definition 24 (Text Unit Descriptor)** *A text unit descriptor is a token that is consistently used to represent one atomic concept when characterizing the content of text units.*

Our notion of text unit descriptors<sup>4</sup> intentionally resembles the standardized definition of a preferred term in the context of indexing documents (cf. ISO 5963, 1985, p. 1). For example, text unit descriptors denote natural language words (e.g., the tokens "announcement" and "acquisition") or named entity types (e.g., company). The latter are obtained via named entity placeholders that occur in pre-processed text units. Eventually, domain-specific descriptors correspond to the dimensions of the reduced feature space required to map pre-processed text units onto vectors of moderate dimensionality.

Text unit non-descriptors are a second category of tokens that are not directly a part of the feature space. Nonetheless, they convey important semantic information as each text unit non-descriptor represents an atomic concept that is equal or otherwise semantically related to an atomic concept represented by a text unit descriptor.

**Definition 25 (Text Unit Non-Descriptor)** *A text unit non-descriptor is a token that (i) represents one atomic concept and (ii) directly or indirectly references one associated text unit descriptor. The atomic concept represented by a text unit non-descriptor is equal or otherwise semantically related to the atomic concept of its associated text unit descriptor. When characterizing the content of text units, the associated text unit descriptor is consistently used to represent the atomic concept of a text unit non-descriptor.*

The above definition of text unit non-descriptors<sup>5</sup> is broader than the standardized notion of a non-preferred term in the context of subject indexing (cf. ISO 5963, 1985, p. 1). A non-descriptor and its associated descriptor are synonyms if they denote the same atomic concept with different tokens. Unlike non-preferred terms defined by ISO 5963 (1985), the relationship between atomic concepts of a non-descriptor and its associated descriptor can also be one of hyponymy (i.e., relation of inclusion; cf. Saeed, 2003, pp. 68–70), meronymy (i.e., part–whole relationship; cf. Saeed, 2003, pp. 70–71), or any other conceivable kind. For example, the disambiguated and sense-tagged token "statement/s:announcement" is a non-descriptor associated with the synonymous text unit descriptor "announcement". Furthermore, the exemplary non-descriptor "takeover" references a broader term, namely, the descriptor "acquisition". Thus, the content of text units comprising the tokens "statement/s:announcement" and "takeover" is characterized by the descriptors "announcement" and "acquisition".

Having introduced text unit descriptors and non-descriptors, we proceed by defining our notion of controlled vocabulary terms and our notion of controlled vocabularies.

**Definition 26 (Controlled Vocabulary Term)** *The 4-tuple  $v := (\check{w}: \text{Token}, i: \text{Integer}, d: \text{Boolean}, j: \text{Integer})$  is a controlled vocabulary term comprising the token  $\check{w}$ , a unique term identifier  $i$ , the boolean variable  $d$  whose value indicates the term type, and the term*

---

<sup>4</sup>Henceforth, the term descriptor is a synonym of the term text unit descriptor.

<sup>5</sup>Henceforth, the term non-descriptor is a synonym of the term text unit non-descriptor.

identifier  $j$  of an optionally associated controlled vocabulary term. If  $d = \text{true}$ , controlled vocabulary term  $v$  is a text unit descriptor and  $j = \text{null}$ . Otherwise, controlled vocabulary term  $v$  is a text unit non-descriptor and consequently  $j \neq \text{null}$ .

The abstract data type `ControlledVocabularyTerm` (see Appendix B.30 on page 235) encapsulates one controlled vocabulary term. For instance, the controlled vocabulary term ("`statement/s:announcement`", 2, false, 1) is a non-descriptor that directly references the associated controlled vocabulary term ("`announcement`", 1, true, null), which represents a text unit descriptor. Furthermore, the controlled vocabulary term ("`public statement`", 3, false, 2) illustrates an indirect reference from a non-descriptor via the directly referenced non-descriptor "`statement/s:announcement`", which in turn provides a link to the associated descriptor "`announcement`". Finally, a set of controlled vocabulary terms constitutes a controlled vocabulary within the scope of our framework:

**Definition 27 (Controlled Vocabulary)** *The non-empty set  $V := \{v_1, v_2, \dots, v_{|V|}\}$  is a controlled vocabulary comprising the controlled vocabulary terms  $v_i$ : `ControlledVocabularyTerm`, where  $i = 1, 2, \dots, |V|$ , such that any token is represented by at most one controlled vocabulary term, and there exists one directly or indirectly associated text unit descriptor for each text unit non-descriptor. All text unit descriptors in controlled vocabulary  $V$  are elements of the non-empty set  $V_D := \{v \mid v: \text{ControlledVocabularyTerm} \in V \text{ s.t. } v.\text{isDescriptor}() = \text{true}\} \subseteq V$  and satisfy the constraint  $v_j.\text{id}() = j$ , where  $v_j: \text{ControlledVocabularyTerm} \in V_D$  and  $j = 1, 2, \dots, |V_D|$ .*

One controlled vocabulary is encapsulated by the ADT `ControlledVocabulary` (see Appendix B.31 on page 235). Ultimately, the dimensionality of text unit vectors, which numerically represent pre-processed text units, is determined by the number of selected descriptors and the cardinality of the set  $V_D$  comprising all descriptors, respectively.

The set of descriptors occurring in a text unit constitutes a limited representation of textual content, which is referred to as a text unit surrogate (cf. Korfhage, 1997, pp. 21–24). Counting and appropriately weighting the frequency of descriptors in text units is a prerequisite for transforming text unit surrogates into vectors. This structured and numerical representation of text unit surrogates is required by most knowledge discovery algorithms. Although text unit surrogates represent as much semantic content as perceived necessary by the domain expert, they do not reflect all available information about the underlying text unit. For example, this bag-of-descriptors representation of textual content entirely, yet intentionally, disregards the ordering of terms.

Ideally, a controlled vocabulary represents a consistent set of domain-specific terms (i.e., atomic concepts) that are appropriate to characterize the content of text units. Taking the information retrieval perspective, Korfhage (1997, pp. 108–109) and Colomb (2002, pp. 34–35) discussed two desirable, application-independent properties of indexing languages. Firstly, specificity denotes the depth of coverage of index terms. A higher specificity is reflected by less abstract index terms and vice versa. In our framework, the adequate level of specificity of text unit descriptors depends on domain-specific requirements. As

**Table 4.8:** Excerpt of ISO-2788 Thesaurus for Text Documents  $\check{t}_{E1}$  through  $\check{t}_{E5}$  and Corresponding DIAsDEM-Specific Controlled Vocabulary Terms

Thesaurus Entry	Controlled Vocabulary Term
announcement	("announcement", 1, true, null)
UF statement (announcement)	
statement (announcement)	("statement/s:announcement", 2, false, 1)
USE announcement	
stock	("stock", 4, true, null)
NT common stock	
common stock	("common stock", 5, false, 4)
BT stock	

the number of descriptors increases proportionally to specificity, however, the positive impact of more specific descriptors has to be balanced with negative effects related to the curse of dimensionality. Secondly, exhaustivity refers to the breadth of coverage of index terms. An indexing language is exhaustive if each document can be assigned at least one content identifier. In contrast to information retrieval systems, complete exhaustivity is not essential in our framework because text archives often contain less important text units as well. Text units are treated as outliers if they neither exhibit a clustering tendency nor contain noteworthy themes. However, exhaustivity must be ensured with respect to the truly relevant atomic concepts.

**Thesaurus** To facilitate the effective re-use of existing terminology collections, which may conform to diverse standards and specifications, our notion of controlled vocabularies is deliberately based on a small number of requirements. Besides simple pre-compiled lists of tokens chosen to serve as text unit descriptors, more sophisticated controlled vocabularies (e.g., taxonomies, thesauri, or ontologies; cf. Subsection 2.2.3) can be utilized within our framework for semantic XML tagging. Given an arbitrary token, these different types of controlled vocabularies can be used to look up the associated text unit descriptor, if one exists. Despite the large number of possible kinds of controlled vocabularies, we suggest using monolingual thesauri since they are a common form of vocabulary control (Moens, 2000, p. 106) and thus widely used both in information retrieval (Baeza-Yates and Ribeiro-Neto, 1999, pp. 170–171) and knowledge discovery (Sullivan, 2001, p. 196). In addition, an international standard (ISO 2788, 1986) furthers the re-use and interchangeability of the great variety of existing, typically domain-specific thesauri (cf. Colomb, 2002, pp. 161–162).

According to ISO 2788 (1986, p. 1), thesauri indicate a priori known, generally recognized, and thus document-independent relationships between atomic concepts denoted by terms. A thesaurus constitutes an agreed-upon indexing language that enables indexers to consistently summarize the subjects of documents and to select index terms by

means of determining the a posteriori, document-dependent relationships between atomic concepts represented by terms. Our framework pursues a similar approach: Domain-specific knowledge in the form of a priori known relationships between atomic concepts (e.g., "statement/s:announcement" is a synonym of "announcement") is exploited during feature selection to ultimately identify and label high-quality concepts, which occur at the text unit level, in the pattern discovery phase. Labels of discovered concepts and names of semantic XML tags, respectively, analogously reflect a posteriori relationships of atomic concepts occurring in semantically similar text units. For the purposes of our framework, thesauri are hence a suitable kind of controlled vocabulary. Thesaurus entries can be mapped onto controlled vocabulary terms.

Table 4.8 illustrates the mapping of thesaurus entries onto DIASDEM-specific controlled vocabulary terms. The left column lists thesaurus entries in ISO-2788 format, which represent domain knowledge in the form of a priori known relationships between atomic concepts. Firstly, the sense of homonymous terms is disambiguated by a supplementary qualifying word in parentheses, like 'statement (announcement)'. Secondly, the abbreviations USE (i.e., use preferred term) and UF (i.e., used for non-preferred term) indicate the equivalence relationship between terms, which are regarded as referring to the same atomic concept. For instance, the terms 'announcement' and 'statement (announcement)' are synonyms, but only the former is a preferred indexing term. Thirdly, different types of hierarchical relationships between atomic concepts are made explicit by the abbreviations BT (i.e., broader term) and NT (i.e., narrower term). For example, 'common stock' is a specific kind of 'stock'. The right column of Table 4.8 indicates the controlled vocabulary terms associated with each thesaurus entry such that preferred indexing terms correspond to text unit descriptors. Narrower index terms, like 'common stock', are considered to be non-descriptors referencing the respective broader term.

**Supporting the Domain Expert** Since text unit descriptors play a key role in retaining semantic meaning when representing textual content as vectors, codifying the relevant archive-specific background knowledge is a pre-processing task of paramount importance. A domain-specific controlled vocabulary can be constructed either by establishing a completely new one or preferably by re-using an existing one. To minimize the costs associated with the interactive knowledge discovery phase, pre-compiled and thematically relevant terminology collections should be re-used and possibly modified or extended. If this preferred option cannot be exercised, however, constructing a controlled vocabulary is by no means confined as a manual task. For example, various techniques for topic discovery in texts (see Subsection 2.2.1) as well as many different methods for learning taxonomies, thesauri, and ontologies (see Subsection 2.2.3) are easily applicable to at least semi-automate the process of constructing a new controlled vocabulary. In addition, Korfhage (1997, pp. 114–125) and Moens (2000, pp. 77–102) summarized applicable techniques for (semi-) automatically selecting useful index terms.

Our framework targets domain-specific, as opposed to general, text archives. Therefore, the terminological characteristics of sublanguages (cf. Sager, 1986) or languages for

special purposes (cf. Bowker and Pearson, 2002, pp.25–27) can be exploited to relieve the laborious task of establishing a controlled vocabulary from scratch. A sublanguage considerably differs from language as a whole by specializations in syntax and the existence of a specific vocabulary (Sager, 1986, p. 1). Thus, domain experts benefit from employing (semi-) automated term extraction methods capable of identifying characteristic words of specialized subject fields (Bowker and Pearson, 2002, p. 26 and pp. 165–174). For instance, Feldman et al. (1999) described a term extraction module that identifies and filters relevant terms in a document collection by applying both linguistic and statistical techniques. Frequently, sublanguage terms are collocations. These multi-part tokens, like "preferred stock", denote specific atomic concepts that are independent of the atomic concepts represented by their components. Manning and Schütze (1999, pp. 151–189), for example, gave an extensive overview of helpful computer-linguistic algorithms for identifying collocations in text archives.

### 4.2.5 Mapping Text Units onto Text Unit Vectors

After establishing a domain-specific controlled vocabulary, the pre-processing step of our knowledge discovery process is concluded by transforming tokenized and fully pre-processed text units into numerical text unit vectors. To that end, we adopt the vector-space model introduced by Salton (1968, pp. 236–243) and summarized in Subsection 2.1.2. Originally proposed to efficiently store and retrieve textual content, the vector-space model is a widely recognized numerical text representation for knowledge discovery purposes as well (Sullivan, 2001, pp. 328–337). In particular, many clustering algorithms, which are frequently employed in KDT applications, require a transformation of input documents into real-valued vectors (e.g., cf. Steinbach et al., 2000; Zhao and Karypis, 2002). Firstly, we thus introduce framework-specific properties of text unit vectors. Secondly, different approaches to descriptor weighting are discussed before we illustrate the mapping of text units onto vectors using a simplified example.

**Terminology** In the vector-space model, pre-processed text units, each of them part of an intermediate text unit (see Definition 18 on page 74), are represented as vectors in a feature space whose dimensions correspond to text unit descriptors of the controlled vocabulary established beforehand. Each vector component is referred to as a weight that reflects the importance of the respective content identifier in the associated text unit (cf. Salton, 1968, p. 236). A weight of zero indicates that the corresponding text unit descriptor either does not occur in or is not applicable to the associated text unit. On the other hand, a descriptor weight other than zero indicates that the respective descriptor (i) occurs in the associated text unit and (ii) is weighted proportionally to the size of the corresponding real-valued vector coefficient. Thus, our preliminary notion of text unit vectors (see Definition 17 on page 74) is extended as follows:

**Definition 28 (Text Unit Vector)** *Given the controlled vocabulary  $V$ : ControlledVocabulary and the tokenized text unit  $\ddot{u}$ : TokenizedTextUnit, a text unit vector  $\mathbf{u} \in \mathbb{R}^n$  is an*

$n$ -dimensional vector such that  $n = V.\text{numberOfDescriptors}()$ . The component  $v_i \in \mathbb{R}$ , where  $i = 1, 2, \dots, n$ , of text unit vector  $\mathbf{u} := [v_1, v_2, \dots, v_n]^T$  represents the weight of text unit descriptor  $V.\text{descriptor}(i)$  in tokenized text unit  $\ddot{u}$ .

According to Salton (1968, p. 236), vector components represent arbitrary document properties. In our framework, vector space dimensions correspond to descriptor tokens that either denote atomic concepts or named entity types. Ultimately, the vector dimensionality is determined by the number of text unit descriptors in the controlled vocabulary. When selecting descriptors, domain and KDT experts nevertheless have to carefully consider any constraints on the feature space dimensionality imposed by the clustering algorithm to be employed during pattern discovery. A text unit vector is encapsulated by the ADT `TextUnitVector` (see Appendix B.25 on page 231).

**Descriptor Weighting** Typically, the descriptors of a domain-specific controlled vocabulary are not equally useful for characterizing the content of text units (cf. Baeza-Yates and Ribeiro-Neto, 1999, p. 24). When weighting the occurrences of descriptors in a text unit, more important descriptors should therefore be emphasized by assigning them a higher weight than less useful descriptors. The systematic procedure of assigning appropriate numerical weights to descriptor occurrences in text units is commonly referred to as a weighting scheme. Setting up a weighting scheme is neither a trivial issue nor a context-free decision. To select a reasonable and framework-specific weighting scheme for text unit descriptors, the objectives of our knowledge discovery process and characteristics of typical input text archives have to be considered in particular.

In a seminal work summarizing 20 years of experimental evidence, Salton and Buckley (1988) concluded that information retrieval performance crucially depends on the weighting scheme for content identifiers that represent documents and queries. Given an information request expressed as a query, IR performance is improved by retrieving a larger proportion of relevant texts (i.e., by increasing recall) and by reducing the number of mistakenly retrieved, irrelevant documents (i.e., by increasing precision). Consequently, term weighting schemes in IR systems should comprise both recall- and precision-enhancing factors. To that end, Salton and Buckley advised using composite weights that consist of a term frequency, a collection frequency, and a normalization component. Aimed at improving recall, the term frequency component increases the weight of content identifiers that appear frequently in individual documents. If highly frequent terms occur in many documents of the collection, however, term frequency by itself is not capable of ensuring an acceptable retrieval precision. Hence, the collection frequency component increases the weight of content identifiers that only occur in a few documents of the respective archive. Thirdly, the normalization component corrects the term frequency bias introduced by widely varying document lengths in the same archive.

Although our framework for semantic XML tagging pursues other objectives than effectively retrieving textual content, the thoughts on term weighting brought forward by Salton and Buckley are nonetheless relevant. Instead of striving for retrieving relevant

documents in response to information requests, we aim at discovering groups of semantically similar text units and assigning them content-descriptive labels. Consequently, the effectiveness of the DIASDEM framework cannot be assessed by standard IR metrics like recall (i.e., ratio of retrieved relevant texts to all relevant texts in a collection) and precision (i.e., ratio of retrieved relevant texts to all retrieved texts). Nevertheless, identifying semantic concepts in a set of text units by means of clustering the associated vectors is facilitated by descriptor weights that exhibit the properties outlined above. In particular, our pattern discovery phase sets out to – literally – recall, or to find, as many concepts at the text unit level as possible, while simultaneously ensuring the desired level of concept – literally – precision, or specificity. Our framework thus necessitates a descriptor weighting scheme that effectively supports a clear separation between text units that represent distinct semantic concepts. This implies that the best descriptors occur frequently in a text unit, but have a low collection frequency. Since this descriptor discrimination consideration is shared with term weighting in IR, the recommendations made by Salton and Buckley are applicable in our knowledge discovery context as well.

Salton and Buckley (1988, p. 521) argued that the effectiveness of weighting schemes for document vectors ultimately depends on characteristics of the respective collections. Based on experimental evidence, Salton and Buckley recommended using different term weighting components, depending on characteristic properties of the focal text archive. In the DIASDEM framework, text unit vectors are created by mapping structural text units, which usually comprise far fewer words than entire documents, onto a lower-dimensional feature space spanned by a controlled vocabulary. Concerning the term frequency component in this case, the authors suggested a binary weight equal to 1 for descriptors present in a text unit and equal to 0 for non-present descriptors. Thereby, the exact term frequency is intentionally ignored. Regarding the collection frequency component, Salton and Buckley generally recommended an inverse document frequency weight (cf. Subsection 2.1.2) unless the collection is very dynamic and thus requires regular updates of document frequencies. This exclusion criterion is not met in our context since we target domain-specific documents of rather homogeneous content only. According to Salton and Buckley, the normalization component is negligible if the deviation in vector length is not large. Consequently, we omit normalizing text unit vectors as they represent structural text units that are not assumed to vary significantly in size.

Given controlled vocabulary  $V$ : ControlledVocabulary,  $n \in \mathbb{N}$  pre-processed text units, and  $m = V.\text{numberOfDescriptors}()$  text unit descriptors, the corresponding collection of  $n$  text unit vectors is an  $(n \times m)$  matrix  $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n]^T$  in the vector-space model. Rows of matrix  $\mathbf{U}$  represent  $n$  text unit vectors and columns correspond to  $m$  text unit descriptors. For  $i = 1, 2, \dots, n$  and  $j = 1, 2, \dots, m$ , the following framework-specific weighting scheme for text unit descriptors reflects the considerations stated above:

$$\mathbf{u}_i[j] = \begin{cases} \ln \frac{n}{\text{tuf}(j)} & , \text{ if } \text{tudf}(i, j) > 0 \text{ and } \text{tuf}(j) > 0; \\ 0 & , \text{ otherwise.} \end{cases} \quad (4.1)$$

**Table 4.9:** Text Unit Descriptors of the Controlled Vocabulary  $V_E$  and Weighting Components for Exemplary Text Documents  $\check{t}_{E1}$  through  $\check{t}_{E5}$ 

$j$	$V_E.\text{descriptor}(j)$	$\text{tuf}(j)$	$\ln \frac{n}{\text{tuf}(j)}$	Non-Descriptors Represented by $V_E.\text{descriptor}(j)$
1	"stock"	2	2.53	"common_stock"
2	place	7	1.27	
3	"expect"	2	2.53	
4	"newsdesk"	4	1.83	"newsdesk"
5	"not"	3	2.12	
6	"closing"	2	2.53	"complete"
7	"disclose"	2	2.53	
8	"sale"	11	0.82	"sell"
9	"unit"	12	0.73	"organization", "subsidiary"
10	"term/s:agreement"	2	2.53	
11	"acquisition"	3	2.12	"acquire"
12	"agreement"	3	2.12	"agree", "definitive_agreement"
13	"announcement"	2	2.53	"statement/s:announcement"

**Table 4.10:** Text Unit Vectors of Intermediate Text Unit Layer  $\bar{r}_{E1}$  (cf. Table 4.3 on Page 79 and Table 4.7 on Page 86)

$\mathbf{u}_{E1,1} = [0, 0, 0, 0, 0, 0, 0, 0, 0.82, 0.73, 0, 0, 0, 0]^T$	$\mathbf{u}_{E1,2} = [0, 0, 0, 0, 0, 0, 0, 0, 0.82, 0.73, 0, 0, 2.12, 0]^T$
$\mathbf{u}_{E1,3} = [0, 0, 0, 0, 2.12, 0, 2.53, 0, 0, 2.53, 0, 0, 0]^T$	$\mathbf{u}_{E1,4} = [0, 1.27, 0, 1.83, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$

In Expression 4.1, the term  $\text{tuf}(i, j)$  denotes the frequency of text unit descriptor  $V.\text{descriptor}(j)$  in pre-processed text unit  $i$ . Furthermore,  $\text{tuf}(j)$  denotes the frequency of text units in the collection of size  $n$ , in which  $V.\text{descriptor}(j)$  occurs at least once. Analogous to Salton and Buckley (1988, p. 518),  $\ln \frac{n}{\text{tuf}(j)}$  is the inverse document frequency factor of our weighting scheme, which is referred to as inverse text unit frequency in our framework. Denoted by  $\mathbf{u}_i[j]$ , the  $j$ th component of text unit vector  $i$  represents the weight of text unit descriptor  $V.\text{descriptor}(j)$  in text unit  $i$ .

Unless otherwise stated, we employ the weighting scheme specified by Expression 4.1. If the properties of input text units differ considerably from our assumptions, this weighting scheme should be augmented by an appropriate term frequency and/or normalization component (cf. Salton and Buckley, 1988, p. 521). However, a discussion of alternatively applicable, but less commonly used, weighing schemes (e.g., signal-to-noise ratio or term discrimination value; cf. Korfhage, 1997, pp. 117–122) is beyond the scope of this work.

We illustrate our descriptor weighting scheme by mapping the fully pre-processed text unit vectors, depicted in Table 4.7 on page 86, onto vectors. Given the small controlled vocabulary  $V_E$ : ControlledVocabulary, Table 4.9 lists all  $V_E.\text{numberOfDescriptors}() = 13$  descriptor tokens  $V_E.\text{descriptor}(j)$ , where  $j = 1, 2, \dots, 13$ . For each descriptor, Table 4.9 also includes the text unit frequency  $\text{tuf}(j)$  and the inverse text unit frequency  $\ln \frac{n}{\text{tuf}(j)}$  in

our exemplary archive comprising the texts  $\check{t}_{E1}$  through  $\check{t}_{E5}$ . For instance, the descriptor "agreement" or its associated non-descriptors, like "agree", occur in three out of  $n = 25$  sentences. Its inverse text unit frequency of 2.12 is higher than the inverse text unit frequency of the descriptor "unit" (i.e., 0.73). This token and its associated non-descriptors (e.g., "organization" and "subsidiary") appear in twelve sentences.

As depicted in Table 4.10, the tokenized text unit  $\check{u}_{E1,2} = \langle \bar{e}_{E1,2}.placeholder(), "say", "it", "agree", "to", "sell", "its", "consulting", "and", "technical", "organization", ",,", \bar{e}_{E1,3}.placeholder(), ",,", "to", \bar{e}_{E1,4}.placeholder() \rangle$  is mapped onto the 13-dimensional text unit vector  $\mathbf{u}_{E1,2} = [0, 0, 0, 0, 0, 0, 0, 0, 0.82, 0.73, 0, 0, 2.12, 0]^T$ . Tokenized text unit  $\check{u}_{E1,2}$  contains zero descriptor and three non-descriptor tokens (i.e., "agree", "sell", and "organization"). In accordance with controlled vocabulary  $V_E$ , the vector component  $\mathbf{u}_{E1,2}[8] = 0.82$  denotes the weight associated with descriptor token "sale" whereas  $\mathbf{u}_{E1,2}[9] = 0.73$  is the slightly lower weight of the frequently occurring descriptor "unit". Vector component  $\mathbf{u}_{E1,2}[12] = 2.12$  corresponds to the highly weighted descriptor "agreement". The remaining vector coefficients are equal to 0 as the respective descriptors or their associated non-descriptors do not occur in text unit  $\check{u}_{E1,2}$ .

**Effect on KDT Process Flow** Creating text unit vectors brings the pre-processing phase to an end since the resulting numerical data set is input to the pattern discovery phase of our KDT process. Unlike the pre-processing algorithms executed beforehand, vectors are normally created more than once because text units are iteratively segmented into semantically similar subgroups. In iteration one, all text units are transformed into vectors to initiate the clustering process. In each subsequent iteration, vectors are created only for text units that were assigned to qualitatively unacceptable and thus semantically unlabeled clusters in the immediately preceding clustering iteration. Since the number of text units to be clustered decreases as the iterative pattern discovery process progresses, updated and hence iteration-specific collection frequencies of descriptors have to be computed. Only iteration-specific descriptor weights properly reflect the change in collection frequency statistics induced by our iterative clustering procedure that, step by step, removes text units assigned to high-quality clusters from the input data set.

In addition to the effect of iterative clustering, weighting descriptor occurrences in the knowledge discovery phase is different from the knowledge application phase. More specifically, the iteration-specific collection frequency components of descriptor weights are once computed in the interactive KDT phase and merely used in the application phase. This approach is necessary because inverse text unit frequencies must reflect the importance of descriptor occurrences in the training archive, which may be an intentionally biased document sample. By contrast, the descriptor frequency components of weights, which are binary in our framework, are not iteration-specific because they solely depend on the descriptor frequencies in the corresponding pre-processed text units.

Since mapping text units onto vectors is closely interrelated with the iterative pattern discovery phase, the effect on the KDT process flow is discussed in detail in Subsection 4.3.4. In addition, we elaborate on the subtle distinction between creating text unit

vectors during knowledge discovery and knowledge application in Section 4.5.

## 4.3 Clustering of Text Unit Vectors

As illustrated in Figure 2.1 on page 22, the pattern discovery step follows goal setting, text selection, text pre-processing, and text transformation in the generic process for knowledge discovery in textual databases. To attain the specific goals of our DIASDEM knowledge discovery process, groups of semantically similar text units have to be identified and assigned content-descriptive labels, which ultimately serve as names of semantic XML tags enclosing the respective text units. After pre-processing, text units are represented by vectors, whose components are numerical weights of subject-specific descriptors from a controlled vocabulary. In this section, we introduce our approach to exploratively discovering domain-specific concepts at the text unit level. This task corresponds to finding groups of similar text units in which certain combinations of atomic concepts prevail. Once discovered, clusters of text unit vectors constitute classification knowledge, which is exploited to semantically tag texts during knowledge application.

After giving an overview on text clustering in general, we elaborate on selecting an appropriate clustering algorithm for the pattern discovery task at hand. Subsequently, we discuss the DIASDEM approach to ranking clusters of text unit vectors by decreasing quality and describe our notion of iterative clustering in detail. Although the former task clearly involves interpreting result patterns and is thus actually different from pattern discovery itself, the framework-specific quality assessment procedure is nevertheless discussed in this section. Undoubtedly, the distinction between qualitatively acceptable and unacceptable clusters of text unit vectors is an integral part of iterative clustering. In contrast, conceptually labeling homogeneous text unit groups is a task of the post-processing step of our KDT process and is therefore introduced in the next section.

### 4.3.1 Clustering Textual Data: An Overview

This subsection summarizes relevant aspects of clustering text by introducing the building blocks of unsupervised learning. Firstly, we outline the core idea of clustering textual data. Secondly, the fundamental concepts of text similarity and dissimilarity are discussed before we present a high-level categorization of existent clustering algorithms. Finally, we concisely survey relevant issues in assessing the validity of clustering results.

**The Notion of Clustering** According to Kaufman and Rousseeuw (1990, p. 1), clustering is “the art of finding groups in data.” The aim of cluster analysis is to identify groups, or clusters, of objects in a data set that exhibit the following characteristics: Objects in the same cluster are very similar, but objects assigned to different clusters are as dissimilar as possible. Analogously, Han and Kamber (2006, pp. 383–384) referred to the activity “of grouping a set of physical or abstract objects into classes of similar objects” as clustering.

The authors stressed that many research disciplines (e.g., statistics and machine learning) have contributed towards the improvement of clustering techniques. Han and Kamber emphasized the wide range of possible application areas for cluster analysis that include pattern recognition, biological studies, image processing, and marketing. Clustering textual data is an important technique, which is often applied, for instance, in information retrieval (cf. Rasmussen, 1992; Wu et al., 2004), Web content mining (cf. Chakrabarti, 2003, pp. 79-123), or topic discovery (cf. Subsection 2.2.1).

Jain et al. (1999, p. 264) introduced clustering as “the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters).” This definition places particular emphasis on the distinction between cluster analysis as an unsupervised learning method and classification as supervised learning. Unlike classification, clustering requires neither pre-defined classes nor class-labeled training objects, but instead strives for the explorative discovery of unknown classes in data sets. Nevertheless, the close link between clustering and classification becomes apparent if the former technique is used to discover initial categories in a data set, which are afterwards assigned to new objects by means of classification (Vazirgiannis et al., 2003, p. 20).

Bringing together the components of a clustering task discussed by Jain et al. (1999, pp. 266–268) and the generic clustering process described by Vazirgiannis et al. (2003, pp. 20–21), a typical clustering process consists of the following five steps:

1. *Representing objects* involves choosing the objects to be clustered, as well as selecting, pre-processing existing, and possibly creating new features that appropriately represent objects. This initial step must ensure the proper encoding of information about objects, which is likely to be relevant for the clustering task at hand.
2. *Defining an object proximity measure* refers to selecting a suitable measure to quantify the (dis-) similarity between usually pairs of objects, which are represented by the features chosen beforehand. Since the notion of (dis-) similarity is fundamental in clustering, the proximity measure must be appropriate to the data domain.
3. *Grouping objects* can be performed in a number of different ways. Hence, a suitable algorithm has to be chosen, parameterized, and executed. In particular, knowledge discovery experts take into account a priori expectations about the characteristics of clusters that are likely to occur in the data set. Selecting an algorithm corresponds above all to choosing a proper clustering criterion (e.g., minimization of a cost function) that determines how groups of similar objects are formed in detail.
4. *Concisely describing clusters*, also referred to as data abstraction, is the optional step of creating a simple and compact description of all discovered clusters. For example, identified clusters may be visualized by representative objects to facilitate the interpretation of clustering results by human beings.
5. *Assessing cluster validity* corresponds to validating the resulting clusters by using appropriate criteria and techniques. Various approaches to cluster validity analysis set out to distinguish ‘good’ clustering results from ‘poor’ ones. To that end, objective criteria are applied to determine whether output clusters are meaningful in

the sense that they are unlikely to have occurred by chance or as an artifact of the algorithm.

In the DIASDEM framework for semantic XML tagging, the clustering task is embedded into a knowledge discovery process as the text mining step. Consequently, the first component of the generic clustering task (i.e., representing objects) corresponds to selecting, pre-processing, and transforming text within our KDT process. As part of the actual pattern discovery step, a proximity measure and an algorithm are chosen prior to grouping text unit vectors. As explained above, the validity of resulting text unit vector clusters is assessed in each iteration of our iterative clustering approach.

In the remainder of this subsection, we focus on relevant proximity measures, concisely categorize major clustering algorithms, and briefly survey important cluster validity concepts. In particular, we deliberately restrict the discussion to grouping objects represented by numerical vectors of ratio-scaled features without missing values and the need to normalize the ranges of features (e.g., cf. Jain and Dubes, 1988, p. 13, pp. 19–20, and pp. 23–25). This restriction is motivated by the corresponding characteristics of text unit vectors to be grouped in our knowledge discovery process. They solely comprise ratio-scaled descriptor weights without missing values because a weight of zero indicates a non-present descriptor. The varying ranges of vector components do not have to be normalized since they represent purposefully chosen descriptor weights measured on the same ‘scale’ of importance. On the contrary, any range normalization, which may be required by certain clustering algorithms, has to preserve the relative differences between weights of more and less important descriptors.

**Similarity and Dissimilarity** Algorithmically assigning similar objects to the same cluster and dissimilar objects to distinct clusters necessitates a suitable notion of proximity. Everitt et al. (2001, p. 35) emphasized that identifying clusters in data sets requires knowledge about how ‘close’ two objects are to each other or how ‘far apart’ they are. Thus, most clustering algorithms assume a numerical measure, or index, of object proximity that solely depends on the values of encoded features (Jain et al., 1999, p. 271).

The proximity between pairs of documents can either be determined directly (e.g., by asking human beings to judge the perceived proximity between any two texts) or indirectly by computing the value of a numerical proximity measure for each pair of encoded document features (cf. Everitt et al., 2001, p. 35). Since the former approach is not feasible when processing large-scale text archives, we limit our discussion to the indirect measurement of document proximity. Adopting the information retrieval vector-space model (cf. Subsection 2.1.2 on page 23), let  $\mathbf{t} = [\tau_1, \tau_2, \dots, \tau_m]$  denote<sup>6</sup> an  $m$ -dimensional property vector representing a text document such that  $m \in \mathbb{N}$ ,  $\|\mathbf{t}\| \neq 0$ ,

<sup>6</sup>Notation: Let  $\mathbf{x} = [\xi_1, \xi_2, \dots, \xi_m]^T$  and  $\mathbf{y} = [\eta_1, \eta_2, \dots, \eta_m]^T$  denote two  $m$ -dimensional column vectors, where  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$  and  $m \in \mathbb{N}$ . The scalar product of vectors  $\mathbf{x}$  and  $\mathbf{y}$  is denoted by  $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^m \xi_i \cdot \eta_i$ . The Euclidean norm of vector  $\mathbf{x}$  is denoted by  $\|\mathbf{x}\| = (\sum_{i=1}^m \xi_i^2)^{1/2}$ .

**Table 4.11:** Common Proximity Measures between Two Text Documents Represented by  $m$ -Dimensional Property Vectors  $\mathbf{t}_1$  and  $\mathbf{t}_2$ 

Proximity Measure	Definition
Cosine similarity	$\text{sim}_{\text{Cos}}(\mathbf{t}_1, \mathbf{t}_2) = \mathbf{t}_1 \cdot \mathbf{t}_2 / \ \mathbf{t}_1\  \cdot \ \mathbf{t}_2\  \in [0; 1]$
Euclidean distance	$\text{dist}_{\text{Euc}}(\mathbf{t}_1, \mathbf{t}_2) = \ \mathbf{t}_1 - \mathbf{t}_2\  \geq 0$
Extended Jaccard similarity	$\text{sim}_{\text{Ext.Jac}}(\mathbf{t}_1, \mathbf{t}_2) = \mathbf{t}_1 \cdot \mathbf{t}_2 / (\ \mathbf{t}_1\ ^2 + \ \mathbf{t}_2\ ^2 - \mathbf{t}_1 \cdot \mathbf{t}_2) \in [0; 1]$

and  $\tau_i \geq 0$  for  $i = 1, 2, \dots, m$ . Furthermore, let  $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n]^T$  denote an  $(n \times m)$  matrix, which represents a text collection and comprises  $n \in \mathbb{N}$  property vectors.

A proximity measure is either a similarity or a dissimilarity. The more two objects resemble each other, the larger a similarity index and the smaller a dissimilarity index (Jain and Dubes, 1988, p. 11). For  $i = 1, 2, \dots, n$  and  $j = 1, 2, \dots, n$ , let  $\text{prox}(\mathbf{t}_i, \mathbf{t}_j)$  denote a proximity index between two property vectors in  $\mathbf{T}$ . According to Jain and Dubes (1988, pp. 14-15), proximity indices must satisfy the following three properties:

1. (a) For a dissimilarity, the dissimilarity of any property vector to itself equals zero:  $\text{prox}(\mathbf{t}_i, \mathbf{t}_i) = 0 \quad \forall i$ .  
 (b) For a similarity, the similarity of any property vector to itself is not less than the similarity of any other property vectors:  $\text{prox}(\mathbf{t}_i, \mathbf{t}_i) \geq \max_j \text{prox}(\mathbf{t}_i, \mathbf{t}_j) \quad \forall i, j$ .
2. Proximity indices are symmetric:  $\text{prox}(\mathbf{t}_i, \mathbf{t}_j) = \text{prox}(\mathbf{t}_j, \mathbf{t}_i) \quad \forall i, j$ .
3. Proximities are non-negative numbers:  $\text{prox}(\mathbf{t}_i, \mathbf{t}_j) \geq 0 \quad \forall i, j$ .

Henceforth, we distinguish similarity indices, which are denoted by  $\text{sim}(\mathbf{t}_i, \mathbf{t}_j)$ , from dissimilarity indices denoted by  $\text{dis}(\mathbf{t}_i, \mathbf{t}_j)$ . A dissimilarity measure is termed metric or distance measure, which is denoted by  $\text{dist}(\mathbf{t}_i, \mathbf{t}_j)$ , if it fulfills the triangular inequality  $\text{dist}(\mathbf{t}_i, \mathbf{t}_j) \leq \text{dist}(\mathbf{t}_i, \mathbf{t}_k) + \text{dist}(\mathbf{t}_k, \mathbf{t}_j) \quad \forall i, j, k$ , where  $k = 1, 2, \dots, n$ , and satisfies the condition  $\text{dist}(\mathbf{t}_i, \mathbf{t}_j) = 0 \rightarrow \mathbf{t}_i = \mathbf{t}_j$  (Jain and Dubes, 1988, pp. 14-15). Measuring distances between objects in a geometric sense is often much more intuitive than using a similarity index (Grabmeier and Rudolph, 2002, p. 312). A distance index can be transformed into a similarity index by various methods, such as  $\text{sim}(\mathbf{t}_i, \mathbf{t}_j) := \exp(-\text{dist}(\mathbf{t}_i, \mathbf{t}_j))$ .

Table 4.11 contains a list of three proximity measures for property vectors, which are often discussed in the context of text clustering for knowledge discovery (e.g., cf. Ghosh and Strehl, 2006, pp. 78–79). The cosine similarity, however, constitutes “the classical information retrieval approach to comparing documents” (Weiss et al., 2005, p. 91). Following extensive usage in conjunction with the seminal SMART information retrieval system (Salton, 1968, p. 238), the cosine similarity has become a widely used proximity measure for textual data in both IR (Grossman and Frieder, 2004, p. 18) and knowledge discovery (Feldman and Sanger, 2007, p. 85).  $\text{sim}_{\text{Cos}}(\mathbf{t}_1, \mathbf{t}_2)$  is defined as the cosine of the angle between property vectors  $\mathbf{t}_1$  and  $\mathbf{t}_2$ . As the angle between two identical vectors is  $0^\circ$ , their cosine similarity equals 1. The cosine similarity between two orthogonal vectors

equals 0 since the angle between them is  $90^\circ$ . The cosine similarity takes values in  $[0; 1]$  due to the non-negative property of vector components in the vector-space model.

Han and Kamber (2006, p. 388) noticed that the Euclidean distance is “the most popular distance measure” for objects described by interval-scaled features.  $\text{dist}_{\text{Euc}}(\mathbf{t}_1, \mathbf{t}_2) \geq 0$  is defined as the geometric distance of the property vectors  $\mathbf{t}_1$  and  $\mathbf{t}_2$  in the multi-dimensional vector space. This metric is a special case (i.e.,  $\lambda = 2$ ) of the Minkowski metric  $\text{dist}_{\text{Mink}}(\lambda, \mathbf{t}_1, \mathbf{t}_2) = (\sum_{i=1}^m |\tau_{1,i} - \tau_{2,i}|^\lambda)^{1/\lambda}$ , where  $\lambda \geq 1$ . The Euclidean distance assumes some degree of commensurability between the different features of property vectors, which is typically accomplished by standardization of variables (cf. Hand et al., 2001, pp. 32-36). In the vector-space model, however, this assumption is always satisfied since all descriptor weights are measured on the same ‘scale’ of importance.

The extended Jaccard similarity is termed in accordance with Ghosh and Strehl (2006, p. 79). Salton (1989, p. 319) and Rasmussen (1992, p. 422), for instance, recommended  $\text{sim}_{\text{ExtJac}}(\mathbf{t}_1, \mathbf{t}_2)$  for usage in document clustering because it is simple and outputs normalized similarity values in  $[0; 1]$  when the components of property vectors are non-negative. For binary property vectors, this similarity measure equals the ratio of the number of features assuming the value 1 in both  $\mathbf{t}_1$  and  $\mathbf{t}_2$  to the number of features assuming the value 1 in  $\mathbf{t}_1$  or  $\mathbf{t}_2$ . Thereby, the similarity index is computed without taking the typically large number of features assuming the value 0 in both vectors into consideration.  $\text{sim}_{\text{ExtJac}}(\mathbf{t}_1, \mathbf{t}_2)$  extends this notion of similarity to the vector-space model.

Comparing proximity measures in detail is beyond the scope of this work. Ghosh and Strehl (2006, p. 80) argued that  $\text{dist}_{\text{Euc}}(\mathbf{t}_1, \mathbf{t}_2)$  is translation-invariant and scale-variant whereas  $\text{sim}_{\text{Cos}}(\mathbf{t}_1, \mathbf{t}_2)$  is scale-invariant and translation-variant. By means of geometric analysis (cf. Jones and Furnas, 1987), Ghosh and Strehl showed that  $\text{sim}_{\text{ExtJac}}(\mathbf{t}_1, \mathbf{t}_2)$  has aspects of both other proximity measures. Having conducted experiments on the impact of similarity measures on clustering text, Ghosh and Strehl concluded that the cosine and the extended Jaccard similarity are best in capturing human categorization behavior. In contrast, the Euclidean distance is not appropriate for high-dimensional, sparse data sets. Choosing a similarity measure for a specific application “is not prescribed by any theoretical considerations,” summarized Salton (1989, p. 319), but is rather “left to the user.” This statement apparently holds (cf. Everitt et al., 2001, pp. 52–53).

**Categorization of Clustering Algorithms** Since unsupervised learning is an important technique in a variety of domains, a huge number of clustering algorithms have been proposed in the last decades (e.g., cf. Hartigan, 1975; Jain and Dubes, 1988; Kaufman and Rousseeuw, 1990; Everitt et al., 2001; Wu et al., 2004). To categorize this variety of methods, several taxonomies of clustering approaches were promoted. For instance, Jain et al. (1999, p. 275), Grabmeier and Rudolph (2002, p. 340), Jain et al. (2004, p. 261), and Berkhin (2006) classified the most important clustering algorithms in slightly different ways. Reviewing the entire literature on categorizing clustering techniques is beyond the scope of this work. However, we concisely introduce the following five major categories, which are selected in accordance with Vazirgiannis et al. (2003, pp. 23–42) as well as Han

and Kamber (2006, pp. 398–401):

- *Partitioning algorithms* decompose the entire data set into exactly one set of disjoint clusters such that each object belongs to one cluster (e.g., see Everitt et al., 2001, pp. 90–118). Typically, partitioning methods, such as the commonly used *k*-means algorithm (MacQueen, 1967), create an initial partitioning of objects and subsequently either minimize or maximize an algorithm-specific clustering criterion by iteratively re-assigning objects to clusters. To avoid an exhaustive enumeration of all possible solutions, most partitioning algorithms adopt heuristics.
- *Hierarchical algorithms* create a tree-like decomposition of the data set into several nested sets of clusters (e.g., see Everitt et al., 2001, pp. 55–89). These clustering methods are either agglomerative or divisive. The former take a bottom-up approach by successively merging individual objects or clusters into larger clusters. Divisive, or top-down, methods initially assign all objects to one cluster and successively create smaller clusters by splitting larger ones. Hierarchical algorithms require a stopping criterion if exactly one ‘optimal’ set of clusters is desired.
- *Density-based algorithms* enable the discovery of arbitrarily shaped clusters and the detection of outliers (e.g., see Han and Kamber, 2006, pp. 418–424). Informally, the density of an object’s neighborhood with a given radius is defined as the number of objects contained therein. Most partitioning algorithms are based on the distance between objects and thus work well for finding only spherical clusters. In contrast, density-based algorithms regard clusters as highly dense, contiguous regions of objects. Clusters are separated by regions of lower density comprising only noise.
- *Grid-based algorithms* initially quantize the feature space into a finite number of multi-dimensional cells. This data structure is a spatial index of the typically large and high-dimensional data set, which allows for a coarse, yet efficient, clustering (e.g., see Han and Kamber, 2006, pp. 424–429). Instead of grouping input objects, all clustering operations are performed on the quantized feature space. Thus, the processing time of most grid-based methods is independent of the object number, but rather depends on the smaller number of cells in each feature space dimension.
- *Model-based algorithms* attempt to learn generative mathematical models from the data such that each model represents one particular cluster (Zhong and Ghosh, 2003, pp. 1001–1002). These methods often assume the input data to be generated by a mixture of probability distributions. Typically, the model type is specified a priori whereas the model structure and its parameters have to be determined and estimated, respectively. Han and Kamber (2006, pp. 429–434) outlined a statistical, a machine learning, and a neural network approach to model-based clustering.

Without doubt, many algorithms adopt ideas from several categories (Han and Kamber, 2006, p. 400). Self-Organizing Maps (cf. Kohonen, 2001) are, for example, constructed by partitioning and model-based clustering algorithms (Zhong and Ghosh, 2003, p. 1003). Furthermore, Jain et al. (1999, pp. 274–275) introduced five additional categorization criteria for clustering algorithms that may affect all methods regardless of their belonging

to the major categories. For instance, the hard vs. fuzzy criterion refers to the number of clusters that each input object can be assigned to. Hard, or so-called crisp, methods allocate each object to exactly one cluster while fuzzy algorithms assign degrees of membership in several clusters to objects (cf. Vazirgiannis et al., 2003, p. 24).

In the context of thematically grouping text documents, the distinction between hierarchical and non-hierarchical methods is often considered the primary dichotomy. For instance, Salton (1989, pp. 326–341), Rasmussen (1992, pp. 425–427), as well as Manning and Schütze (1999, pp. 495–528) used this top-level criterion to survey clustering algorithms for textual data. Manning and Schütze emphasized that hierarchical methods are preferable for detailed analysis because a document hierarchy conveys more information than a single document partition. Non-hierarchical algorithms should be favored when processing large collections or if efficiency is a decisive issue. Referring to Steinbach et al. (2000), however, Grossman and Frieder (2004, p. 107) noticed that hierarchical methods do not necessarily outperform other methods with respect to cluster quality.

**Cluster Validity Assessment** Hand et al. (2001, p. 295) stressed that in general the “validity of a clustering is often in the eye of the beholder.” Quantifying cluster validity, the authors continued, “is difficult, if not impossible, since the interpretation of how interesting a clustering is will inevitably be application-dependent and subjective to some degree.” Analogously, Silberschatz and Tuzhilin (1996) emphasized the fundamental role of two subjective measures of interestingness (i.e., unexpectedness and actionability) in assessing knowledge discovery results. Furthermore, He et al. (2004, pp. 112–113) raised awareness of the inherent subjectivity when quantitatively measuring cluster quality.

Although the important human influence on quality assessment is recognized, the terms cluster validation and cluster validity assessment, respectively, denote the process of quantitatively and objectively evaluating the results of cluster analysis (cf. Jain and Dubes, 1988, pp. 143–160). To ensure objectivity, Jain and Dubes based their validation framework on the premise that problems of cluster validity are inherently statistical. More specifically, a clustering is valid if it is statistically unusual in some objectively defined sense. Proposing cluster validity indices is easy whereas selecting appropriate index thresholds to distinguish usual from unusual clusterings is difficult and thus requires a sound statistical approach (i.e., hypothesis testing or Monte Carlo analysis).

Indices of cluster validity measure the adequacy of a clustering in an objectively interpretable way (Jain and Dubes, 1988, pp. 160–165). More specifically, a clustering is adequate if it provides true information about the data set or reflects the intrinsic character of the data. Jain and Dubes as well as Vazirgiannis et al. (cf. 2003, pp. 95–96) distinguished the following three types of cluster validity indices:

- *External indices* evaluate the result against a priori specified, external information (e.g., thematic labels) that reflects the truly existing clusters in the data set.
- *Internal indices* assess the adequacy of a clustering by utilizing only the data set itself and internally derived information, such as the proximity matrix.

**Table 4.12:** Five Relative Cluster Validity Indices

Cluster Validity Index	Range	Objective
Original Dunn index (e.g., cf. Vazirgiannis et al., 2003, pp. 104–105)	$[0; \infty)$	Maximize!
Davies-Bouldin index (e.g., cf. Vazirgiannis et al., 2003, pp. 105–106)	$[0; \infty)$	Minimize!
Average silhouette width (Kaufman and Rousseeuw, 1990, pp. 83–88)	$[-1; 1]$	Maximize!
SDbw index (e.g., cf. Vazirgiannis et al., 2003, pp. 113–115)	$[0; 2]$	Minimize!
Overall cluster quality index (He et al., 2004, pp. 114–116)	$[0; 1]$	Minimize!

- *Relative indices* allow for the comparison between different clusterings of the same data set. In particular, the most likely number of clusters in the data and the optimal parameters for a given algorithm, respectively, can be determined by relative criteria (cf. Milligan and Cooper, 1985; Vazirgiannis et al., 2003, pp. 102–103).

In this work, we focus on relative validity indices for two reasons. Unlike benchmark studies that compare clustering algorithms on the basis of pre-classified document collections (e.g., Steinbach et al., 2000), our DIAsDEM framework does not assume the existence of test archives marked up at the text unit level. In contrast to both external and internal criteria with high computational demands, relative clustering criteria typically require fewer computations and do not involve statistical tests (Vazirgiannis et al., 2003, p. 102). Table 4.12 lists five commonly used relative indices applicable for crisp algorithms, which represent different approaches to relative cluster validity assessment. Elaborating on their intricacies, however, is beyond the scope of this work.

Most relative criteria compare intra-cluster similarity, which is often referred to as cluster compactness or cohesion, and inter-cluster similarity, the so-called cluster separation. Each specific validity index nevertheless takes an individual approach to actually defining a valid clustering. For example, the original Dunn index aims at identifying compact, well-separated, and spherical clusters in a geometrical sense. However, this index is “overly sensitive to noisy clusters” (Bezdek and Pal, 1998, p. 301). To reduce the effect of outliers present in the data, Bezdek and Pal thus proposed several generalizations of this index. The SDbw index, for instance, takes the density of regions between clusters into consideration when assessing cluster separation. Finally, the overall cluster quality index is a weighted, additive combination of cluster compactness and separation.

Determining the number of clusters in a data set is one of the most difficult decisions to be made in cluster analysis (Everitt et al., 2001, p. 179). In this context, several authors successfully applied cluster validity criteria in empirical experiments (e.g., cf. Vesanto and Alhoniemi, 2000; Steinbach et al., 2000; Stein et al., 2003). Vesanto and Alhoniemi (2000, p. 593) concluded that “in practice, though, it is better to use the index values as a guideline rather than absolute truth.” Although Jain and Dubes (1988, pp. 143–201) and Vazirgiannis et al. (2003, pp. 93–121) presented a detailed overview of cluster

validity assessment, the authors (perhaps intentionally) avoided giving any advice on how to select an appropriate validity index. In addition, Estivill-Castro (2002, p. 71) confirmed the application-dependency of cluster quality by stressing that validity “depends on the data set where a claim of existence of structure was made.” Given the large number of available validity indices, a KDT expert apparently has no choice, but to employ several of them when conducting a cluster analysis in a new domain.

### 4.3.2 Selecting a Clustering Algorithm

In Subsection 2.1.1, we have stressed that knowledge discovery strives to find new, previously unknown patterns in data. Facing the vast and continuously growing literature on clustering methods, researchers and practitioners alike have to bear in mind that different algorithms are effective in identifying different kinds of clusters (Hand et al., 2001, pp. 295–296). Stein and Meyer zu Eissen (2003, p. 257), for instance, characterized several algorithms with respect to geometrical properties of the recognized clusters. Since clustering is above all employed to discover unexpected structures in data, Hand et al. simultaneously warned of imposing too many preconceptions on the analysis.

Consequently, selecting appropriate clustering algorithms for usage in the DIASDEM knowledge discovery process is of paramount importance. In this subsection, we initially describe criteria for choosing clustering algorithms suitable for application in our framework. Thereafter, we outline three adequate clustering algorithms.

**Selection Criteria** According to Everitt et al. (2001, p. 179), choosing an appropriate clustering method requires considering issues in algorithm design to ensure an effective discovery of the suspected types of clusters. In addition, suitable clustering algorithms must be “insensitive to error” and “available in software.” Although these suggestions are fairly general and assume a priori knowledge of cluster properties, Everitt et al. (2001, p. 178) provided hints concerning the general applicability of methods depending on the types of attributes occurring in the input data. For example, Self-Organizing Maps (cf. Kohonen, 2001) are considered useful for large data sets of objects described by continuous features. Analogously, Kaufman and Rousseeuw (1990, pp. 50–52) suggested choosing algorithms primarily based on the measurement scale of object features.

Although most clustering applications pose individual requirements, Han and Kamber (2006, pp. 385–386) identified nine typical ones. These generic requirements stand as potential selection criteria for choosing clustering methods in our knowledge discovery process. Two of them, the ability to deal with different types of attributes and constraint-based clustering, are not relevant in our framework. Text unit vectors have real-valued components only (see Definition 28 on page 94), and there are no additional application-specific constraints that need to be satisfied. The ability to find clusters of arbitrary shape is possibly relevant, but properties of clusters to be discovered are often likely to be unknown in advance. Minimizing the domain knowledge necessary to determine input parameters is only partially relevant as we explicitly incorporate domain knowledge any-

way (cf. Subsection 4.2.4). The effective handling of high-dimensional data sets is another partially relevant requirement. Despite employing a controlled vocabulary when mapping text units onto vectors (cf. Subsection 4.2.5), the resulting, typically 100- through 300-dimensional vectors are neither low- nor particularly high-dimensional. In our context, it is finally desirable, rather than strictly required, that clustering algorithms are insensitive to the order of input text unit vectors.

Three generic requirements for clustering algorithms proposed by Han and Kamber (2006, pp. 385–386) are particularly relevant in our KDT process. Firstly, the method has to be highly scalable with respect to the number of text unit vectors because training archives may be very large. Secondly, the ability to deal with noisy data is relevant since we cannot assume real-world archives to comprise solely text units that exhibit a clustering tendency. Thirdly, the clustering results (i.e., groups of text unit vectors) must be interpretable, comprehensible, and usable to distinguish qualitatively acceptable from unacceptable clusters (cf. Section 3.3).

If specific requirements are addressed at all, authors presenting the application of clustering techniques in information retrieval or knowledge discovery in textual databases mostly focus on requirements similar to those outlined by Han and Kamber (2006, pp. 385–386). For example, Zamir et al. (1997) discussed two domain-specific requirements in addition to constraints concerning the scalability of the clustering algorithm. Wen et al. (2001) searched for a clustering algorithm satisfying one application-specific requirement as well as two generic ones: scalability and minimal needs of domain knowledge to determine input parameters of the algorithm. Furthermore, Hotho et al. (2003a, pp. 138–139) elaborated on requirements concerning the high-dimensionality of textual data sets as well as the interpretability and subjectivity of clustering results.

Clustering algorithms to be employed within the DIASDEM framework should comply with the following six main requirements, which are ordered by decreasing importance:

1. *Ability to deal with real-valued features*: Our notion of text unit vectors (see Definition 28 on page 94) entails this fundamental, compulsory requirement.
2. *Ability to execute in application mode*: The second compulsory requirement is imposed by the two-phase DIASDEM framework (cf. Chapter 3). A domain-specific KDT process flow is interactively parameterized in the knowledge discovery phase. Subsequently, this process flow is automatically executed in the knowledge application phase. More specifically, new text documents are automatically tagged in the second phase by utilizing classification knowledge acquired in phase one. Therefore, adequate clustering methods must provide an application mode in which new text unit vectors are assigned to previously discovered clusters (e.g., centroid-based classification; cf. Weiss et al., 2005, pp. 113–114).
3. *Interpretability and usability*: Basically, algorithms must be available in software to be usable. Our notion of interpretability corresponds to distinguishing qualitatively acceptable clusters of text unit vectors from unacceptable ones. Concretely, clustering results must be output in a form that enables the application of our

DIASDEM-specific cluster quality criteria. To meet this compulsory requirement, text unit vectors may, for instance, be assigned the cluster identifiers.

4. *Ability to deal with noisy data:* Although we assume a clustering tendency at the text unit level in training and application archives (cf. Section 3.2), it is not reasonable to assume noise-free data sets. In our context, noise corresponds to text unit vectors that represent text units featuring rarely occurring subjects, topics, or themes. Since noisy text units are not semantically marked up at best, the clustering algorithm should be capable of treating their respective vectors accordingly.
5. *Scalability:* Furthermore, adequate algorithms should be highly scalable to efficiently process training archives comprising a large number of text units. Thereby, the need to perform cluster analysis based on random, often relatively small samples of text unit vectors should be eliminated to the greatest extent possible.
6. *Proven track of KDT applications:* Finally, our less important requirements shall be covered by this proxy, but neither necessary nor sufficient, criterion. This includes the ability to find arbitrarily shaped clusters, the minimization of required knowledge to determine parameters, as well as challenges related to the dimensionality and sparsity of input data. Hence, algorithms are assumed to be appropriate for our framework if they were successfully used for clustering textual data.

Having established six criteria for the framework-specific selection of adequate clustering methods, the “user’s dilemma” (Dubes and Jain, 1976, p. 247) of finally choosing a specific algorithm still remains to be solved. Unfortunately, it is highly likely that several existing clustering algorithms fulfill our requirements. In a similar context, Dubes and Jain (1976, p. 247) stressed that searching for a single ‘best’ clustering technique “would be fruitless and contrary to the very nature of clustering.” Due to the exploratory, rather than confirmatory, character of cluster analysis (cf. Estivill-Castro, 2002, p. 71), this somewhat pragmatic argument is widely supported. For example, Kaufman and Rousseeuw (1990, p. 37), Rasmussen (1992, p. 436), and Everitt et al. (2001, p. 177) emphasized that it is usually advisable to employ multiple clustering algorithms in combination with cluster validation methods capable of checking the reliability of the results.

In the remainder of this subsection, we characterize three clustering techniques that satisfy most requirements for usage in the DIASDEM framework: the bisecting  $k$ -means algorithm (Steinbach et al., 2000), one algorithm based on the Self-Organizing Map paradigm (i.e., BATCH MAP; Kohonen, 2001, pp. 139–140), and a method taking a shared nearest neighbor approach (i.e., SNN Clustering; Ertöz et al., 2004). Our selection is of course debatable, but it reflects three distinct approaches to clustering textual data. Table 4.13 provides a qualitative summary of these three algorithms by focusing on the fulfillment of our six selection criteria.

**Bisecting  $k$ -Means** The bisecting  $k$ -means algorithm (Steinbach et al., 2000) extends the  $k$ -means algorithm (MacQueen, 1967) or, more precisely, the large family of related methods (cf. Everitt et al., 2001, p. 100). The standard  $k$ -means algorithm is a partitioning

**Table 4.13:** Summary of Three Proposed Clustering Algorithms w.r.t. the Fulfillment of DIAsDEM-Specific Selection Criteria

Requirement	Bisecting $k$ -Means	BATCH MAP	SNN Clustering
Ability to deal with real-valued features	yes	yes	yes
Ability to execute in application mode	yes	yes	yes
Interpretability and usability	yes	yes	yes
Ability to deal with noisy data	no	no	yes
Scalability	yes	yes	no
Proven track of KDT applications	yes	yes	yes

clustering method, which groups the input data set into an a priori specified number of clusters (i.e., the clustering). Each cluster is represented by one centroid defined as the mean vector of all vectors assigned to the respective cluster. Given a distance measure, the desired number of clusters  $k \in \mathbb{N}$ , and a data set comprising  $n$  vectors  $\mathbf{x}_i \in \mathbb{R}^m$  ( $n, m \in \mathbb{N}$ ,  $n > k$ , and  $i = 1, 2, \dots, n$ ), the standard  $k$ -means algorithm is defined as follows (cf. Han and Kamber, 2006, p. 403):

1. Select  $k$  arbitrary (e.g., randomly chosen) vectors as the initial centroids.
2. For each vector, find the nearest centroid and assign the vector to its cluster.
3. For each cluster, compute the centroid.
4. Unless the assignment of vectors to clusters does not change, go to step 2.

Steinbach et al. (2000) as well as Han and Kamber (2006, p. 403) referred to the above method as the  $k$ -means algorithm. Strictly speaking, however, it is the so-called Forgy’s algorithm (see Forgy, 1965; Gose et al., 1996, pp. 211-213). MacQueen without a doubt coined the term  $k$ -means, but simultaneously referred to Forgy (1965) as an iterative, “two-step improvement procedure” (MacQueen, 1967, p. 294). Unlike Forgy’s algorithm, the original  $k$ -means method incrementally recomputes the centroids and performs only two passes through the data set (see MacQueen, 1967; Gose et al., 1996, pp. 213–215).

As the standard  $k$ -means minimizes the squared distances between vectors and their centroids, it is only capable of discovering convex clusters of approximately equal size (Han and Kamber, 2006, pp. 403–404). When heuristically minimizing the total sum of squared distances,  $k$ -means often converges to a local, instead of a global, minimum. The validity of the resulting clustering thus depends on the initial centroids (Berkhin, 2006, p. 41). To alleviate this negative effect,  $k$ -means is typically executed multiple times with different initializations. The final clustering is chosen based on an appropriate cluster validity criterion. Moreover, the standard  $k$ -means is very sensitive to noise, but it can be augmented to detect outliers (e.g., cf. Larsen and Aone, 1999, p. 18).

The bisecting  $k$ -means algorithm requires the same input as the standard  $k$ -means and encompasses the following steps (cf. Steinbach et al., 2000; Savaresi et al., 2002):

1. The initial clustering consists of one cluster that represents the entire data set.
2. Selection step: Arbitrarily choose one cluster, for instance the largest, to be split.
3. Bisecting step: Execute the standard  $k$ -means algorithm to find two sub-clusters of the selected cluster and remove the selected cluster from the clustering.
4. Unless the clustering consists of  $k$  clusters, go to step 2.

Clearly, the bisecting  $k$ -means method combines elements of both partitioning (i.e., the bisecting step) and divisive hierarchical clustering (i.e., the selection step). This algorithm shares many characteristics of the standard  $k$ -means: discovery of convex clusters, sensitivity to noise, and the importance of initial centroids. Consequently, Steinbach et al. (2000) suggested executing the bisecting step multiple times and selecting the ‘best’ split based on cluster validity. According to Steinbach et al., the bisecting  $k$ -means has a time complexity that is linear in the number of input vectors and can therefore be considered scalable. In addition, this algorithm is executable in application mode using a centroid-based classifier (cf. Weiss et al., 2005, pp. 113–114). After performing exactly the same data pre-processing steps as in the knowledge discovery phase, a new vector is assigned to the cluster that is represented by the nearest centroid. Finally, the generated mapping of input vectors onto cluster identifiers can be easily interpreted and used to compute our DIASDEM cluster quality criteria.

Having conducted a large-scale experiment in comparing document clustering techniques, Steinbach et al. (2000) concluded that the bisecting  $k$ -means algorithm outperforms the standard  $k$ -means. More surprisingly, this technique performed as good as, or even better than, several agglomerative hierarchical clustering methods. In addition, Dhillon and Modha (2001), Zhao and Karypis (2006), as well as Yoo and Hu (2006), for example, successfully deployed the bisecting  $k$ -means algorithm to cluster textual data.

**Batch Map** Introduced by Kohonen (1993, pp. 1149–1150) as a very fast algorithm, BATCH MAP belongs to the large family of Self-Organizing Map (SOM; cf. Kohonen, 2001) algorithms. Before discussing BATCH MAP, we therefore concisely outline relevant properties of the basic SOM algorithm (cf. Kohonen, 2001, pp. 105–115).

Conceived by Teuvo Kohonen in 1982, the original SOM algorithm maps a high-dimensional input space onto a low-dimensional output space, which is usually a two-dimensional grid referred to as the map. In particular, the “SOM thereby compresses information while preserving the most important topological and/or metric relationships” of the input vectors (Kohonen, 2001, p. 106). In the two-dimensional case, the resulting map consists of units that are arranged as an ordered, mostly rectangular or hexagonal array. The topological neighborhood of a map unit encompasses, for example, all units in the array that are positioned within a certain radius around the focal unit. Given  $n \in \mathbb{N}$  input vectors  $\mathbf{x}_i \in \mathbb{R}^m$ , where  $i = 1, 2, \dots, n$  and  $m \in \mathbb{N}$ , each of  $k \in \mathbb{N}$  units is associated with one reference vector  $\mathbf{m}_j \in \mathbb{R}^m$ , where  $j = 1, 2, \dots, k$ . Each reference vector, referred to as a generalized median by Kohonen, fully represents input vectors assigned to its unit and partially represents input vectors that are assigned to other units in the topological

neighborhood of its unit. Hence, the SOM algorithm is effective in both visualizing and clustering high-dimensional data based on topological similarity.

Elaborating on the intricacies of training a Self-Organizing Map is beyond the scope of this work. It is noteworthy, however, that the SOM is a competitive-learning neural-network algorithm (Kohonen, 1993, p. 1147). During the learning phase, all topologically connected reference vectors are initialized and thereafter incrementally adjusted such that the grid “adaptively assumes a form by which it best describes the input vectors in an ordered, structured fashion.” The entire learning process is highly parameterizable to adequately define, for instance, the shape and size of the topological neighborhood.

Given a proximity measure, the properties of the desired result map (i.e., its form and the number of map units  $k \in \mathbb{N}$ ), a definition of the topological neighborhood, and a data set comprising  $n$  vectors  $\mathbf{x}_i \in \mathbb{R}^m$  as specified above, the BATCH MAP algorithm is defined as follows (cf. Kohonen, 1993, p. 1150):

1. Select  $k$  arbitrary (e.g., randomly chosen) vectors as initial reference vectors.
2. For each map unit  $j = 1, 2, \dots, k$ , collect a list of input vectors whose nearest reference vector belongs to the topological neighborhood of map unit  $j$ .
3. For each map unit  $j$ , the new reference vector  $\mathbf{m}_j$  is the mean vector of all input vectors assigned to the respective list of map unit  $j$  created in step 2.
4. Go to step 2 “a few times.”

Kohonen (1993, p. 1150) emphasized that BATCH MAP resembles the standard  $k$ -means algorithm outlined above. Unlike the standard  $k$ -means, however, it imposes a topological order onto the final clustering. In particular, Kohonen suggested starting with a larger topological neighborhood and gradually decreasing its size during the iterative learning process. In the last iterations, the neighborhood of map unit  $j$  may even solely contain the unit  $j$  to ensure stable reference vectors. In this case, the final iterations are equivalent to the standard  $k$ -means algorithm.

The BATCH MAP algorithm is able to deal with real-valued features and capable of executing in application mode. Similar to the bisecting  $k$ -means and its centroid-based classifier described above, BATCH MAP allows for utilizing a classifier based on the nearest reference vector of new input vectors. Analogous to the bisecting  $k$ -means, the results output by BATCH MAP can easily be processed to apply the DIAsDEM cluster quality criteria. Due to the topological ordering of map units in a two-dimensional array, the interpretability is undoubtedly improved because the graphical result map facilitates deeper insights into the discovered classification knowledge. Like the bisecting  $k$ -means, BATCH MAP is also sensitive to noisy input vectors per se, but this algorithm can be extended to deal with outliers (e.g., cf. Larsen and Aone, 1999, p. 18).

According to Vesanto and Alhoniemi (2000, p. 589), the standard SOM scales linearly with the number of input vectors and quadratically with the number of map units. Hence, training huge maps can be very time consuming. However, Kohonen (2001, p. 312) noticed that BATCH MAP “has the advantage of being approximately an order of magnitude faster than the standard SOM.” Since we do not strive to train especially huge maps, BATCH

MAP is thus considered scalable for our KDT process. In addition, the standard SOM algorithm and BATCH MAP were frequently employed to cluster textual data. Using both algorithms, Kohonen (2001, pp. 296–299) processed 6,840,568 English patent abstracts to construct a map comprising 1,002,240 units. Schweighofer et al. (2001), Rauber and Merkl (2003), Chen et al. (2003), and Lagus et al. (2004) presented other relevant applications of the Self-Organizing Map.

**SNN Clustering** Ertöz et al. (2003, 2004) proposed a novel clustering technique capable of finding clusters of different sizes, shapes, and densities in high-dimensional, noisy data sets. Since this method is based on the shared nearest neighbor clustering algorithm presented by Jarvis and Patrick (1973), we briefly motivate the concept of shared nearest neighbors and outline the fundamental algorithm. Thereafter, we introduce the new approach to shared nearest neighbor, or abbreviated SNN, clustering.

Ertöz et al. (2003) argued that the commonly used proximity measures introduced in Subsection 4.3.1 do not work well in high-dimensional, sparse data sets representing, for example, text documents. On average, the similarity between any two vectors in these kinds of data sets is typically low. Beyer et al. (1999) explored the effect of dimensionality on the nearest neighbor problem. As the dimensionality increases, Beyer et al. showed that the distance from any vector to its nearest vector approaches the distance to its farthest vector under a broad set of conditions. This effect may occur for as few as 10–15 dimensions. Using a well-known text classification archive and the cosine similarity, Ertöz et al. (2004, p. 88) demonstrated that “a document’s closest neighbor actually belongs to a different class 20% of the time.” To alleviate the problems associated with direct proximity computations, Ertöz et al. (2003) advocated the concept of shared nearest neighbors. More specifically, the similarity between any two vectors is defined on the basis of nearest neighbors shared by them. If two vectors have many common nearest neighbors, the similarity between them is ‘confirmed’ and thus more ‘reliable’ than directly computed proximity indices.

Jarvis and Patrick (1973, pp. 1026–1027) proposed a shared nearest neighbor approach to discovering non-globular clusters. Given a proximity measure, the size of the nearest neighbor list  $k \in \mathbb{N}$ , a similarity threshold  $k_t \in \mathbb{N}, k_t \leq k$ , and a data set comprising  $n$  vectors  $\mathbf{x}_i \in \mathbb{R}^m$  ( $n, m \in \mathbb{N}, n > k$ , and  $i = 1, 2, \dots, n$ ), this clustering algorithm starts by determining the  $k$  nearest neighbors for each vector. Subsequently, two vectors are assigned to the same cluster if (i) they are contained in each other’s list of  $k$  nearest neighbors and (ii) have at least  $k_t$  nearest neighbors in common. The first condition avoids combining a small, relatively isolated group of vectors with a highly dense cluster.

To overcome a few limitations (see Ertöz et al., 2004, p. 89), the authors extended the Jarvis-Patrick method and proposed a novel SNN clustering algorithm. This technique utilizes the shared nearest neighbor approach to effectively process high-dimensional data sets and additionally uses representative vectors, so-called topic points, to find clusters of differing sizes and shapes. Given a proximity measure, the data set specified above, the size of the nearest neighbor list  $k \in \mathbb{N}$ , and five parameters described below, the SNN

clustering algorithm is defined as follows (Ertöz et al., 2004, pp. 89–90):

1. Determine the list of  $k$  nearest neighbors for each vector  $i = 1, 2, \dots, n$ . Each document is considered to be its  $0^{th}$  nearest neighbor.
2. Construct the shared nearest neighbor graph whose vertices represent vectors. This graph has an edge between two vertices if the corresponding vectors have each other in their nearest neighbor list. Each edge is annotated with the so-called link strength defined as the number of nearest neighbors shared by the two vectors, which correspond to the connected vertices. An edge is labeled ‘strong link’ if its link strength is greater than a user-supplied strong link threshold  $t_{\text{StrongLink}} \in \mathbb{N}$ , where  $t_{\text{StrongLink}} < k + 1$ .
3. For each vector  $i$ , compute the connectivity  $\text{conn}(i)$  that is defined as the number of strong links connected with the vertice of vector  $i$  in the SNN graph.
4. Discard each vector  $i$  as noise assigned to a specific outlier cluster if  $\text{conn}(i) < t_{\text{Noise}}$ , where  $t_{\text{Noise}} \in \mathbb{N} < n$  is a user-specified threshold. Outliers are not considered in the clustering because they are similar to only a few of their neighbors.
5. Label each vector  $i$  as a representative vector if  $\text{conn}(i) > t_{\text{Topic}}$ , where  $t_{\text{Topic}} \in \mathbb{N}$  is a parameter such that  $t_{\text{Noise}} < t_{\text{Topic}} < n$ . Representative vectors, or topic points, are similar to most of their neighbors and thus represent the neighborhood.
6. Assign any two vectors to the same cluster if (i) at least one vector is a representative vector and (ii) they share a significant number of nearest neighbors. The second condition requires that the link strength between the vertices that correspond to the respective vectors in the SNN graph is greater than the user-supplied threshold  $t_{\text{Merge}} \in \mathbb{N}$ , where  $t_{\text{StrongLink}} < t_{\text{Merge}} < k + 1$ .
7. Finally, consider non-noise vectors not assigned to any cluster yet: Assign these vectors to the respective clusters if their associated vertices in the SNN graph are connected to vertices of vectors assigned to a cluster with a link strength greater than the user-supplied threshold  $t_{\text{Labeling}} \in \mathbb{N}$ , where  $t_{\text{StrongLink}} < t_{\text{Labeling}} < t_{\text{Merge}}$ .

According to Ertöz et al. (2004, p. 90), this clustering algorithm is capable of discovering “communities of documents, where a document in a community shares a certain fraction of its neighbors with at least some number of neighbors.” The clustering method is based on the idea that the probability of a document belonging to a different thematic class than its nearest neighbor decreases as these two documents share an increasing number of common neighbors. Ultimately, Ertöz et al. designed the new SNN clustering technique to find topics occurring in text archives. Since the authors successfully conducted experiments on real-world text collections, SNN clustering has a proven track of KDT applications. Nevertheless, selecting appropriate parameters requires considerable expertise (e.g., the size of nearest neighbor list; see Ertöz et al., 2004, pp. 90–91).

This method is able to process real-valued vectors and outputs results that are easily interpretable and usable within the DIAsDEM framework. Unlike the other two algorithms, SNN clustering is capable of dealing with noisy data. In contrast to the methods introduced previously, however, SNN clustering cannot be considered highly scalable with

respect to the size of the training data set. The run-time complexity of this clustering algorithm scales quadratically with the number of input vectors because the similarity matrix has to be computed to determine the nearest neighbor list for each vector (cf. Ertöz et al., 2003). However, Ertöz et al. discussed several optimizations to process large data sets efficiently. Finally, SNN clustering can be executed in application mode by employing a  $k$ -nearest-neighbor classifier (cf. Weiss et al., 2005, pp. 88–89).

### 4.3.3 Ranking Clusters of Text Unit Vectors

Separating qualitatively acceptable clusters of text unit vectors from unacceptable ones is a core step in our knowledge discovery process (cf. Section 3.3). When retrieving information from an archive, ranking refers to sorting the result documents by decreasing relevancy to present the most relevant texts on top of the often large result list (e.g., cf. Frakes and Baeza-Yates, 1992, pp. 363–392). In the DIASDEM framework, however, ranking denotes the process of sorting clusters of text unit vectors by decreasing quality. Before we elaborate on our concept of iterative clustering, we therefore motivate and introduce the fundamental DIASDEM cluster quality criteria in this subsection.

**Our Notion of Cluster Quality** As discussed in Subsection 4.3.1, there exists a large number of different relative cluster validity indices designed to assess the validity of a clustering, which is the entirety of clusters output by an algorithm. Some of them, for example the average silhouette width (Kaufman and Rousseeuw, 1990, pp. 83–88), are also capable of evaluating the validity of individual clusters. Nevertheless, we intentionally propose a novel, framework-specific set of cluster quality criteria for two purposes:

- Above all, the objectives of our DIASDEM framework (cf. Section 3.2) impose conditions on the quality of text unit clusters. By employing a specific knowledge discovery approach, we strive to discover clusters that (i) represent semantic concepts frequently occurring at the text unit level and that (ii) later serve as names of semantic XML tags. Our main goal entails specific characteristics of qualitatively acceptable, or valid, clusters. These framework-specific properties, which are described in detail below, cannot be guaranteed by generic cluster validity indices.
- In addition, the pattern discovery step of our KDT process adopts a plug-in approach and thus allows for the execution of various clustering algorithms if they fulfill our selection criteria. As outlined in Subsection 4.3.1, however, the effectivity of cluster validity assessment strongly depends on the appropriateness of assumptions underlying the applied criteria (e.g., convex clusters of similar sizes). Hence, choosing a generic validity index is hardly appropriate for separating acceptable from unacceptable clusters of text unit vectors in an algorithm-independent way.

To linguistically distinguish our framework-specific criteria from generic validity indices, we intentionally use the term cluster quality criteria instead of cluster validity criteria. All

**Table 4.14:** Sentences Assigned to Qualitatively Acceptable Text Unit Cluster 4 in Iteration 1 (cf. News Items in Table 4.1 on Page 72)

Sentence of Reuters News Item	Item	Text	Sentence
USA: VASCO says to sell consulting unit.	16106	$\check{t}_{E1}$	1
USA: Weyerhaeuser may sell subsidiary.	71177	$\check{t}_{E3}$	1
Timber giant Weyerhaeuser Company said Monday it may sell its mortgage loan subsidiary, the Weyerhaeuser Mortgage Company.	71177	$\check{t}_{E3}$	2
USA: Miller to sell unit to Modtech.	78899	$\check{t}_{E4}$	1
USA: Helmerich to sell unit to Occidental.	17109	$\check{t}_{E2}$	2
USA: Zurn sells unit to Constellation Capitol.	115995	$\check{t}_{E5}$	1
Zurn Industries Inc said on Tuesday that it will sell its Zurn Mechanical Power Transmission Group to Constellation Capital Partners LLC, for an undisclosed amount.	115995	$\check{t}_{E5}$	2

automatically created suggestions concerning the quality of each cluster (i.e., acceptable or unacceptable), which can later be approved or rejected by the domain expert, solely depend on the DIASDEM cluster quality criteria. Nevertheless, we explicitly encourage using appropriate relative cluster validity indices to find the optimal parameters of the respective clustering method (e.g., number of clusters). This approach corresponds to the ‘best practice’ of employing several relative validity indices when fine-tuning the parameters of clustering algorithms (cf. Subsection 4.3.1).

What exactly characterizes a ‘good’ and thus qualitatively acceptable cluster of text unit vectors? To achieve the objectives of our DIASDEM framework, qualitatively acceptable text unit clusters exhibit the following two characteristics:

1. An acceptable cluster represents only one coherent semantic concept. The corresponding text units are assumed to feature a relatively homogeneous topic if their content can be described by clearly predominating text unit descriptors that occur in almost all text units assigned to the respective cluster.
2. An acceptable cluster comprises a sufficient, relatively large number of text unit vectors. This requirement facilitates the discovery of frequently occurring and thus more important semantic concepts at the text unit level. However, the concretely required minimum cluster size depends on both the domain and the archive.

Since ‘good’ clusters are semi-automatically assigned content-descriptive labels in the post-processing phase, the desired properties of qualitatively acceptable clusters are directly related to the purpose of their semantic labels. Cluster labels eventually serve as names of semantic XML tags and elements of the domain-specific, concept-based XML document type definition, respectively. As specified in Definition 1 on page 7, names of

**Table 4.15:** Sentences Assigned to Qualitatively Unacceptable, Inhomogeneous Text Unit Cluster 7 in Iteration 1 (cf. News Items in Table 4.1 on Page 72)

Sentence of Reuters News Item	Item	Text	Sentence
Terms were not disclosed.	16106	$\check{t}_{E1}$	3
– Chicago newsdesk 312 408-8787	17109	$\check{t}_{E2}$	5
Terms were not disclosed.	78899	$\check{t}_{E4}$	4
Chicago Newsdesk 312-408-8787	78899	$\check{t}_{E4}$	5
–Chicago Newsdesk 312-408-8787	16106	$\check{t}_{E1}$	4
– New York Newsdesk +1 212 859 1610	115995	$\check{t}_{E5}$	5

**Table 4.16:** Sentence Assigned to Qualitatively Unacceptable, Small Text Unit Cluster 0 in Iteration 1 (cf. News Items in Table 4.1 on Page 72)

Sentence of Reuters News Item	Item	Text	Sentence
Final settlement is scheduled for October 21, Miller said.	78899	$\check{t}_{E4}$	3

semantic XML tags convey the meaning of marked-up content by concisely describing concepts that domain experts typically associate with annotated text units. The first condition contributes towards semantically correct markup because only clusters representing homogeneous text units with clearly predominating descriptors are considered acceptable. The second condition concerning the size of acceptable clusters ensures the discovery of frequently recurring thematic concepts at the text unit level, which is one goal of our framework (cf. Section 1.4). Ignoring infrequent concepts contributes towards the effective usage of semantic markup in the form of concept-based XML DTD elements.

Prior to formalizing our notion of cluster quality, we illustrate it using the exemplary text archive comprising five news stories  $\check{t}_{E1}$  through  $\check{t}_{E5}$ , which altogether consists of 25 sentences (cf. Table 4.1 on page 72). For illustrative purposes, we refrain from discussing pre-processing issues (see Section 4.2), but emphasize that sentences correspond to text units. Executing the bisecting  $k$ -means algorithm to discover  $k = 8$  sentence clusters in clustering iteration 1 returned, among others, three clusters listed in Tables 4.14 through 4.16. Visualized in Table 4.14, cluster 4 is qualitatively acceptable because it represents one semantic concept at the sentence level (i.e., the theme ‘sale of business unit’) and it is relatively large (i.e., seven of 25 sentences). Although containing six sentences, cluster 7 shown in Table 4.15 is not acceptable since it lacks one coherent semantic concept. Two of six sentences feature a different subject than the remaining ones. Since cluster 0, which is visualized in Table 4.16, comprises merely one sentence, it is too small and cannot be considered qualitatively acceptable as well.

**DIAsDEM Cluster Quality Criteria** Within our knowledge discovery process, conventional clustering algorithms satisfying certain requirements are employed to find clusters of text unit vectors, which are in turn an integral part of intermediate text units (cf. Definition 18 on page 74). The details of our iterative approach to clustering are described in the next subsection. At this point, we nevertheless anticipate that executing a clustering algorithm results in modified iteration identifiers and cluster identifiers of the input intermediate text units. To simplify the subsequent discussion, we define text unit clusters as multi-sets<sup>7</sup> of intermediate text units:

**Definition 29 (Text Unit Cluster)** *Given intermediate text archive  $\bar{a}$ : IntTextArchive and cluster identifier  $i \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ , the multi-set  $C_{t,i} := \{\{\bar{u} \mid \bar{u}: \text{IntTextUnit} := \bar{a}.\text{intTextUnit}(j, k_j) \ \forall j = 1, 2, \dots, \bar{a}.\text{size}() \ \forall k_j = 1, 2, \dots, \bar{a}.\text{intTextUnitLayerSize}(j) \text{ s.t. } \bar{u}.\text{iteration}() = t \text{ and } \bar{u}.\text{cluster}() = i\}\}$  is a text unit cluster comprising all intermediate text units in  $\bar{a}$ , whose text unit vectors are assigned to cluster  $i$  in clustering iteration  $t$ .*

Since archives may comprise duplicate intermediate text units, text unit clusters are defined as multi-sets instead of sets. Text unit clusters may be empty, but each intermediate text unit in the corresponding archive is assigned to one text unit cluster at most. One text unit cluster is encapsulated by the abstract data type TextUnitCluster (see Appendix B.32 on page 236). The entirety of text unit clusters discovered in the same clustering iteration constitute a text unit clustering, whose instances are encapsulated by the ADT TextUnitClustering (see Appendix B.33 on page 238).

**Definition 30 (Text Unit Clustering)** *Given intermediate text archive  $\bar{a}$ : IntTextArchive and maximum cluster identifier  $i_{max} \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ , the set  $C_t := \{C_{t,1}, C_{t,2}, \dots, C_{t,i_{max}}\}$  is a text unit clustering comprising all  $i_{max}$  text unit clusters  $C_{t,i}$ : TextUnitCluster, where  $i = 1, 2, \dots, i_{max}$ , discovered by a clustering algorithm in clustering iteration  $t$ .*

Our quality criteria can only exploit characteristics of text unit clusters that are available in any case because they are independent of the concretely applied algorithm. Thus, we deliberately limit the components of quality criteria and merely consider occurrences of text unit descriptors in clusters and cluster sizes. In particular, our framework-specific notion of descriptor support within a cluster is of great importance.

**Definition 31 (Descriptor Support)** *Let  $C_{t,i}$ : TextUnitCluster denote text unit cluster  $i \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ . Let  $v_D$ : ControlledVocabularyTerm denote a text unit*

<sup>7</sup>Notation: Let the set  $X$  denote an arbitrary domain. A non-empty, finite multi-set  $Y := \{y_1, y_2, \dots, y_i\}$  is a collection comprising  $i \in \mathbb{N}$  unordered, not necessarily distinct elements  $y_1, y_2, \dots, y_i$  such that  $y_j \in X$ , where  $j = 1, 2, \dots, i$ . Analogous to sets, the number of elements in multi-set  $Y$  (i.e., its cardinality) is denoted by  $|Y| \in \mathbb{N}$ , and an empty multi-set is denoted by  $\emptyset$ , such that  $|\emptyset| = 0$ . Unlike sets, however, multi-sets may comprise duplicate elements.

descriptor such that  $v_D.isDescriptor() = true$ . The descriptor support  $C_{t,i}.descriptorSupport(v_D)$  of text unit descriptor  $v_D$  in non-empty text unit cluster  $C_{t,i}$  is defined as the ratio of the number of intermediate text units in  $C_{t,i}$  whose processed text unit contains the token  $v_D.token()$  to the total number of intermediate text units in  $C_{t,i}$ . If text unit cluster  $C_{t,i}$  is empty,  $C_{t,i}.descriptorSupport(v_D)$  is defined to be zero.

A more formal, multi-set-based definition of descriptor support is given in the specification of the ADT `TextUnitCluster` in Appendix B.32. Descriptor support takes values in  $[0; 1]$ . The term descriptor support is inspired by the support of association rules in a data set (cf. Agrawal et al., 1993, p. 208). Unlike the notion of support in association rule discovery, however, the support of text unit descriptors is always cluster-specific. The greater the fraction of intermediate text units in a cluster that comprise a certain descriptor, the higher its support within the respective cluster. A text unit descriptor, whose support is not very large in a cluster, is likely to be less significant for the purpose of semantically describing the respective cluster content, and vice versa.

Considering the exemplary controlled vocabulary listed in Table 4.9 on page 97, text unit descriptor ("unit", 9, true, null), which also represents the two non-descriptors ("organization", 20, false, 9) and ("subsidiary", 23, false, 9), occurs in six of seven text units assigned to the acceptable text unit cluster 4, which is shown in Table 4.14 on page 116, and thus exhibits a high descriptor support of 6/7 therein. The descriptor ("stock", 1, true, null) does not appear in any sentence and thus has the minimum support of 0 in cluster 4. In the qualitatively unacceptable text unit cluster 7, as shown in Table 4.15 on page 117, the descriptor ("unit", 9, true, null) occurs in two of six text units and has thus a moderate support of 1/3 therein.

To assess the quality of text unit clusters for the ultimate purpose of semantic tagging, highly supported descriptors are favored over less supported ones. In addition, noisy descriptor occurrences are neglected to avoid a high influence of outliers onto the assessment of cluster quality. To incorporate these requirements into the quality evaluation, we introduce dominant and rare text unit descriptors defined as follows:

**Definition 32 (Dominant Descriptor)** Let  $C_{t,i}: TextUnitCluster$  denote text unit cluster  $i \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ . Text unit descriptor  $v_D: ControlledVocabularyTerm$ , where  $v_D.isDescriptor() = true$ , is a dominant descriptor in text unit cluster  $C_{t,i}$  if its support  $C_{t,i}.descriptorSupport(v_D)$  in  $C_{t,i}$  is greater than or equals the user-supplied dominant descriptor threshold  $p_{DD} \in [0; 1]$ .

**Definition 33 (Rare Descriptor)** Let  $C_{t,i}: TextUnitCluster$  denote text unit cluster  $i \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ . Text unit descriptor  $v_D: ControlledVocabularyTerm$ , where  $v_D.isDescriptor() = true$ , is a rare descriptor in text unit cluster  $C_{t,i}$  if its support  $C_{t,i}.descriptorSupport(v_D)$  in  $C_{t,i}$  is less than or equals the user-supplied rare descriptor threshold  $p_{RD} \in [0; 1]$ .

The ADT `TextUnitCluster` provides two operations that check whether a given text unit descriptor is dominant and rare, respectively. It must be stressed that choosing

appropriate parameter values for  $p_{DD}$  and  $p_{RD}$  depends on the application domain and on characteristics of the text archive. For typical domain-specific document collections, appropriate settings are  $p_{RD} \in [0.005; 0.020]$  and  $p_{DD} \in [0.80; 0.95]$  such that  $p_{RD} \ll p_{DD}$ . For the exemplary controlled vocabulary listed in Table 4.9 on page 97,  $p_{DD} = 0.80$ , and  $p_{RD} = 0.01$ , the acceptable text unit cluster 4, which is shown in Table 4.14, contains only dominant descriptors, namely, ("sale", 8, true, null) occurring in all sentences of this cluster and ("unit", 9, true, null) exhibiting a high descriptor support of 6/7 therein. Other text unit descriptors do not appear in sentences assigned to this text unit cluster. Our example text archive is too small to illustrate rarely supported descriptors.

Having specified dominant and rare descriptors, we proceed by defining three properties of text unit clusters that are the building blocks of our DIAsDEM quality criteria. As mentioned above, a qualitatively acceptable cluster represents one coherent semantic concept and contains a sufficient, relatively large number of text units. The former requirement is satisfied by putting constraints on both descriptor coverage and descriptor dominance whereas the latter requirement is solely related to the cluster size. These framework-specific characteristics of text unit clusters remain to be introduced:

**Definition 34 (Descriptor Coverage)** Let  $C_{t,i}$ : TextUnitCluster denote text unit cluster  $i \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ , and let  $V$ : ControlledVocabulary denote a controlled vocabulary, where  $V.numberOfDescriptors() > 0$ . Given a user-supplied rare descriptor threshold  $p_{RD} \in [0; 1]$ , the descriptor coverage  $C_{t,i}.descriptorCoverage(V, p_{RD})$  of text unit cluster  $C_{t,i}$  is the ratio of the number of distinct, non-rare descriptors occurring in intermediate text units of  $C_{t,i}$  to the total number of text unit descriptors in  $V$ .

Descriptor coverage measures the proportion of text unit descriptors in the controlled vocabulary that is covered by, or occurs in, text units assigned to a specific cluster. However, only non-rare descriptors that have a descriptor support greater than the specified rare descriptor threshold are taken into account to eliminate a possible bias introduced by outliers. Qualitatively acceptable clusters tend to have a lower descriptor coverage than unacceptable ones because they feature only one semantic theme. Text unit clusters with a very high descriptor coverage are more likely to be less useful ‘garbage clusters’ that comprise text units featuring a variety of different themes.

**Definition 35 (Descriptor Dominance)** Let  $C_{t,i}$ : TextUnitCluster denote text unit cluster  $i \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ , and let  $V$ : ControlledVocabulary denote a controlled vocabulary, where  $V.numberOfDescriptors() > 0$ . Given a user-supplied dominant descriptor threshold  $p_{DD} \in [0; 1]$  and a rare descriptor threshold  $p_{RD} \in [0; 1]$ , the descriptor dominance  $C_{t,i}.descriptorDominance(V, p_{RD}, p_{DD})$  of text unit cluster  $C_{t,i}$  is the ratio of the number of distinct, dominant descriptors occurring in intermediate text units of  $C_{t,i}$  to the number of distinct, non-rare descriptors occurring in intermediate text units of  $C_{t,i}$  if  $C_{t,i}.descriptorCoverage(V, p_{RD}) > 0$ . Otherwise, the descriptor dominance of  $C_{t,i}$  is defined to be zero.

The content of qualitatively acceptable clusters can be described by a few predominating text unit descriptors. To this end, descriptor dominance measures the proportion of dominant descriptors within one cluster. Again, rare descriptors are not considered at all when computing the descriptor dominance of a text unit cluster. The quality of text unit clusters increases as their respective descriptor dominance increases. Since the content of qualitatively acceptable clusters is characterized by clearly dominating descriptors, they tend to have a higher descriptor dominance than unacceptable clusters.

**Definition 36 (Cluster Size)** *Let  $C_{t,i}$ : TextUnitCluster denote text unit cluster  $i \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ . The cluster size  $C_{t,i}.size()$  of text unit cluster  $C_{t,i}$  is the total number of intermediate text units assigned to  $C_{t,i}$ .*

The size of text unit clusters, which is occasionally referred to as cluster cardinality, cannot be neglected because qualitatively acceptable clusters represent frequently occurring, more ‘important’ semantic concepts. Therefore, qualitatively acceptable clusters must be large enough to represent frequently featured subjects of the application domain. Nevertheless, less frequently occurring topics of great importance have to be discovered as well without putting too much emphasis on topical outliers. In general, a greater cluster size is preferred over a smaller one when discovering semantic concepts.

After introducing three fundamental characteristics of text unit clusters separately, we finally bring them together and define our notion of cluster quality:

**Definition 37 (Qualitatively Acceptable Cluster)** *Let  $C_{t,i}$ : TextUnitCluster denote text unit cluster  $i \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ , and let  $V$ : ControlledVocabulary denote a controlled vocabulary, where  $V.numberOfDescriptors() > 0$ . Given the user-supplied*

- *dominant descriptor threshold  $p_{DD} \in [0; 1]$ ,*
- *rare descriptor threshold  $p_{RD} \in [0; 1]$ , where  $p_{RD} \ll p_{DD}$ ,*
- *maximum descriptor coverage  $p_{maxDC} \in [0; 1]$ ,*
- *minimum descriptor dominance  $p_{minDD} \in [0; 1]$ , and*
- *minimum cluster size  $p_{minCS} \in \mathbb{N}$ ,*

*text unit cluster  $C_{t,i}$  is qualitatively acceptable if*

1.  $C_{t,i}.descriptorCoverage(V, p_{RD}) \leq p_{maxDC}$ ,
2.  $C_{t,i}.descriptorDominance(V, p_{RD}, p_{DD}) \geq p_{minDD}$ , and
3.  $C_{t,i}.size() \geq p_{minCS}$ .

Together, the DIASDEM cluster quality criteria 1 and 2 facilitate the discovery of homogeneous clusters that represent only one semantic concept. In addition, DIASDEM cluster quality criterion 3 ensures that acceptable clusters are sufficiently large. By combining three criteria, we put our notion of cluster quality into practice. Besides the dominant and the rare descriptor threshold introduced above, three additional parameter

thresholds have to be carefully chosen by the KDT expert. We recognize that setting appropriate values for maximum descriptor coverage, minimum descriptor dominance, and minimum cluster size requires adequate domain insight. However, the apparent diversity of existing application domains and text archives necessitates our parameter-based definition of cluster quality.

Any text unit cluster that does not meet these three criteria is referred to as a qualitatively unacceptable cluster. The abstract data type `TextUnitCluster` provides an operation that determines whether the instantiated cluster is qualitatively acceptable. For notational convenience, the abstract data type `ClusterQualityCriteria` (see Appendix B.36 on page 241) encapsulates all five DIASDEM cluster quality thresholds introduced in Definition 37. Although Definition 37 enables a clear distinction between acceptable and unacceptable clusters, it does not support the ranking of text unit clusters by decreasing quality. Since a ranking facilitates the cluster inspection by domain experts, we additionally define a real-valued quality index that can be computed for all clusters.

**Definition 38 (Cluster Quality Index)** *Let  $C_{t,i}$ : `TextUnitCluster` denote text unit cluster  $i \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ , which is an element of the text unit clustering  $C_t$ : `TextUnitClustering`, where  $C_t.\text{numberOfIntTextUnits}() > 0$ , and let  $V$ : `ControlledVocabulary` denote a controlled vocabulary. Given a user-supplied dominant descriptor threshold  $p_{DD} \in [0; 1]$  and a rare descriptor threshold  $p_{RD} \in [0; 1]$ , the cluster quality index  $C_{t,i}.\text{qualityIndex}(V, p_{RD}, p_{DD}, C_t)$  of text unit cluster  $C_{t,i}$  is defined as*

$$\begin{aligned} & 1/3 \cdot (1 - C_{t,i}.\text{descriptorCoverage}(V, p_{RD})) + \\ & 1/3 \cdot C_{t,i}.\text{descriptorDominance}(V, p_{RD}, p_{DD}) + \\ & 1/3 \cdot C_{t,i}.\text{size}()/C_t.\text{numberOfIntTextUnits}() \end{aligned}$$

*if  $C_{t,i}.\text{descriptorCoverage}(V, p_{RD}) > 0$ . Otherwise, the cluster quality index is defined as  $1/3 \cdot C_{t,i}.\text{size}()/C_t.\text{numberOfIntTextUnits}()$ .*

The operation `numberOfIntTextUnits()` of ADT `TextUnitClustering` returns the total number of intermediate text units contained in the entire clustering. It is required to compute the relative,  $[0; 1]$ -bound size of a cluster within its text unit clustering. Our cluster quality index takes values in  $[0; 1]$  such that greater values indicate a higher cluster quality, and vice versa. In accordance with our notion of cluster quality, this index is positively correlated with descriptor dominance and relative size of the cluster. Furthermore, it is negatively correlated with descriptor coverage of the respective cluster.

Finding semantically homogeneous clusters that feature exactly one concept can be trivially achieved by assigning each text unit to its own cluster. Conversely, the largest cluster to be discovered comprises all text units. To offset the disadvantages associated with these limit cases, our cluster quality index gives equal weight to the individual effects of all three criteria. Thereby, we intentionally favor homogeneity of clusters, which is measured by descriptor dominance and descriptor coverage, over cluster size. Combining three criteria measured on different scales, however, entails limitations on interpreting

**Table 4.17:** Text Unit Descriptors Occurring in Qualitatively Acceptable Text Unit Cluster 4 in Iteration 1 (cf. Table 4.14 on Page 116)

Text Unit Descriptor	Descriptor Support	Text Unit Descriptor	Descriptor Support
"sale"	7/7 = 1.000	"unit"	6/7 $\approx$ 0.857

**Table 4.18:** Text Unit Descriptors Occurring in Qualitatively Unacceptable, Inhomogeneous Text Unit Cluster 7 in Iteration 1 (cf. Table 4.15 on Page 117)

Text Unit Descriptor	Descriptor Support	Text Unit Descriptor	Descriptor Support
place	4/6 $\approx$ 0.667	"Newsdesk"	4/6 $\approx$ 0.667
"not"	2/6 $\approx$ 0.333	"disclose"	2/6 $\approx$ 0.333
"term/s:agreement"	2/6 $\approx$ 0.333		

concrete index values. Nevertheless, our cluster quality index allows for the comparison of different text unit clusters with respect to their quality.

In Subsection 4.3.1, we have emphasized the importance of relative cluster validity indices for comparing different clusterings of the same data set. To facilitate, for example, searching for the optimal number of text unit clusters in a data set, we finally introduce a relative cluster quality criterion at the text unit clustering level.

**Definition 39 (Clustering Quality Index)** *Let  $C_t$ : TextUnitClustering denote text unit clustering in clustering iteration  $t \in \mathbb{N}$ , where  $C_t.\text{numberOfIntTextUnits}() > 0$  and  $i_{max} \in \mathbb{N}$  is the maximum cluster identifier, and let  $V$ : ControlledVocabulary denote a controlled vocabulary. Given the user-supplied dominant descriptor threshold  $p_{DD} \in [0; 1]$  and rare descriptor threshold  $p_{RD} \in [0; 1]$ , the quality index  $C_t.\text{qualityIndex}(V, p_{RD}, p_{DD})$  of text unit clustering  $C_t$  is defined as  $1/C_t.\text{numberOfIntTextUnits}() \cdot \sum_{j=1}^{i_{max}} C_{t,j}.\text{size}() \cdot C_{t,j}.\text{qualityIndex}(V, p_{RD}, p_{DD}, C_t)$ .*

Taking values in the interval  $[0; 1]$ , our clustering quality index equals the weighted average quality index of its text unit clusters. Since a greater value indicates a higher average cluster quality, this relative cluster quality index can be used to compare clusterings created by different algorithms and/or different parameters. However, comparing different values of our clustering quality index is only meaningful if they all characterize a text unit clustering that has been discovered (i) in the same intermediate text archive, (ii) in the identical KDT process iteration, and (iii) by employing the same controlled vocabulary. The ADT TextUnitClustering provides an operation that returns the quality index of instantiated text unit clusterings.

Tables 4.17 through 4.19 illustrate our concept of cluster quality by again referring to the small example text archive and its text unit clustering in iteration 1. In particular, Table 4.17 and Table 4.18 list all text unit descriptors that occur in the focal text

**Table 4.19:** Text Unit Clustering after Executing the Bisecting 8-Means in Iteration 1

Cluster Rank	Cluster Number	Quality Index	Qualitatively Acceptable?	Descriptor Coverage	Descriptor Dominance	Cluster Size	See Table(s)
1	4	0.709	Yes	$2/13 \approx 0.154$	$2/2 = 1.000$	7	4.14, 4.17
2	6	0.629	No	$2/13 \approx 0.154$	$2/2 = 1.000$	1	
3	5	0.603	No	$3/13 \approx 0.231$	$3/3 = 1.000$	1	
4	1	0.577	No	$4/13 \approx 0.308$	$4/4 = 1.000$	1	
5	2	0.338	No	$5/13 \approx 0.385$	$1/5 = 0.200$	5	
6	3	0.296	No	$3/13 \approx 0.231$	$0/3 = 0.000$	3	
7	7	0.285	No	$5/13 \approx 0.385$	$0/5 = 0.000$	6	4.15, 4.18
8	0	0.013	No	$0/13 = 0.000$	$(0/0) 0.000$	1	4.16

unit cluster along with their cluster-specific descriptor support. The unacceptable, small cluster visualized in Table 4.16 does not comprise any descriptor at all. The complete text unit clustering obtained by executing the bisecting  $k$ -means algorithm is illustrated in Table 4.19. All eight text unit clusters are ranked by decreasing quality index. For each cluster, the cluster quality criteria are listed along with automatically generated cluster quality decisions. This text unit clustering contains only one acceptable and seven unacceptable text unit clusters based on the following thresholds: dominant descriptor threshold  $p_{DD} = 0.8$ , rare descriptor threshold  $p_{RD} = 0.01$ , maximum descriptor coverage  $p_{macDC} = 0.4$ , minimum descriptor dominance  $p_{minDD} = 0.6$ , and minimum cluster size  $p_{minCS} = 4$ . Cluster 7, shown in Tables 4.15 and 4.18, for example, has a high descriptor coverage of approx. 38.5% and none of its five appearing descriptors is dominant, which results in the minimum descriptor coverage of zero. Due to its low quality index of 0.285, this unacceptable cluster is ranked second worst in the first clustering iteration.

### 4.3.4 Iterative Clustering of Text Unit Vectors

As coarsely outlined in Section 3.3, a clustering algorithm is repeatedly invoked during the DIAsDEM knowledge discovery process. Combined with our framework-specific cluster quality criteria to distinguish acceptable from unacceptable clusters, this approach to pattern discovery is referred to as iterative clustering. In the remainder of this subsection, we explain the concept of iterative clustering in detail, give an illustrative example, and describe the effect of iterative clustering on the KDT process flow.

**Our Notion of Iterative Clustering** The ‘grouping objects’ step of the typical clustering process introduced in Subsection 4.3.1 aims at finding clusters that make explicit the valid structure inherent in the data. Despite the common practice of exploratively executing several algorithms and/or using different parameters to find the ‘best’ clustering, the finally selected combination of clustering method and its parameters is typically executed only once. In most domains it is sufficient to directly find one clustering of the entire

input data set, such as a market segmentation or a taxonomy of astronomical objects.

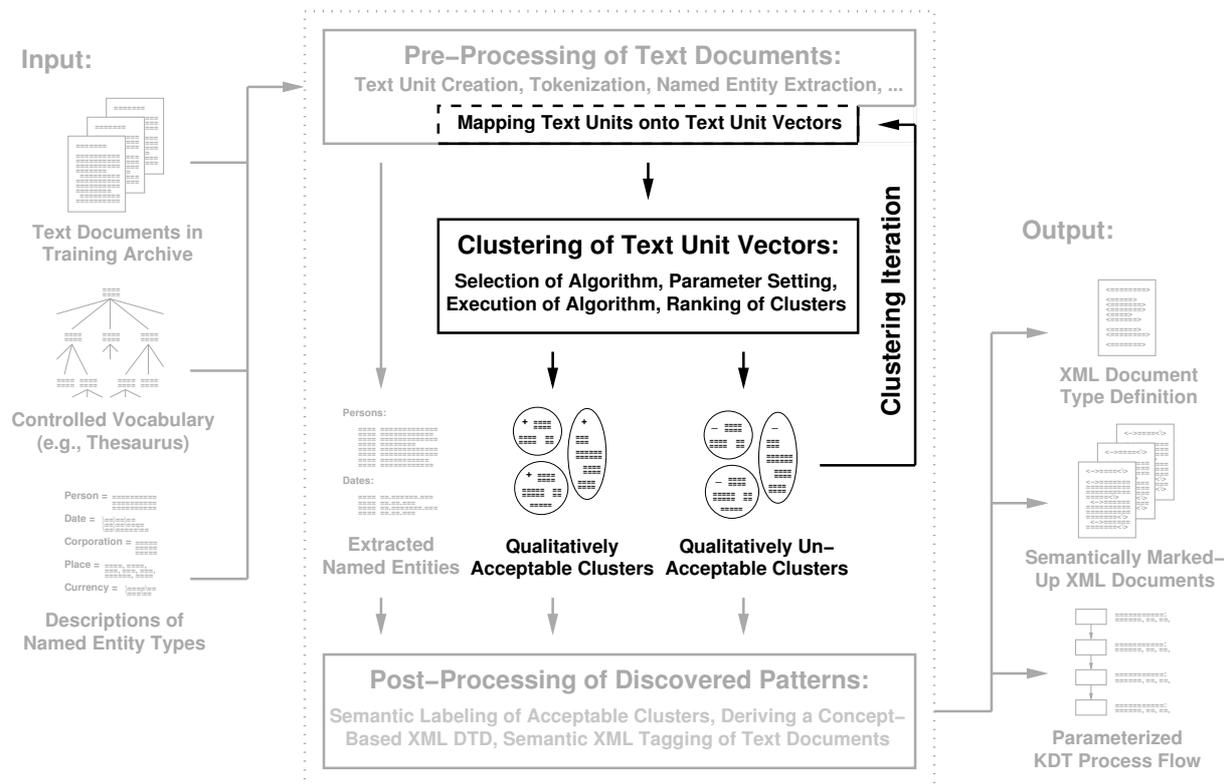
Instead of pursuing this conventional approach to clustering, we advocate applying a slightly different procedure, which we refer to as iterative clustering, for two reasons:

- The DIASDEM knowledge discovery process shall discover both specific and general concepts that occur in the training archive. To that end, we need to find text unit clusters that represent semantic concepts as specific as possible, without missing more general concepts at the same time. Ultimately, semantic XML markup shall characterize the annotated content properly and as precisely as possible.
- Our framework-specific KDT process shall discover more frequent as well as less frequent semantic concepts as long as a domain-specific minimum frequency of occurrence is satisfied. In particular, less frequently appearing themes shall not be suppressed by more frequent ones because they may also convey important topics.

The framework-specific notion of iterative clustering is designed to meet the above mentioned requirements in conjunction with our DIASDEM cluster quality criteria. Unlike internal iterations performed by several clustering algorithms prior to converging towards the final result (e.g., bisecting  $k$ -means; cf. Subsection 4.3.2), we advocate performing clustering iterations outside the employed algorithm. To that end, a clustering algorithm, but not necessarily the same one across all iterations, is repeatedly executed with reduced input data, modified parameters, and/or cluster quality criteria.

Figure 4.2 highlights all components of the DIASDEM knowledge discovery process that are associated with our notion of iterative clustering. Each clustering iteration consists of the following three mandatory steps 1 through 3 and the optional step 4:

1. *Mapping text units onto text unit vectors:* Although converting fully pre-processed text units into real-valued vectors is clearly a data pre-processing step, it is nevertheless an important component of our iterative clustering approach. In iteration 1, all text units in the training archive are mapped onto text unit vectors, as described in Subsection 4.2.5. Subsequent to the initial clustering iteration, only text units assigned to qualitatively unacceptable clusters in the preceding iteration are mapped onto vectors. Due to this selective mapping, the number of text unit vectors to be clustered is typically reduced in each iteration.
2. *Selecting, parameterizing, and executing a clustering algorithm:* Having created an iteration-specific input data set, the KDT expert carefully chooses an appropriate clustering technique and its parameters (cf. Subsection 4.3.2). When parameterizing a selected clustering algorithm, the expert takes into account the current progress of the knowledge discovery process, which is measured by the fraction of text units already assigned to qualitatively acceptable clusters. If possible, the algorithm is at first parameterized to discover frequently occurring and/or very specific thematic concepts. As the number of completed iterations increases, parameters are modified to find more general and/or less frequent semantic concepts as well. The necessary parameter modifications depend on the chosen algorithm.



**Figure 4.2:** Iterative Clustering in the DIASDEM Knowledge Discovery Process

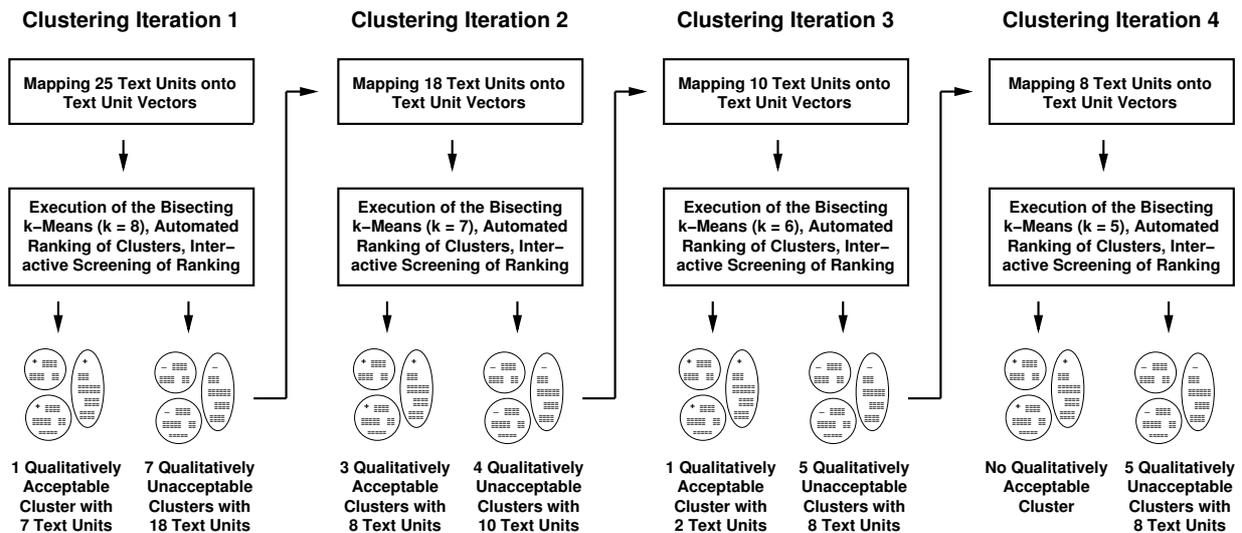
3. *Automatic ranking of the resulting text unit clusters:* Based on DIASDEM cluster quality criteria specified by the KDT expert, all text unit clusters output by the executed algorithm are partitioned into qualitatively acceptable and unacceptable clusters. Within each category, text unit clusters are ranked by decreasing quality index (cf. Subsection 4.3.3) to facilitate an efficient inspection by human experts. As the number of completed iterations increases, qualitatively acceptable clusters tend to be smaller and less homogeneous. Hence, the KDT expert gradually relaxes the thresholds for acceptable clusters by modifying the iteration-specific cluster quality criteria (i.e., dominant descriptor threshold, rare descriptor threshold, maximum descriptor coverage, minimum descriptor dominance, and/or minimum cluster size).
4. *Interactive screening of the cluster ranking:* As emphasized in Section 3.3, the entire knowledge discovery process is interactive, or semi-automated, to ensure a high quality of semantic markup. In particular, we highly recommend checking the automatically derived cluster quality assessments. Although the DIASDEM cluster quality criteria are carefully designed to distinguish ‘good’ clusters from ‘bad’ ones, they cannot guarantee completely error-free assessments due to the complexity and diversity of written language. Thus, a domain expert is asked to approve or reject

automatically generated cluster quality assessments.

Subsequent to the final human cluster validation, all text units assigned to acceptable clusters are put aside for semantic labeling in the post-processing step. By contrast, text units assigned to the remaining, unacceptable clusters constitute the input to the next clustering iteration. If no qualitatively acceptable text unit cluster is discovered at all, the iterative clustering procedure is discontinued, and the post-processing phase of our knowledge discovery process starts. Otherwise, iterative clustering is continued until a stopping criterion specified by the domain or KDT expert is satisfied. Conceivable stopping criteria are, for example, a minimum number of remaining text units or the non-existing subjective interestingness, or relevance, of all/most semantic concepts that are represented by qualitatively acceptable text unit clusters in the current iteration.

To achieve a high quality of markup, our knowledge discovery process is inherently interactive and offers various possibilities for well-directed human intervention. Basically, the expert may select a different clustering algorithm in each iteration. In earlier iterations, for example, SNN clustering (Ertöz et al., 2004) is appropriate to discover highly specific thematic clusters while ignoring thematic outliers. In later iterations, the bisecting  $k$ -means (Steinbach et al., 2000) is, for instance, applicable to find a small, predetermined number of clusters that are more likely to represent less specific thematic concepts. When executing the same algorithm in multiple iterations, the KDT expert may furthermore relax its parameters. For example, gradually reducing the size of a Self-Organizing Map (SOM; cf. Kohonen, 2001) typically results in discovering less specific semantic concepts. Moreover, the human expert may step by step reduce the requirements for acceptable text unit clusters in terms of the applied DIASDEM cluster quality criteria. Finally, mapping text units onto real-valued vectors involves decisions concerning the weighting of text unit descriptors (cf. Subsection 4.2.5), which may also be adjusted between different iterations. When employing the weighting scheme suggested in Subsection 4.2.5, however, the collection frequency components of descriptor weights typically vary without human intervention due to the step-by-step reduction of the collection size from one iteration to the other. The discrepancy between the desire to automatically discover semantic concepts on the one side and the need to incorporate human domain, process, and algorithm knowledge on the other is discussed in Section 4.6.

We present and discuss the successful application of our DIASDEM framework in two real-world case studies in Chapter 6. Despite the achieved success in semantically marking up text documents, two particularities of the iterative clustering approach have to be taken into account when employing our framework. Firstly, one semantic concept might be represented by several text unit clusters. This particularity is mostly caused by the concretely utilized clustering algorithms and their parameter settings. Additionally, it is conceivable that one semantic concept is represented by several combinations of descriptors. While important, a careful parameterization of clustering algorithms thus cannot entirely prevent this first effect. Secondly, text unit vectors cannot be assigned to qualitatively acceptable clusters discovered in any preceding iteration. Due to this particularity, the first effect may of course occur such that the same semantic concept is



**Figure 4.3:** Illustration of Pattern Discovery in Four Clustering Iterations

spread over multiple clusters across different iterations. Furthermore, text unit vectors may be considered as outliers although they represent a semantic concept associated with an acceptable cluster from a preceding iteration. Again, the second effect can be reduced by executing suitable and appropriately parameterized clustering techniques.

The fundamental idea of iterative clustering, namely, the repeated execution of a clustering algorithm on a gradually and interactively reduced data set, resembles the core idea of SCATTER/GATHER (Cutting et al., 1992). Unlike a focused, query-based search for narrowly specified documents, this cluster-based information retrieval system is designed to facilitate an open-ended exploration of large document collections in iterative and interactive browsing sessions. Each iteration comprises a so-called scatter step, in which documents are clustered, and a so-called gather step, in which the user selects interesting document clusters for further, detailed analysis. Initially, the entire archive is segmented, or scattered. Thereafter, only documents assigned to interesting, manually selected clusters form a new sub-collection to be segmented. Hence, SCATTER/GATHER collects all documents considered interesting by the user and re-scatters them to form new groups of similar documents. In contrast, our DIAsDEM framework groups structural text units instead of entire documents. Above all, however, we advocate ‘gathering’ and ‘re-scattering’ text units assigned to unacceptable, uninteresting clusters. Members of acceptable clusters are put aside for semantic labeling in DIAsDEM whereas documents in less interesting clusters are ignored in the following SCATTER/GATHER iterations.

**An Example** Our iterative clustering approach to pattern discovery is illustrated using the small example text archive of news stories  $\check{t}_{E1}$  through  $\check{t}_{E5}$ , which altogether contain 25 sentences (cf. Table 4.1 on page 72). We refrain from discussing the completed

**Table 4.20:** Summary of Pattern Discovery in Four Clustering Iterations

Clustering Iteration	Input Text Units	Parameter of Bisecting $k$ -Means	Dominant Descriptor Threshold	Rare Descriptor Threshold	Maximum Descriptor Coverage	Minimum Descriptor Dominance	Minimum Cluster Size
1	25	$k = 8$	0.80	0.01	0.20	0.60	4
2	18	$k = 7$	0.80	0.01	0.30	0.40	2
3	10	$k = 6$	0.80	0.01	0.40	0.30	2
4	8	$k = 5$	0.80	0.01	0.50	0.20	2

Clustering Iteration	Number of Clusters (Number of Text Units Therein) after:				Stop Iterative Clustering
	Automated Ranking Acceptable	Ranking of Clusters Unacceptable	Interactive Screening Acceptable	Screening of Ranking Unacceptable	
1	1 (7)	7 (18)	1 (7)	7 (18)	No
2	4 (10)	3 (8)	3 (8)	4 (10)	No
3	3 (6)	3 (4)	1 (2)	5 (8)	No
4	1 (2)	4 (6)	0 (0)	5 (8)	Yes

pre-processing steps extensively described throughout Section 4.2. Figure 4.3 schematically depicts the pattern discovery phase, which comprises four clustering iterations. In each iteration, (i) text units are mapped onto vectors using the weighting scheme suggested in Subsection 4.2.5, (ii) the bisecting  $k$ -means algorithm (Steinbach et al., 2000) is executed, (iii) the resulting clusters are automatically ranked to separate qualitatively acceptable clusters from unacceptable ones, and (iv) a domain expert interactively checks and corrects, if necessary, the automatically generated quality assessment.

**Table 4.21:** Sentences Assigned to Qualitatively Acceptable Text Unit Cluster 1 in Iteration 2 (cf. News Items in Table 4.1 on Page 72)

Sentence of Reuters News Item	Item	Text	Sentence
– Chicago newsdesk 312 408-8787	17109	$\check{t}_{E2}$	5
Chicago Newsdesk 312-408-8787	78899	$\check{t}_{E4}$	5
–Chicago Newsdesk 312-408-8787	16106	$\check{t}_{E1}$	4
– New York Newsdesk +1 212 859 1610	115995	$\check{t}_{E5}$	5

The bisecting  $k$ -means, which is used here in combination with cosine similarity, is a suitable technique for our KDT process (cf. Subsection 4.3.2). This algorithm was chosen for illustrative purposes because only one intuitive parameter needs to be set: the number of desired clusters  $k$ . All relevant settings of four performed clustering iterations along with their respective results are compiled in Table 4.20 on page 129.

To find less specific semantic concepts, the number  $k$  of clusters is reduced in each

**Table 4.22:** Sentences Assigned to Qualitatively Acceptable Text Unit Cluster 3 in Iteration 2 (cf. News Items in Table 4.1 on Page 72)

Sentence of Reuters News Item	Item	Text	Sentence
Terms were not disclosed.	16106	$\check{t}_{E1}$	3
Terms were not disclosed.	78899	$\check{t}_{E4}$	4

**Table 4.23:** Sentences Assigned to Qualitatively Unacceptable, Inhomogeneous Text Unit Cluster 2 in Iteration 2 (cf. News Items in Table 4.1 on Page 72)

Sentence of Reuters News Item	Item	Text	Sentence
The company said it has retained the New York investment banking firm of Goldman, Sachs & Co. to explore its strategic options with the 1,500-employee unit.	71177	$\check{t}_{E3}$	3
The mortgage unit, which is headquartered in Woodland Hills, Calif., originated \$2.1 billion in residential first mortgages in the eight months that ended Aug. 31.	71177	$\check{t}_{E3}$	4

iteration. Analogously, the DIAsDEM cluster quality criteria are gradually relaxed. After four iterations, 17 of 25 sentences are assigned to four qualitatively acceptable clusters. The only ‘good’, highly specific text unit cluster discovered in the first iteration is visualized in Table 4.14 on page 116. Recall the unacceptable, rather inhomogeneous cluster found in iteration 1 and listed in Table 4.15 on page 117. In the second iteration, this text unit cluster is split into two distinct, acceptable clusters featuring the semantic concepts ‘editing newsdesk’ (see Table 4.21) and ‘nondisclosure of terms’ (see Table 4.22), respectively. Furthermore, Table 4.23 illustrates a text unit cluster from iteration 2, which is acceptable according to the automated ranking based on DIAsDEM cluster quality criteria. Although the descriptors "unit" and place are truly dominant therein, both sentences do not share a common, domain-specific semantic concept other than the trivial theme ‘statement about a specified business unit’. Since these two sentences are assigned to the same cluster in iterations 2 through 4, the domain expert constantly rejects the automatically generated, positive cluster quality assessment.

**Effect on KDT Process Flow** The iterative clustering phase of our knowledge discovery process starts subsequent to the completed execution of the three algorithms `DecomposeAndTokenizeTextDocuments`, `ExtractNamedEntities`, and `LemmatizeAndDisambiguateWords` in the pre-processing phase. Prior to describing an algorithm that encapsulates a single clustering iteration, we introduce a few framework-specific abstract data types.

The pre-processing step of mapping text units onto vectors is considered a core step

of iterative clustering. Although we have proposed a suitable weighting scheme for common text archives in Subsection 4.2.5, different types of archives may necessitate the application of other weighting schemata. Therefore, we do not define one algorithm for generating text unit vectors, but rather introduce the corresponding abstract data type `DescriptorWeightingScheme` (see Appendix B.34 on page 239). Descriptor weighting schemata typically consist of three components, some of which (e.g., the collection frequency components) are computed once in the knowledge discovery phase and afterwards retrieved for usage in the batch-oriented knowledge application phase. Therefore, the ADT `DescriptorWeightingScheme` provides two different mapping operations as well: `createVectorsInDiscoveryMode( $\bar{a}_T$ : IntTextArchive, V: ControlledVocabulary): IntTextArchive` and `createVectorsInApplicationMode( $\bar{a}_A$ : IntTextArchive, V: ControlledVocabulary): IntTextArchive`. Both operations create text unit vectors for all intermediate text units in the respective intermediate text archive that are not yet assigned to an acceptable cluster and whose semantic concept assumes the null value. If necessary, the operation `createVectorsInDiscoveryMode` additionally computes and persistently stores iteration-invariant weighting components for usage during knowledge application.

The DIASDEM framework for semantic XML tagging incorporates a plug-in concept for conventional clustering algorithms that satisfy the requirements introduced in Subsection 4.3.2. Due to the potentially large number of applicable clustering techniques, we merely define the abstract data type `ClusteringAlgorithm` (see Appendix B.35 on page 240) without elaborating on the intricacies of particular methods. Consequently, instances of this ADT are only characterized by the respective clustering iteration number as well as two strings that contain the name and parameters, respectively, of the algorithm to be executed. This abstract data type provides two main operations, namely, `executeInDiscoveryMode( $\bar{a}_T$ : IntTextArchive): IntTextArchive` and `executeInApplicationMode( $\bar{a}_A$ : IntTextArchive): IntTextArchive`. The former operation is executed in the knowledge discovery phase. Its objective is to discover a new text unit clustering of all text units in the training archive  $\bar{a}$  that are not yet assigned to an acceptable cluster and whose semantic concept hence takes the null value. Subsequent to cluster discovery, algorithm-specific cluster descriptions (e.g., cluster centroids; cf. Subsection 4.3.2) are persistently stored by the respective ADT instance. The latter operation executes the encapsulated algorithm in application mode. This mode utilizes existing cluster descriptions for the purpose of assigning new text unit vectors to existing clusters in the batch-oriented knowledge application phase.

Each clustering iteration is closely associated with the following, iteration-specific metadata: iteration number, parameterized descriptor weighting scheme, parameterized clustering algorithm, DIASDEM cluster quality criteria, and resulting text unit clustering. Since these metadata items are required in the corresponding classification iteration of the knowledge application phase, they have to be persistently stored for subsequent usage as well. For example, the clustering algorithm is executed in application mode to assign new text unit vectors to clusters discovered in a clustering iteration. In addition, the text unit clustering encapsulates the final cluster quality decisions. To fulfill this requirement,

**Algorithm 4.4** PerformClusteringIteration (Outline)

---

**Input:** ( $\bar{a}$ : IntTextArchive,  $m$ : IterationMetadata)**Output:** ( $\bar{a}$ : IntTextArchive,  $m$ : IterationMetadata)

- 1:  $\bar{a} := m.descriptorWeightingScheme().createVectorsInDiscoveryMode(\bar{a})$
  - 2:  $\bar{a} := m.clusteringAlgorithm().executeInDiscoveryMode(\bar{a})$
  - 3:  $C_t: TextUnitClustering := C_t.create(\bar{a}, m.iterationNumber(), m.clusteringAlgorithm().maximumClusterIdentifier())$
  - 4:  $C_t.automatedRankingOfClusters(m)$
  - 5:  $C_t.interactiveScreeningOfClusterRanking(m)$
  - 6: // iteration-specific post-processing of discovered patterns: semantically label acceptable text
  - 7: // unit clusters and update semantic concept of affected intermediate text units accordingly
  - 8:  $m.setTextUnitClustering(C_t)$
- 

we introduce the abstract data type `IterationMetadata` (see Appendix B.37 on page 241). This ADT is instantiated by specifying an iteration number, a parameterized descriptor weighting scheme, a parameterized clustering algorithm, and DIAsDEM cluster quality criteria. Once discovered, an iteration-specific text unit clustering is added to the corresponding instance of the abstract data type `IterationMetadata`.

Algorithm 4.4 represents one clustering iteration. This algorithm takes an intermediate text archive and an instantiated iteration metadata container as input, performs one clustering iteration as specified by the KDT expert when instantiating the metadata container, and finally returns the modified input parameters. In lines 1 through 5 and 8, all iteration-specific operations described in this subsection are performed by executing ADT operations. Apart from the interactive screening, all operations in Algorithm 4.4 are automatically executed after the initial parameterization.

In the next section, we discuss the post-processing phase of our knowledge discovery process in detail. In particular, the important issue of semi-automated cluster labeling is introduced in Subsection 4.4.1. Analogous to creating text unit vectors by pre-processing text units, semantic cluster labeling is a post-processing step in principle. However, as indicated by lines 6 and 7 of algorithm `PerformClusteringIteration`, we nevertheless include this important post-processing step directly in a DIAsDEM clustering iteration. Hence, Algorithm 4.4 is extended accordingly in Subsection 4.4.1. Subsequent to interactively screening ranked text unit clusters, qualitatively acceptable ones are semi-automatically labeled and their text units are assigned the corresponding semantic concept.

## 4.4 Post-Processing of Discovered Patterns

In the generic KDT process outlined in Subsection 2.1.1, all discovered patterns are interpreted and evaluated in the post-processing phase to eliminate uninteresting, useless, and already known patterns. The remaining ones are considered to be knowledge, which is finally applied to attain the initial objective. As indicated in Section 3.3, the post-processing phase of our specific knowledge discovery process encompasses (i) the semi-

automated semantic labeling of qualitatively acceptable clusters introduced in the next subsection, (ii) the automated establishment of a domain-specific, concept-based XML document type definition discussed in Subsection 4.4.2, as well as (iii) the transformation of training texts into semantically tagged XML documents described in Subsection 4.4.3.

#### 4.4.1 Recommending Semantic Cluster Labels

In the preceding section, we have concluded that discovered text unit clusters are partitioned into qualitatively acceptable and unacceptable clusters in each clustering iteration. As part of the post-processing of discovered patterns within each clustering iteration, acceptable text unit clusters are semi-automatically assigned a content-descriptive, semantic label. In accordance with Definitions 1 and 6, a semantic cluster label concisely describes the concept that domain experts typically associate with text units assigned to the respective cluster. To facilitate the creation of high-quality semantic markup, we intentionally take a semi-automated approach to cluster labeling. More concretely, default cluster labels are automatically derived for all acceptable clusters. Subsequently, label suggestions are approved or rejected (i.e., corrected) by the domain expert.

We proceed by briefly reviewing relevant work on labeling clusters of text documents. Subsequently, we introduce the DIASDEM approach to suggesting content-descriptive labels for text unit clusters, which is inspired by related work. We conclude this subsection by outlining the effect of labeling acceptable clusters on the KDT process flow.

**Related Work on Cluster Labeling** In general, as Jarvis and Patrick (1973, p. 1029) emphasized, “clustering cannot provide naming.” The authors clearly separated the task of employing unsupervised learning to find clusters in a data set from assigning names to clusters. The latter task involves attaching meaning to identified clusters and therefore requires an external interpretation by human investigators who have contextual, domain-specific knowledge. Automatic attempts at naming clusters thus cannot guarantee meaningful results. This view apparently holds three decades later. According to Feldman and Sanger (2007, p. 91), for example, “the problem is to give the user a meaningful cluster label.” In this context, Sacco (2000, p. 475) argued that clustering methods are capable of finding concepts in a corpus, but they cannot synthesize human-understandable concept labels. Despite the variety of existing approaches to discovering concepts in texts, Stein and Meyer zu Eissen (2004, p. 353) as well as Treeratpituk and Callan (2006, p. 278) emphasized that less attention had been directed at adequately characterizing them in a concise, content-descriptive, and human-comprehensible way.

Many techniques for labeling clusters of textual data exploit properties of specific clustering algorithms or certain classes of algorithms. Karypis and Han (2000a) as well as Hotho et al. (2003b), for instance, employed labeling methods that are restricted to centroid-based clustering algorithms, like the bisecting  $k$ -means (Steinbach et al., 2000) outlined in Subsection 4.3.2. Both groups of authors characterized clusters of text documents by the  $n \in \mathbb{N}$  highest weighted terms in their centroids. Thereby, emphasis is placed

on the ‘most important’ terms according to the clustering result. Obviously, this labeling method is highly influenced by the chosen term weighting scheme. This approach to cluster labeling, however, does not output labels as created by humans, but rather creates lists of important key words. Hotho et al. (2003b) acknowledged this disadvantage and suggested utilizing an existing, domain-specific ontology (cf. Subsection 2.2.3) to explain clustering results. This extension is not applicable in our DIAsDEM framework because we do not require an existing ontology as input. For analogous reasons, cluster description methods that require a priori given, formalized domain knowledge (e.g., see Jain et al., 1999, pp. 290–292) are not applicable either.

Inferring meaningful descriptions of hierarchically clustered texts is another area of research into cluster labeling. For example, Chakrabarti et al. (1998, p. 165), Glover et al. (2002, 2003), as well as Treeratpituk and Callan (2006) presented techniques designed to infer content characterizations and labels, respectively, for document clusters in a hierarchy. As the DIAsDEM framework does not aim at discovering hierarchies of text unit clusters, we refrain from discussing hierarchical cluster label inference in detail.

Another strain of research is focused on Self-Organizing Maps (cf. Kohonen, 2001), as introduced in Subsection 4.3.2. For instance, the LABELSOM algorithm extracts features from map units, or clusters, that best characterize the input data assigned to these map units (Rauber, 1999; Rauber and Merkl, 2003). Analogous to centroid-based labeling methods, LABELSOM takes advantage of the reference vector associated with each map unit. In contrast, LABELSOM does not extract terms that are assigned the highest weights in each centroid, but outputs terms shared by many documents in a cluster. WEBSOM is a SOM-based document clustering method that reduces the dimensionality of document vectors by means of random projection prior to training a map (see Kohonen, 2001, pp. 286–299). Due to the dimensionality reduction, reference vectors of map units cannot be directly exploited as label clusters. In this context, Lagus and Kaski (1999) presented a WEBSOM-specific method for selecting cluster keywords. The authors argued that good keywords characterize outstanding properties of the focal cluster in relation to the rest of the document archive. Thus, Lagus and Kaski suggested a labeling technique that exploits differences of term frequency distribution between the focal cluster and the entire archive. Because this approach requires the computation of many relative term frequencies, Azcarraga et al. (2004) proposed a similar, yet much faster, labeling algorithm that takes advantage of the random projection method.

Many techniques that are independent of the concretely applied clustering algorithm only take term frequencies into consideration. For example, Cody et al. (2002, pp. 702–703) proposed a so-called category labeling algorithm that is solely based on the occurrence of terms in clusters. To create a label, this method firstly concatenates all words occurring in more than 90% of all documents using the character ‘&’. If no term appears in more than 10% of the documents in a cluster, it is labeled ‘miscellaneous’. Otherwise, the cluster label is a successively created, comma-separated list comprising the most frequent terms. Popescul and Ungar (2000), however, warned of merely considering the cluster-specific term frequency for labeling purposes. Although the list of most frequent words

often reveals the high-level subject, it may fail to convey cluster-specific themes that best differentiate between clusters. Often, the most frequent terms in a cluster are actually meaningless, collection-specific stopwords (e.g., ‘compute’ in computer science abstracts) that do not convey additional information about a cluster’s content.

According to Sanderson and Lawrie (2000, pp. 243–244), the classic approach to selecting terms as components of cluster labels involves searching for ‘unusually frequent’ terms in each cluster. To that end, the authors suggested comparing the frequency of each term’s occurrence with its frequency in the entire collection. Subsequent to ranking all terms appearing in a cluster by decreasing comparison ratio, the top  $n \in \mathbb{N}$  terms serve as cluster labels. Strehl et al. (2000) took a slightly different approach by characterizing each cluster with three descriptive and three discriminative terms. The former ones exhibit the greatest term frequency in the respective cluster whereas the latter terms have the highest frequency multiplier when compared to the average document in the collection. Kulkarni and Pedersen (2005) extracted both descriptive labels defined analogously to Strehl et al. and discriminating labels. The latter ones consist of descriptive terms that do not serve as descriptive labels in any other document cluster. Analogously, Geraci et al. (2006) emphasized that the quality of cluster labels depends on their well-formedness, their descriptive power, and their discriminative power.

**Semantic Cluster Labeling in DIASDEM** Instead of generating truly meaningful labels like the ones created by humans, existing approaches to automated cluster labeling typically extract terms from documents assigned to a cluster that altogether characterize its content. This approach is also adopted in the DIASDEM framework for semantic XML tagging. Due to our plug-in concept for clustering algorithms, we cannot employ cluster labeling techniques that are restricted to a certain clustering algorithm or a family thereof. Hence, our KDT process encompasses an automated, frequency-based method of creating default labels for qualitatively acceptable text unit clusters. To ensure a high quality of semantic XML markup, however, we strongly recommend checking, and if necessary correcting, all default cluster labels by a domain expert. Consequently, text unit clusters are semantically labeled in a two-step process:

1. *Automatic default labeling of acceptable clusters:* Initially, all qualitatively acceptable clusters are assigned a default label. To generate the default label for a text unit cluster, all dominant descriptors occurring therein are identified, ordered by decreasing descriptor dominance, concatenated using the underscore character “\_”, and finally prefixed with “DEFAULT\_”. By utilizing dominant text unit descriptors as content-descriptive terms, we again consider our framework-specific notion of cluster quality introduced in Subsection 4.3.3. Since cluster labels serve as names of semantic XML tags and elements of the concept-based XML DTD, respectively, they are restricted to comprise a limited, standardized set of characters only (see World Wide Web Consortium, 2000). Unlike some approaches to cluster labeling discussed above, we intentionally refrain from using discriminative terms when labeling text unit clusters. Discriminative descriptors are neither determined nor

**Table 4.24:** Summary of Semantic Cluster Labeling in Four Clustering Iterations

Clustering Iteration	Number of Acceptable Cluster	Automatically Created Default Cluster Label	Final Cluster Label upon Completion of Interactive Review	See Table
1	4	DEFAULT_sale_unit	SaleOfBusinessUnit	4.14
2	1	DEFAULT_Newsdesk_place	EditingNewsdesk	4.21
2	3	DEFAULT_termsagreement_not_disclose	NondisclosureOfTerms	4.22
2	6	DEFAULT_expect_closing	ExpectedDealClosing	4.25
3	5	DEFAULT_unit_sale_agreement	AgreementOnUnitSale	4.26

utilized in the labeling step because our DIAsDEM knowledge discovery process aims at finding frequently recurring thematic concepts. The most discriminative terms, however, are often “rather obscure words” (Popescul and Ungar, 2000, p. 1) and thus typically provide limited insight into the prevailing thematic subject of a text unit cluster. Furthermore, structural text units are typically smaller than entire documents addressed by the work discussed above. Clusters of smaller text units tend to be thematically more focused than clusters of larger documents.

2. *Interactive review of default cluster labels:* Ultimately, cluster labels serve as names of semantic XML tags and elements of the concept-based XML DTD, respectively, to reap the benefits of semantic markup (cf. Section 1.3). Merely concatenating dominant descriptors may provide an accurate, coarse, and high-level content description, but descriptor lists rarely facilitate a meaningful, domain-specific, and above all usable markup of text units. To overcome this limitation, we strongly recommend an interactive screening of automatically generated cluster labels. The domain expert is asked to inspect text units assigned to a cluster, or a sample thereof, in combination with cluster-specific frequency statistics of both text unit descriptors and terms not covered by the corresponding controlled vocabulary. Supported by this automatically created cluster visualization, the human expert is capable of efficiently correcting default cluster labels, if necessary. This interactive process involves, but is not limited to, (i) rearranging dominant text unit descriptors in another order, (ii) substituting dominant descriptors with more general terms, (iii) adding words to labels, which are neither descriptors nor non-descriptors, that frequently appear in a cluster, and/or (iv) adding coordinating conjunctions and prepositions, such as ‘of’ or ‘by’. Upon completion of this interactive review, cluster labels explicitly convey informal metadata about the meaning of the respective text units by concisely describing concepts that domain experts typically associate with these text units (cf. Definitions 1 and 6).

Table 4.24 illustrates our framework-specific concept of semantic cluster labeling on the basis of the running example. Each line of this table corresponds to one qualitatively

**Table 4.25:** Sentences Assigned to Qualitatively Acceptable Text Unit Cluster 6 in Iteration 2 (cf. News Items in Table 4.1 on Page 72)

Sentence of Reuters News Item	Item	Text	Sentence
Closing is expected on August 30.	17109	$\check{t}_{E2}$	4
The companies indicated that the sale is expected to be completed by the end of November.	115995	$\check{t}_{E5}$	3

**Table 4.26:** Sentences Assigned to Qualitatively Acceptable Text Unit Cluster 5 in Iteration 3 (cf. News Items in Table 4.1 on Page 72)

Sentence of Reuters News Item	Item	Text	Sentence
Miller Building Systems Inc said it agreed to sell its Miller Structures Inc unit in California to Modtech Inc, a maker of modular classrooms.	78899	$\check{t}_{E4}$	2
VASCO Corp said it agreed to sell its consulting and technical organization, VASCO Performance System, to Wizdom Systems Inc.	16106	$\check{t}_{E1}$	2

acceptable text unit cluster and lists the iteration number, its cluster number, the automatically generated default cluster, the final cluster label chosen by a domain expert, as well as a reference to the table visualizing all text units assigned to the respective cluster. In this context, Tables 4.25 and 4.26 show the content of two qualitatively acceptable text unit clusters that have not yet been referred to in the course of this work. Inspecting all five clusters reveals that list-based default labels, which are prefixed with "DEFAULT\_", apparently provide a correct, high-level content description for text units assigned to the corresponding clusters. For all five clusters, however, the domain expert modified the default label to make explicit the prevailing semantic concept in a concise and human comprehensible way. In this example, dominant descriptors were typically re-ordered and augmented with conjunctions or prepositions (e.g., `AgreementOnUnitSale`). Furthermore, a few words were added to default labels to make the respective context more explicit (e.g., `ExpectedDealClosing`). In particular, the label `EditingNewsdesk` illustrates the necessity of utilizing human domain knowledge.

**Effect on KDT Process Flow** When introducing the algorithm `PerformClusteringIteration` in Subsection 4.3.4 on iterative clustering, we have anticipated that discovered patterns remain to be post-processed prior to starting a new iteration. Acceptable text unit clusters are semantically labeled, and the concept of their intermediate text units is set accordingly. Consequently, Algorithm 4.5 on page 138 extends the algorithm `PerformClusteringIteration` such that each clustering iteration is finalized by appropriately labeling discovered text unit clusters that are considered qualitatively acceptable.

**Algorithm 4.5** PerformClusteringIteration (Complete)

---

**Input:** ( $\bar{a}$ : IntTextArchive,  $m$ : IterationMetadata)**Output:** ( $\bar{a}$ : IntTextArchive,  $m$ : IterationMetadata)

- 1:  $\bar{a} := m.descriptorWeightingScheme().createVectorsInDiscoveryMode(\bar{a})$
  - 2:  $\bar{a} := m.clusteringAlgorithm().executeInDiscoveryMode(\bar{a})$
  - 3:  $C_t: TextUnitClustering := C_t.create(\bar{a}, m.iterationNumber(), m.clusteringAlgorithm().maximumClusterIdentifier())$
  - 4:  $C_t.automatedRankingOfClusters(m)$
  - 5:  $C_t.interactiveScreeningOfClusterRanking(m)$
  - 6:  $C_t.automatedLabelingOfClusters(m)$
  - 7:  $C_t.interactiveReviewOfClusterLabels(m)$
  - 8:  $m.setTextUnitClustering(C_t)$
- 

Subsequent to automatically ranking clusters and interactively screening the result, the operation `automatedLabelingOfClusters(m: IterationMetadata)` of abstract data type `TextUnitClustering` is executed (cf. line 6 of Algorithm 4.5). For each qualitatively acceptable cluster in the iteration-specific text unit clustering, a default label is automatically created. More specifically, cluster labels are encapsulated by the corresponding concepts (see Definition 6 on page 59 and Appendix B.8). In addition, all intermediate text units assigned to acceptable clusters are updated accordingly to associate them with semantic concepts as well. The strongly recommended interactive review of default cluster labels is performed by executing the operation `interactiveReviewOfClusterLabels(m: IterationMetadata)` of abstract data type `TextUnitClustering`. This operation supports domain experts in interactively approving or, if necessary, correcting automatically generated cluster labels. Correcting cluster labels and concepts, respectively, also triggers a concept update of intermediate text units assigned to the respective clusters. At the end of each clustering iteration, all intermediate text units assigned to qualitatively acceptable clusters comprise the corresponding concept. Analogously, concepts of intermediate text units assigned to unacceptable clusters take the null value. Unless the iterative clustering process is discontinued, the latter text units are input to the next iteration because they are assigned to unacceptable and thus unlabeled clusters.

#### 4.4.2 Establishing a Concept-Based XML DTD

Upon completion of the iterative clustering phase, two important steps of the post-processing phase remain to be accomplished. The final conversion of the entire training text archive into semantically marked-up XML documents requires an existing concept-based XML document type definition. In this subsection, we focus on the task of deriving this XML document type definition and therefore concisely recap relevant terminology, describe the process of establishing a concept-based DTD, present an example, and outline the effect of DTD creation on the KDT process flow.

**Terminology** In Definition 12 on page 61, we have introduced the framework-specific notion of conceptual document structures, whose instances are encapsulated by the abstract data type `ConceptualDocumentStructure` (see Appendix B.18 on page 227). Each instance of this ADT encompasses the information necessary to establish the corresponding XML DTD: the concepts identified during the pattern discovery phase along with frequently occurring named entity types. Following the construction of a conceptual document structure from an intermediate text archive, the corresponding XML document type definition can be straightforwardly output by applying the syntax rules defined by the World Wide Web Consortium (2000). To enable a valid semantic XML markup of new text documents originating from the same domain as the training archive, the resulting conceptual document structure is persistently stored for usage in the batch-oriented knowledge application phase of our framework.

Subsequent to its creation, a conceptual document structure comprises all concepts discovered in the corresponding intermediate text archive. Furthermore, the resulting conceptual document structure establishes a connection between concepts and named entity types that frequently appear in intermediate text units assigned to them. To focus on concept-specific named entity types that typically occur within a certain semantic context, infrequent or noisy occurrences of named entity types are omitted when constructing a conceptual document structure. Particularities of the domain under consideration often necessitate utilizing an adjustable threshold for distinguishing between signal (i.e., typical named entity types in text units featuring a certain concept) and noise. Since named entity types ultimately correspond to attributes of semantic XML tags, we introduce our framework-specific notion of attribute support as follows:

**Definition 40 (Attribute Support)** *Let  $\bar{a}$ : IntTextArchive denote an intermediate text archive. The attribute support  $\bar{a}.\text{attributeSupport}(o, p)$  of named entity type  $p$ : NamedEntityType within intermediate text units of  $\bar{a}$  assigned to concept  $o$ : Concept is defined as the ratio of the number of intermediate text units in  $\bar{a}$  that are assigned to concept  $o$  and whose set of intermediate named entities contains a named entity of type  $p$ , to the total number of intermediate text units in  $\bar{a}$  assigned to concept  $o$ . If no intermediate text unit is assigned to concept  $o$ ,  $\bar{a}.\text{attributeSupport}(o, p)$  is defined to be zero.*

A formal, multi-set-based definition of attribute support is given in the specification of the ADT `IntTextArchive` in Appendix B.29. Attribute support takes values in the interval  $[0; 1]$ . Analogous to our notion of descriptor support, this term is inspired by the support of association rules in a data set (cf. Agrawal et al., 1993, p. 208). Attribute support serves as a thresholding input parameter and thus allows for incorporating domain knowledge in the knowledge discovery process. For typical text archives, a threshold of 0.1 constitutes an appropriate setting. In this case, a named entity type only becomes an attribute of a certain XML tag if at least 10% of the text units assigned to the corresponding semantic concept comprise named entities of the focal type.

**Table 4.27:** Labels of Conceptual Document Structure `NewsItem` Describing the Exemplary Text Archive (cf. News Items in Table 4.1 on Page 72)

$i$	Label of Concept $i$	Labels of Named Entity Types Occurring within Concept $i$
1	<code>AgreementOnUnitSale</code>	<code>Company</code> , <code>Place</code>
2	<code>EditingNewsdesk</code>	<code>Place</code>
3	<code>ExpectedDealClosing</code>	<code>Date</code>
4	<code>NondisclosureOfTerms</code>	
5	<code>SaleOfBusinessUnit</code>	<code>Company</code>

**From Conceptual Document Structures to Concept-Based XML DTDs** As introduced in Chapter 3, a conceptual document structure is once created in the knowledge discovery phase of the DIASDEM framework. Initializing a domain-specific conceptual document structure requires (i) enumerating the distinct semantic concepts identified during the preceding iterative pattern discovery phase, (ii) collecting the concept-specific set of extracted named entity types for each semantic concept, and finally (iii) applying a user-supplied attribute support threshold to remove noise from the concept-specific lists of named entity types. This procedure is encapsulated by the constructor `create( $\check{l}_{init}$ : String,  $\bar{a}$ : IntTextArchive,  $p_{AS}$ : Real)` of the abstract data type `ConceptualDocumentStructure` (see Appendix B.18 on page 227). Given the alphanumeric label  $\check{l}_{init}$  that concisely describes the common theme of text documents in intermediate text archive  $\bar{a}$  and the attribute support threshold  $p_{AS}$ , this constructor performs one pass over all text units in  $\bar{a}$  to collect semantic concepts and named entity types occurring in their respective text units as well as one pass over all concept-specific sets of named entity types to apply the attribute support threshold.

Table 4.27 lists the main content of the conceptual document structure established for the small exemplary text archive used throughout this chapter. Automatically created on the basis of an attribute support of 10%, this conceptual document structure comprises all semantic concepts discovered and labeled during the pattern discovery phase (cf. Table 4.24 on page 136). For instance, at least 10% of all text units featuring the concept labeled `AgreementOnUnitSale` contain named entities of type `Company` and `Place`, respectively. The corresponding text unit cluster is visualized in Table 4.26 on page 137. By contrast, there are no named entity types whose instances appear in at least 10% of all text units that are associated with the concept `NondisclosureOfTerms`.

The operation `xmlDtd()` of abstract data type `ConceptualDocumentStructure` outputs the concept-based XML document type definition as a string. This DTD consists of domain-specific declarations in a syntax standardized by the World Wide Web Consortium (2000) that define both the components and the structure of semantically marked-up XML documents (cf. Geroimenko, 2004, p. 42). Table 4.28 lists an exemplary, concept-based XML document type definition. It defines the components and the structure of semantically tagged news items from the example text archive on announcements of busi-

**Table 4.28:** Line-Numbered, Concept-Based XML Document Type Definition of the Conceptual Document Structure `NewsItem` (cf. Table 4.27)

---

```

1: <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3: <!ELEMENT NewsItem (MetaData*, TaggedDocument)>
4:
5: <!ELEMENT MetaData (Name, Content)>
6: <!ELEMENT Name (#PCDATA)>
7: <!ELEMENT Content (#PCDATA)>
8:
9: <!ELEMENT TaggedDocument (#PCDATA
10:   | AgreementOnUnitSale
11:   | EditingNewsdesk
12:   | ExpectedDealClosing
13:   | NondisclosureOfTerms
14:   | SaleOfBusinessUnit
15: )* >
16:
17: <!ELEMENT AgreementOnUnitSale (#PCDATA)>
18: <!ELEMENT EditingNewsdesk (#PCDATA)>
19: <!ELEMENT ExpectedDealClosing (#PCDATA)>
20: <!ELEMENT NondisclosureOfTerms (#PCDATA)>
21: <!ELEMENT SaleOfBusinessUnit (#PCDATA)>
22:
23: <!ATTLIST AgreementOnUnitSale Company CDATA #IMPLIED>
24: <!ATTLIST AgreementOnUnitSale Place CDATA #IMPLIED>
25: <!ATTLIST EditingNewsdesk Place CDATA #IMPLIED>
26: <!ATTLIST ExpectedDealClosing Date CDATA #IMPLIED>
27: <!ATTLIST SaleOfBusinessUnit Company CDATA #IMPLIED>

```

---

ness unit sales. The construction of concept-based DTDs is henceforth coarsely described and illustrated by referring to Table 4.28. The DTD generation involves five major steps, each of which performs basic string concatenations:

1. Create the XML document type definition header that consists of an XML declaration specifying the XML version and the file encoding (e.g., see line 1 of Table 4.28) and the declaration of the root element, such as `NewsItem` defined in line 3 of Table 4.28. In DTDs created within our framework, the root element solely consists of two types of children. The optional sequence of metadata elements labeled `MetaData` allows for the persistent storage of metadata about the original text document (e.g., its source file name). Metadata management is not covered by this work. The mandatory element `TaggedDocument` comprises the results of converting the original text document into a semantically marked-up document.
2. Generate a DTD section that defines the structure of the metadata element named `MetaData`. As illustrated in lines 5 through 7 of Table 4.28, the `MetaData` element

**Algorithm 4.6** EstablishConceptualDocumentStructure**Input:** ( $\bar{a}$ : IntTextArchive,  $\bar{l}$ : String,  $p_{AS}$ : Real)**Output:** ( $\hat{s}$ : ConceptualDocumentStructure)1:  $\hat{s} := \text{ConceptualDocumentStructure.create}(\bar{l}, \bar{a}, p_{AS})$ 

consists of two sub-elements (i.e., **Name** and **Content**) that contain the actual meta-data in the form of character data. The keyword **#PCDATA** is historically derived from ‘parsed character data’ (cf. World Wide Web Consortium, 2000).

3. Output a DTD section that specifies the content of the element **TaggedDocument**. Since our framework for semantic XML tagging aims at marking up structural, non-overlapping text units (cf. Definition 5 of text unit layers), the element **TaggedDocument** consists of mixed content in the form of marked-up text units, which are represented by elements labeled in accordance with semantic concepts, and plain text units (i.e., character data specified by the keyword **#PCDATA**). The tagged document section is a sequence of marked-up and plain text units. In particular, DTD elements may occur multiple times within a single marked-up document. In lines 9 through 15 of Table 4.28, the element **TaggedDocument** is declared for the exemplary text archive and enumerates the labels of all discovered concepts.
4. For each DTD element that represents a semantic concept, create the corresponding element declaration. As indicated by the stand-alone keyword **#PCDATA** in lines 17 through 21 of Table 4.28, these elements are defined to comprise merely character content as our framework is intentionally restricted to generate non-overlapping text unit markup only.
5. For DTD elements representing a semantic concept associated with at least one named entity type in the conceptual document structure, generate the corresponding attribute declarations. As illustrated in lines 23 through 27 of Table 4.28, each attribute declaration associates an element name (e.g., **AgreementOnUnitSale** in line 23) with one named entity type label (e.g., **Company** in line 23). The keyword **CDATA** specifies that only string literals are allowed attribute values. All attributes of XML tags are declared optional by the keyword **#IMPLIED**.

Once the concept-based XML document type definition is created, a copy is saved in the local file system and subsequently referenced by automatically generated XML documents. In accordance with the objectives of our framework, the generated concept-based DTD neither supports nested semantic XML tags nor imposes any other structural constraints on the sequence of XML tags within marked-up documents.

**Effect on KDT Process Flow** Algorithm 4.6 (EstablishConceptualDocumentStructure) is executed subsequent to completing the final clustering iteration. This algorithm requires three input parameters: the focal intermediate text archive  $\bar{a}$ , an alphanumeric label  $\bar{l}$  that describes the common theme of text documents therein, and the attribute support threshold  $p_{AS}$ . Algorithm 4.6 instantiates a conceptual document structure for the intermediate

**Algorithm 4.7** CreateSemanticallyMarkedUpTextArchive**Input:** ( $\bar{a}$ : IntTextArchive,  $\hat{s}$ : ConceptualDocumentStructure)**Output:** ( $\hat{a}$ : SmuTextArchive)

---

```

1:  $\hat{a}$ : SmuTextArchive :=  $\hat{a}$ .create() // initialize output smu. text archive
2: for all  $i$ : Integer := 1, 2, ...,  $\bar{a}$ .size() do // iterate through int. text documents
3:    $\bar{t}$ : IntTextDocument :=  $\bar{a}$ .intTextDocument( $i$ ) // extract  $i$ th int. text document
4:    $\bar{r}$ : IntTextUnitLayer :=  $\bar{t}$ .intTextUnitLayer() // extract int. text unit layer
5:    $\hat{r}$ : SmuTextUnitLayer :=  $\hat{r}$ .create() // initialize output smu. text unit layer
6:   for all  $j$ : Integer := 1, 2, ...,  $\bar{r}$ .size() do // iterate through int. text units
7:      $\bar{u}$ : IntTextUnit :=  $\bar{r}$ .intTextUnit( $j$ ) // extract  $j$ th int. text unit
8:      $\hat{u}$ : SmuTextUnit := null // declare output smu. text unit
9:     if  $\bar{u}$ .concept()  $\neq$  null and  $\hat{s}$ .contains( $\bar{u}$ .concept()) then // check for valid concept
10:       $\hat{u}$  :=  $\hat{u}$ .create( $\bar{u}$ .originalTextUnit(),  $\bar{u}$ .concept()),
11:       $\hat{s}$ .validNamedEntities( $\bar{u}$ .concept(),  $\bar{u}$ .intNamedEntities()) // text unit is marked-up
12:     else
13:       $\hat{u}$  :=  $\hat{u}$ .create( $\bar{u}$ .originalTextUnit(), null, null) // text unit is not marked-up
14:     end if
15:      $\hat{r}$ .appendSmuTextUnit( $\hat{u}$ ) // append new smu. text unit
16:   end for
17:    $\hat{t}$ : SmuTextDocument :=  $\hat{t}$ .create( $\bar{t}$ .textDocument(),  $\hat{r}$ ) // create output smu. text document
18:    $\hat{a}$ .appendSmuTextDocument( $\hat{t}$ ) // append smu. text document
19: end for

```

---

text archive under consideration, as introduced in this subsection. The result constitutes input to semantic XML tagging, which is performed in the final post-processing step of both the knowledge discovery process and the batch-oriented knowledge application process. Hence, the domain-specific conceptual document structure is persistently stored for future usage.

### 4.4.3 Semantic XML Tagging of Text Documents

The post-processing step of our framework-specific knowledge discovery phase is concluded by transforming the intermediate text archive into a semantically marked-up text archive, as specified in Definition 11 on page 61. In addition, semantically tagged XML documents remain to be created for the purpose of evaluating the quality of semantic markup. In this subsection, we describe these two post-processing steps and outline their effect on the KDT process flow.

**Creating a Semantically Marked-Up Text Archive** Given a domain-specific conceptual document structure, Algorithm 4.7 (CreateSemanticallyMarkedUpTextArchive) converts the input intermediate text archive into a semantically marked-up text archive. This conversion procedure iterates through all intermediate text documents and their text units to instantiate the corresponding semantically marked-up text documents.

Since our framework is designed to identify only frequently recurring thematic concepts at the text unit level, original text units are either semantically annotated, if

**Table 4.29:** Semantically Marked-Up Text Units of the Text Document  $\check{t}_{E1}$  (cf. Table 4.1 on Page 72 and Table 4.2 on Page 76)

---

$\hat{u}_{E1,1}$ : SmuTextUnit = ( $\langle$ "USA: VASCO says to sell consulting unit." $\rangle$ , saleOfBusinessUnit, {(company, "ID: 1; Name: VASCO")})
$\hat{u}_{E1,2}$ : SmuTextUnit = ( $\langle$ "VASCO Corp said it agreed to sell its consulting and technical organization, VASCO Performance System, to Wizdom Systems Inc." $\rangle$ , agreementOnUnitSale, {(company, "ID: 5; Name: VASCO Corp"), (company, "ID: 6; Name: VASCO Performance System"), (company, "ID: 7; Name: Wizdom Systems Inc.")})
$\hat{u}_{E1,3}$ : SmuTextUnit = ( $\langle$ "Terms were not disclosed." $\rangle$ , nondisclosureOfTerms, $\emptyset$ )
$\hat{u}_{E1,4}$ : SmuTextUnit = ( $\langle$ "-Chicago Newsdesk 312-408-8787" $\rangle$ , editingNewsdesk, {(place, "Chicago")})

---

they are assigned a concept, or copied without semantic markup into the marked-up document. Consequently, two cases are distinguished in lines 9 through 14 of Algorithm 4.7. If the concept associated with an intermediate text unit assumes the null value or if it is not contained in the domain-specific conceptual document structure, the corresponding semantically marked-up text unit is in fact neither assigned a thematic concept nor associated with named entities occurring therein. Otherwise, the new semantically marked-up text unit is associated with the respective concept and a set of named entities identified in the original text unit. However, the operation  $\text{validNamedEntities}(O_{\text{input}}: \text{Concept}, P_{\text{input}}: \text{SetOfIntNamedEntities}): \text{SetOfNamedEntities}$  of ADT  $\text{ConceptualDocumentStructure}$  ensures that the latter set comprises only named entities whose type is valid in the context of the respective semantic concept, as defined by the conceptual document structure.

Table 4.29 illustrates the results of converting the text units of exemplary text  $\check{t}_{E1}$  into semantically marked-up text units on the basis of the conceptual document structure established beforehand (cf. Table 4.27). In this example, all text units are assigned a semantic concept and a possibly empty set of extracted named entities. In accordance with Definition 8 on page 60, the semantically marked-up text unit  $\hat{u}_{E1,1}$  is a 3-tuple comprising the original text unit (i.e.,  $\langle$ "USA: VASCO says to sell consulting unit." $\rangle$ ), the domain-specific concept that experts typically associate with similar text units (i.e.,  $\text{saleOfBusinessUnit}$ ), and the set of extracted named entities. The latter contains only one element, namely, (company, "ID: 1; Name: VASCO") mentioned in the text unit. For the sake of clarity, alphanumeric labels of semantic concepts (e.g.,  $\text{AgreementOnUnitSale}$  in  $\hat{u}_{E1,2}$ ) and named entity types (e.g.,  $\text{Company}$  in  $\hat{u}_{E1,2}$ ) are omitted in Table 4.29. Furthermore, the semantically marked-up text unit  $\hat{u}_{E1,3}$  contains an empty set of named entities because no instances of valid types have been extracted during the pre-processing phase. If a text unit were not assigned a semantic concept at all, the concept and the set of named entities would assume the null value in the corresponding intermediate text unit. For instance, the marked-up text unit  $\hat{u}_{E4,3}$ : SmuTextUnit = ( $\langle$ "Final settlement is

**Table 4.30:** Line-Numbered, Semantically Marked-Up XML Document Created by Tagging the Text Document  $\check{t}_{E1}$  (cf. Table 4.1 on Page 72)

---

1:	<?xml version="1.0" encoding="ISO-8859-1"?>
2:	<!DOCTYPE NewsItem SYSTEM "NewsItem.dtd">
3:	
4:	<NewsItem>
5:	<TaggedDocument>
6:	<SaleOfBusinessUnit Company="ID: 1; Name: VASCO">USA: VASCO says to sell consulting unit.</SaleOfBusinessUnit>
7:	<AgreementOnUnitSale Company="ID: 5; Name: VASCO Corp [AND] ID: 6; Name: VASCO Performance System [AND] ID: 7; Name: Wizdom Systems Inc.">VASCO Corp said it agreed to sell its consulting and technical organization, VASCO Performance System, to Wizdom Systems Inc.</AgreementOnUnitSale>
8:	<NondisclosureOfTerms>Terms were not disclosed.</NondisclosureOfTerms>
9:	<EditingNewsdesk Place="Chicago">-Chicago Newsdesk 312-408-8787</EditingNewsdesk>
10:	</TaggedDocument>
11:	</NewsItem>

---

scheduled for October 21, Miller said."}, null, null) of the forth exemplary text (cf. Table 4.1 on Page 72) is not assigned to a semantic concept.

**Outputting Semantically Tagged XML Documents** Subsequent to creating a semantically marked-up text archive, the operation `xmlDocuments(â: SmuTextArchive): String[]` provided by ADT `ConceptualDocumentStructure` outputs semantically tagged XML documents into an array of strings. More specifically, each string corresponds to one valid, semantically tagged XML document that adheres to the XML specification (cf. World Wide Web Consortium, 2000). Furthermore, the structure of each XML document conforms to the concept-based XML document type definition established beforehand. Creating a semantically tagged XML document that corresponds to one specific semantically marked-up text document involves the following steps. Analogous to DTD creation, these steps mainly involve template-based string operations.

1. Generate the XML document prolog that consists of an XML declaration specifying the XML version and the file encoding (e.g., see line 1 of Table 4.30), as well as the document type declaration (i.e., `NewsItem` defined in line 2 of Table 4.30), which references the concept-based document type definition depicted in Table 4.28.
2. Initialize the XML document body by appending the start tag of the root element (e.g., `<NewsItem>` in line 4 of Table 4.30). Optionally, append a sequence of metadata elements, such as `<MetaData><Name>lang</Name><Content>en</Content></MetaData>`. Since metadata management is not covered by this work, Table 4.30 does not contain this section. Moreover, append the start tag `<TaggedDocument>`

**Table 4.31:** Line-Numbered, Semantically Marked-Up XML Document Created by Tagging the Text Document  $\check{t}_{E4}$  (cf. Table 4.1 on Page 72)

---

1:	<?xml version="1.0" encoding="ISO-8859-1"?>
2:	<!DOCTYPE NewsItem SYSTEM "NewsItem.dtd">
3:	
4:	<NewsItem>
5:	<TaggedDocument>
6:	<SaleOfBusinessUnit Company="ID: 1; Name: Modtech">USA: Miller to sell unit to Modtech.</SaleOfBusinessUnit>
7:	<AgreementOnUnitSale Company="ID: 6; Name: Miller Building Systems Inc [AND] ID: 7; Name: Miller Structures Inc [AND] ID: 8; Name: Modtech Inc" Place="California">Miller Building Systems Inc said it agreed to sell its Miller Structures Inc unit in California to Modtech Inc, a maker of modular classrooms.</AgreementOnUnitSale>
8:	Final settlement is scheduled for October 21, Miller said.
9:	<NondisclosureOfTerms>Terms were not disclosed.</NondisclosureOfTerms>
10:	<EditingNewsdesk Place="Chicago">-Chicago Newsdesk 312-408-8787</EditingNewsdesk>
11:	</TaggedDocument>
12:	</NewsItem>

---

(cf. line 5 of Table 4.30) to open the section containing both semantically marked-up and plain text units in the order of their occurrence in the original document.

3. Sequentially iterate through all semantically marked-up text units contained in the text unit layer of the semantically marked-up text document to be converted into an XML document. Since text units are either assigned to a semantic concept or left without an annotation, we distinguish the following two cases:
  - a) If the text unit shall not be marked-up and its concept assumes the null value, respectively, append a string that merely contains the original text unit. This task can be accomplished by executing the operation `textUnit(): TextUnit` of ADT `SmuTextUnit` followed by the operation `content(): String` of ADT `TextUnit`. For example, line 8 of Table 4.31 represents a sentence without semantic markup as its content is not enclosed by semantic XML tags.
  - b) If the text unit shall be marked-up and its concept assumes a value other than null, the original text unit is initially retrieved as described above. Subsequently, the original text unit is enclosed by a semantic start tag and the corresponding end tag. Both tags have the same element name that is equal to the label of the semantic concept. Hence, the element name is created via the operation `concept(): Concept` of ADT `SmuTextUnit` followed by the operation `label(): String` of ADT `Concept`. Furthermore, the start tag is augmented with attributes if the focal semantically marked-up text unit comprises a non-empty set of named entities. To this end, an attribute name-value pair is constructed

**Algorithm 4.8** CreateSemanticallyTaggedXmlDocuments**Input:** ( $\bar{a}$ : IntTextArchive,  $\hat{s}$ : ConceptualDocumentStructure)**Output:** ( $\hat{a}$ : SmuTextArchive,  $\hat{x}_{\text{DTD}}$ : String,  $\hat{x}_{\text{Docs}}$ : String[])

```

1:  $\hat{a} := \text{CreateSemanticallyMarkedUpTextArchive}(\bar{a}, \hat{s})$  // convert int. into smu. text archive
2:  $\hat{x}_{\text{DTD}} := \hat{s}.\text{xmlDtd}()$  // generate concept-based XML DTD
3:  $\hat{x}_{\text{Docs}} := \hat{s}.\text{xmlDocuments}(\hat{a})$  // generate semantically tagged XML documents

```

for each named entity in the respective set of named entities. For each named entity, the attribute name equals the label of its type that is directly retrieved by executing the operation `label(): String` of ADT `NamedEntity`. Its attribute value corresponds to the canonical string representation of a named entity (cf. Subsection 4.2.2), which is accessible via the operation `value(): String` of ADT `NamedEntity`. For example, line 9 of Table 4.31 represents a marked-up sentence without any extracted named entities. In contrast, three companies and one place are extracted from the marked-up sentence listed in line 7.

4. Finally, append two end tags (i.e., `</TaggedDocument>` and `</NewsItem>`) that close the respective sections. The former one terminates the section of semantically annotated text units whereas the latter end tag closes the entire XML document body. The usage of these tags is illustrated in lines 11 and 12 of Table 4.31.

After creating an array of strings, each of which represents one semantically tagged XML document, a standard file operation is executed to copy all XML documents to the local file system. In the knowledge discovery phase of our framework, a random sample of these XML documents is utilized to assess the tagging quality. The quality evaluation of semantic markup is discussed in Section 6.1. In the knowledge application phase, semantically tagged XML documents constitute the major output.

**Effect on KDT Process Flow** Having established a conceptual document structure, Algorithm 4.8 (`CreateSemanticallyTaggedXmlDocuments`) is executed to conclude our knowledge discovery process by converting the intermediate text archive into a semantically marked-up text archive. Given intermediate text archive  $\bar{a}$  and conceptual document structure  $\hat{s}$ , this algorithm transforms  $\bar{a}$  into the semantically marked-up text archive  $\hat{a}$  using the algorithm `CreateSemanticallyMarkedUpTextArchive` described above in detail. In addition, both the concept-based XML document type definition  $\hat{x}_{\text{DTD}}$  and the array  $\hat{x}_{\text{Docs}}$ , which contains all semantically tagged XML documents, are created and returned. To this end, Algorithm 4.8 utilizes two operations provided by ADT `ConceptualDocumentStructure`, which have been introduced above and in the preceding subsection, respectively. The string representation of the XML document type definition is referenced by all XML documents. We deliberately refrain from discussing any issues related to exporting the string-based results of Algorithm 4.8 to the local file system or an XML-enabled database.

## 4.5 Bridging Knowledge Discovery and Knowledge Application

Initially, this section summarizes the generic KDT process flow for the knowledge discovery phase of the DIASDEM framework, which has been described in detail in the preceding sections. Once parameterized during an interactive knowledge discovery session, this KDT process flow constitutes the basis for an automated XML tagging of thematically similar text documents. Subsequently, we introduce the framework-specific notion of classification iterations that are performed in the batch-oriented knowledge application phase. For each clustering iteration executed during the iterative pattern discovery, there exists a corresponding classification iteration that invokes the clustering algorithm in application mode to assign new text units to previously identified text unit clusters. Finally, we outline the generic KDT process flow for the fully automated knowledge application phase. This generic process flow illustrates the sequence of parameterized KDT algorithms, which is executed without human intervention during knowledge application.

**KDT Process Flow for Knowledge Discovery Phase** Algorithm 4.9 abstractly summarizes the entire generic KDT process flow that is parameterized and executed during the knowledge discovery phase of our DIASDEM framework for semantic XML tagging. In the preceding sections of this chapter, all relevant pre-processing, pattern discovery, and post-processing steps have been discussed at the necessary level of detail.

As illustrated in Figure 3.2 on page 65, the interactive KDT process takes as input a training archive of plain text documents, a set of named entity types to be extracted during pre-processing, and a domain-specific controlled vocabulary. Upon completion of the KDT process, the following output is created: a domain-specific conceptual document structure along with a semantically marked-up training archive, a concept-based XML document type definition, and one semantically marked-up XML document for each input training text document. Furthermore, an instance of the abstract data type `KdtProcessMetadata` (see Appendix B.38 on page 242) is created, which enables the persistent storage of important metadata associated with one specific parameterized KDT process flow. Besides storing the set of relevant named entity types, this process flow metadata container encapsulates all iteration metadata items generated during the interactive, iterative pattern discovery. Along with the conceptual document structure derived during knowledge discovery, the KDT process metadata container is an important input for the knowledge application process.

Subsequent to instantiating a KDT process metadata container in line 1 of Algorithm 4.9, the input text archive is pre-processed in lines 2 through 4, as described in Section 4.2. The iteration identifier  $t$  is instantiated in line 5 prior to starting the pattern discovery step, which corresponds to lines 6 through 14. Within each clustering iteration, a KDT expert interactively parameterizes the descriptor weighting scheme, an appropriate clustering algorithm, and the set of cluster quality criteria (cf. Section 4.3). Each clustering iteration is concluded by persistently saving the corresponding iteration

**Algorithm 4.9** Generic KDT Process Flow for the Knowledge Discovery Phase

---

**Input:** ( $\hat{a}_T$ : TextArchive, P: SetOfNamedEntityTypes, V: ControlledVocabulary)  
**Output:** ( $\hat{s}$ : ConceptualDocumentStructure, k: KdtProcessMetadata,  
 $\hat{a}_T$ : SmuTextArchive,  $\hat{x}_{D_{TD}}$ : String,  $\hat{x}_{TD_{ocs}}$ : String[])

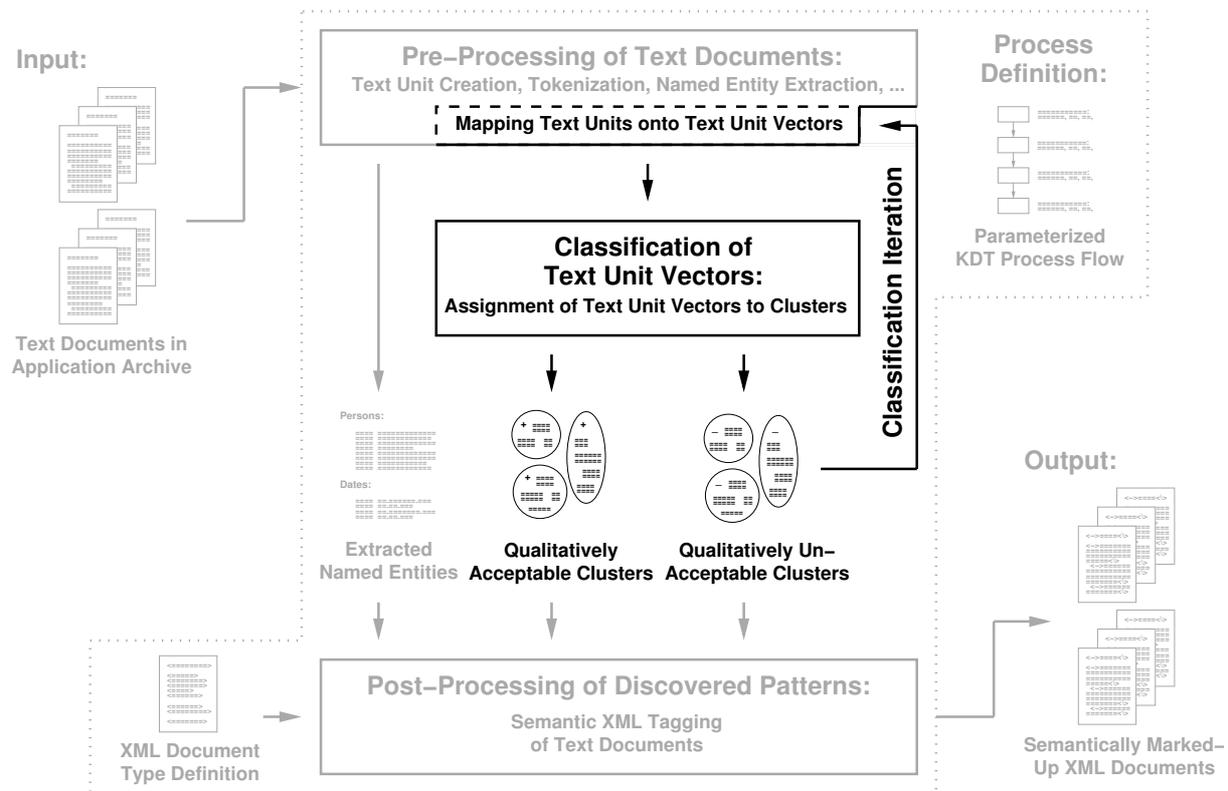
- 1: k: KdtProcessMetadata := k.create(P) // instantiate KDT process metadata
- 2:  $\bar{a}$ : IntTextArchive := DecomposeAndTokenizeTextDocuments( $\hat{a}_T$ ) // pre-process text documents
- 3:  $\bar{a}$  := ExtractNamedEntities( $\bar{a}$ , k.setOfNamedEntityTypes())
- 4:  $\bar{a}$  := LemmatizeAndDisambiguateWords( $\bar{a}$ )
- 5:  $t$ : Integer := 0 // initialize iteration counter
- 6: **repeat**
- 7:    $t := t + 1$  // increment iteration counter
- 8:   w: DescriptorWeightingScheme := w.create( $t, \cdot, \cdot, V$ ) // interactively parameterize w
- 9:   g: ClusteringAlgorithm := g.create( $t, \cdot, \cdot$ ) // interactively parameterize g
- 10:   q: ClusterQualityCriteria := q.create( $\cdot, \cdot, \cdot, \cdot, \cdot$ ) // interactively parameterize q
- 11:   m: IterationMetadata := m.create( $t, w, g, q$ ) // instantiate iteration metadata
- 12:   ( $\bar{a}, m$ ) := PerformClusteringIteration( $\bar{a}, m$ ) // execute clustering iteration
- 13:   k.appendIterationMetadata(m) // persistently store iteration metadata
- 14: **until** no qualitatively acceptable text unit cluster is found in iteration  $t$
- 15:  $\hat{s}$  := EstablishConceptualDocumentStructure( $\bar{a}, \cdot, \cdot$ ) // interactively parameterize algorithm
- 16: ( $\hat{a}_T, \hat{x}_{D_{TD}}, \hat{x}_{TD_{ocs}}$ ) := CreateSemanticallyTaggedXmlDocuments( $\bar{a}, \hat{s}$ ) // mark up text documents

---

metadata container for subsequent use in the knowledge application phase. Finally, the post-processing steps of establishing a domain-specific conceptual document structure and transforming the input text archive into a semantically marked-up text archive (cf. Section 4.4) correspond to lines 15 and 16 of Algorithm 4.9.

At the outset of this work in Section 3.1, we have introduced our framework-specific notion of KDT process flows as well as the corresponding abstract data types KdtAlgorithm (see Appendix B.19 on page 228) and KdtProcessFlow (see Appendix B.20 on page 229). At this point, we assume an existing KDT workbench that effectively supports domain and KDT experts in parameterizing and executing the generic KDT process flow listed in Algorithm 4.9. As indicated by the operation `appendKdtAlgorithm( $g_{\text{new}}$ : KdtAlgorithm)` of ADT KdtProcessFlow, a suitable workbench shall provide the necessary means of recording all KDT algorithms parameterized and executed by the KDT expert in the course of knowledge discovery. Upon completion of this interactive knowledge discovery session, the entire sequence of recorded KDT algorithms constitutes a parameterized KDT process flow, which is required for automated semantic tagging in the knowledge application phase. Prior to its first automated batch execution, however, the parameterized KDT process flow needs to be edited. Below we describe the necessary modifications that in fact bridge knowledge discovery and knowledge application within our framework. In Chapter 5, we describe the research prototype DIASDEM WORKBENCH, which fully supports the knowledge discovery process of our framework.

**Iterative Classification of Text Unit Vectors** The knowledge application phase of our framework is an effective means of automatically utilizing the domain-specific concepts



**Figure 4.4:** Iterative Classification in the DIAsDEM Knowledge Application Process

acquired beforehand to semantically annotate text documents from the same domain. As outlined in Section 3.4 and Subsection 4.3.4, each clustering iteration performed during pattern discovery corresponds to one classification iteration in the knowledge application process. Figure 4.4 highlights all components of the latter process that form the iterative classification step.

Analogous to iterative text unit clustering, each classification iteration starts with a pre-processing step that maps text units that are not yet assigned to an acceptable cluster onto real-valued text unit vectors. The iteration-specific descriptor weighting scheme required by this operation is retrieved from the corresponding iteration metadata container. Subsequently, the clustering algorithm selected, parameterized, and executed in the KDT process is run in application mode (cf. Subsection 4.3.2). Each text unit vector is assigned to one of the previously discovered text unit clusters. Again, the clustering model is retrieved from the corresponding iteration metadata item. Based on the cluster quality decisions made in the KDT phase, text units are finally associated with semantic concepts if their vectors are assigned to qualitatively acceptable text unit clusters. Semantic labels of acceptable clusters are also stored in the iteration metadata container. All remaining text units constitute the input data set for the next classification itera-

**Algorithm 4.10** PerformClassificationIteration**Input:** ( $\bar{a}$ : IntTextArchive, m: IterationMetadata)**Output:** ( $\bar{a}$ : IntTextArchive)

- 1:  $\bar{a} := m.descriptorWeightingScheme().createVectorsInApplicationMode(\bar{a})$
- 2:  $\bar{a} := m.clusteringAlgorithm().executeInApplicationMode(\bar{a})$
- 3:  $\bar{a} := m.textUnitClustering().conceptAssignmentInApplicationMode(\bar{a})$

**Algorithm 4.11** Generic KDT Process Flow for the Knowledge Application Phase**Input:** ( $\hat{a}_A$ : TextArchive,  $\hat{s}$ : ConceptualDocumentStructure, k: KdtProcessMetadata)**Output:** ( $\hat{a}_A$ : SmuTextArchive,  $\hat{x}_{DTD}$ : String,  $\hat{x}_{ADocs}$ : String[])

- 1:  $\bar{a}$ : IntTextArchive := DecomposeAndTokenizeTextDocuments( $\hat{a}_A$ ) // pre-process text documents
- 2:  $\bar{a} := ExtractNamedEntities(\bar{a}, k.setOfNamedEntityTypeTypes())$
- 3:  $\bar{a} := LemmatizeAndDisambiguateWords(\bar{a})$
- 4: **for**  $t$ : Integer := 1, 2, ..., k.numberofIterations() **do**
- 5:    $\bar{a} := PerformClassificationIteration(\bar{a}, k.iterationMetadata(t))$  // execute classification iteration  $t$
- 6: **end for**
- 7: ( $\hat{a}_A, \hat{x}_{DTD}, \hat{x}_{ADocs}$ ) := CreateSemanticallyTaggedXmlDocuments( $\bar{a}, \hat{s}$ ) // mark up text documents

tion. Since neither the cluster quality is assessed nor acceptable text unit clusters are semantically labeled, the entire iterative classification step executes without any human intervention. Consequently, iteration metadata items are not modified at all, but merely accessed in a read-only manner during the iterative text unit classification.

The abstract Algorithm 4.10 (PerformClassificationIteration) encapsulates one complete iteration of text unit classification. Taking a pre-processed, intermediate text archive and an iteration metadata container as input, this algorithm performs one classification iteration and returns an archive comprising updated intermediate text documents. Unlike its explorative counterpart, namely, Algorithm 4.5 (PerformClusteringIteration; cf. page 138), PerformClassificationIteration does not comprise any interactive operations at all. Furthermore, Algorithm 4.10 neither instantiates nor modifies iteration metadata items. Instead, this algorithm only executes operations provided by instantiations of abstract data types (e.g., the parameterized descriptor weighting scheme) that are persistently stored in the iteration metadata container passed as an input parameter.

**KDT Process Flow in Knowledge Application Phase** The entire generic KDT process flow for the second phase of our DIASDEM framework is covered by abstract Algorithm 4.11. It takes as input a typically large application text archive comprising documents that are thematically similar to the ones marked up for training purposes, the conceptual document structure derived in the knowledge discovery phase, and the metadata container instantiated therein. Upon completion of this algorithm, the automatically generated output consists of the semantically marked-up application text archive and one semantically marked-up XML document for each input text document. In addition, the concept-based XML document type definition established during knowledge discovery is output because this DTD is referenced by all XML documents. The concept-based

XML document type definition is copied into an output string by executing the operation `xmlDtd(): String`, which is provided by the instantiated conceptual document structure passed as an input parameter to Algorithm 4.11.

Unlike the generic KDT process for the knowledge discovery phase discussed above, Algorithm 4.11 does not comprise any interactive components at all. In lines 1 through 3, the application text archive is pre-processed by applying exactly the same parameterized algorithms as in the initial knowledge discovery phase. Having concluded the text pre-processing steps, an iterative text unit classification is performed (see lines 4 through 6) as introduced above. In line 7 of this algorithm, finally, all input text documents are semantically marked up to generate the required output.

Performing knowledge discovery in an appropriate workbench results in a fully parameterized KDT process flow that instantiates the abstract data type `KdtProcessFlow` (see Appendix B.20 on page 229). In addition, we assume that this workbench supports (i) the conversion of a parameterized KDT process flow recorded during knowledge discovery into a process flow for the knowledge application phase and (ii) the batch-oriented, automated execution of the resulting parameterized KDT process flow. Transforming the parameterized KDT process flow for execution during knowledge application involves deleting all interactive instantiations of abstract data types, as well as instantiations and updates of metadata items. Furthermore, each executed clustering iteration is replaced by the corresponding classification iteration. Moreover, the algorithm `EstablishConceptualDocumentStructure` is removed from the process flow. The resulting KDT process flow is fully parameterized for a completely automated, typically repeated execution within the knowledge application phase of our framework. Our prototypical DIASDEM WORKBENCH, which is described in Chapter 5, supports the entire knowledge application process.

## 4.6 Process Automation vs. Expert Involvement

As emphasized throughout this work, the DIASDEM framework consists of an intentionally semi-automated knowledge discovery phase followed by a fully automated knowledge application phase. Thereby, we deliberately accept the costs incurred by involving domain experts and their valuable knowledge in the initial phase to ultimately generate high-quality XML markup in the second, batch-oriented knowledge application phase. Table 4.32 summarizes six items of domain knowledge that are highly recommended for incorporation into our knowledge discovery process. The use of these six knowledge items has been motivated and explained in detail in the preceding sections of this chapter. In addition, Table 4.32 outlines the effect of ignoring this recommendation. Without doubt, an entirely automated knowledge discovery process would be preferable over our semi-automated approach to attaining the research questions raised in Section 1.4.

As Fayyad et al. (1996a, p. 42) put it in their seminal article, the “KDD process is interactive and iterative, involving numerous steps with many decisions made by the user.”

**Table 4.32:** Summary of Domain Knowledge Recommended for Incorporation into the DIASDEM Knowledge Discovery Process

Step in Process	KDT	Domain Knowledge to be Supplied by Human Experts	Effect of Ignoring Recommendation to Incorporate Domain Knowledge
Text unit creation		Specification of structural text units as well as selection and parameterization of an appropriate algorithm for their identification in text documents	Default structural text units (e.g., sentences) are created, as implemented by the employed workbench.
Named entity extraction		Choice of relevant, domain-specific named entity types as well as selection and parameterization of appropriate extraction algorithms	Default extraction is limited to domain-independent named entity types (e.g., dates and names of persons), as implemented by the employed workbench.
Mapping text units onto text unit vectors		Establishment of domain-specific controlled vocabulary, which shall be supported by the employed workbench	Automated feature selection techniques (cf. Subsection 4.2.4), if implemented by the employed workbench, may result in an inferior set of text unit descriptors.
Iterative text unit clustering		Selection and parameterization of clustering algorithm that most likely finds a valid, if not the true, clustering present in text unit vectors	Text unit clusters identified by the default clustering algorithm implemented by the employed workbench may not constitute a highly valid text unit clustering.
Iterative text unit clustering		Specification of appropriate DIASDEM cluster quality criteria and interactive screening of the automatically created cluster ranking	Applying default cluster quality criteria, as implemented by the workbench, may result in marking semantically inhomogeneous text unit clusters as acceptable.
Iterative text unit clustering		Interactive review of the automatically generated semantic cluster labels	Automatically derived cluster labels may not fully facilitate a meaningful, domain-specific, and usable semantic markup.

Furthermore, Fayyad et al. (p. 43) emphasized that deciding whether discovered patterns and fitted models, respectively, “reflect useful or interesting knowledge is part of the overall, interactive KDD process where subjective human judgment is typically required.” Analogously, Han and Kamber (2006, p. 31) argued that both interactive mining of knowledge and the incorporation of background knowledge are still major KDD issues. For example, Yan et al. (2004, p. 54) stressed the importance of effectively exploiting existing domain knowledge in a real-world KDD project. When applying knowledge discovery techniques to find structure in a document collection, Weiss et al. (2005, pp. 120–122) emphasized the central role of humans in reviewing, assessing, and understanding the results of the clustering process. This point of view is shared by Agrawal et al. (2000), Estivill-Castro (2002, p. 69), as well as Huang and Mitchell (2006).

By intentionally employing domain knowledge in an interactive manner, we adopt this approach to knowledge discovery. In this context, an experimental study conducted by

Wilcox and Hripcsak (2003) is particularly noteworthy. The authors analyzed the effect of expert knowledge on the inductive learning process of creating classifiers for medical texts in terms of data representation, performance of classifiers, and costs of incorporated domain knowledge. More specifically, different classification algorithms were executed with varying degrees and types of medical expert knowledge as input. According to Wilcox and Hripcsak (2003, p. 337), the experiments “showed expert knowledge to be the most significant factor affecting inductive learning performance, outweighing differences in learning algorithms.” Although conducted to assess the performance of text classifiers, these experiments corroborate our decision to actively incorporate domain knowledge for unsupervised learning as well.

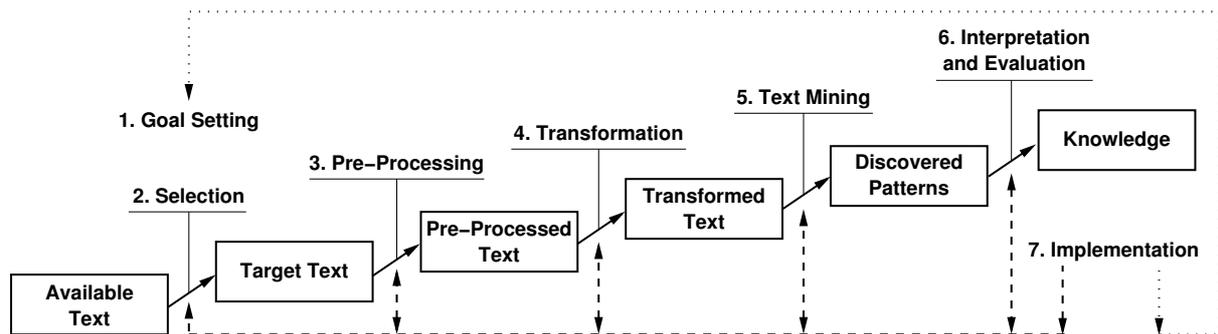
The survey of relevant literature, in particular of the existing work on automated semantic markup in Subsection 2.3.3, revealed that a fully automated, semantic XML markup of domain-specific text documents is an open research challenge. Consequently, we conjecture that a high-quality semantic XML markup of large, domain-specific text archives currently necessitates the utilization of domain knowledge provided by human experts. Future progress in complementary research areas (e.g., research into automated learning of controlled vocabularies) may gradually reduce the necessary human efforts.

Prior to applying the DIASDEM framework, however, it shall be carefully checked that the input text archive is large and potentially valuable enough to outweigh the initial knowledge discovery costs (cf. Section 3.2). After all, the value added, which is generated by transforming textual content into semantically tagged XML documents (cf. Section 1.3), shall exceed the knowledge discovery costs. Otherwise, allocating the typically scarce time of domain and KDT experts to the project cannot be justified from an economic perspective.

## 4.7 Summary

The iterative and inherently interactive DIASDEM knowledge discovery process has been specified in detail in this chapter. Our framework-specific KDT process offers a comprehensive solution to the research questions 1 through 5 posed in Section 1.4. It instantiates the generic process of knowledge discovery in textual databases illustrated in Figure 4.5 and introduced in Subsection 2.1.1. To sum up this chapter, our DIASDEM knowledge discovery process is aligned with the seven steps of the generic KDT process.

1. *Goal setting:* Our knowledge discovery process is designed to ultimately enable the automated semantic XML tagging of domain-specific text documents at the text unit level. The generic, framework-specific KDT process flow guides knowledge discovery experts when interactively preparing a fully parameterized KDT process flow for execution in the knowledge application phase of our framework.
2. *Selection:* The applicability of our KDT process is deliberately restricted to large, domain-specific text collections comprising rather homogeneous content. A natural



**Figure 4.5:** Generic Process of Knowledge Discovery in Textual Databases (Adapted and Extended from Fayyad et al., 1996b, p. 10)

clustering tendency at the text unit level is required to apply our knowledge discovery process. The potential value added of semantically marking up text archives shall be greater than the costs of incorporating human expert knowledge.

3. *Pre-processing:* Having selected an appropriate text archive, all text documents are pre-processed by decomposing them into structural text units of the chosen granularity, tokenizing text units, identifying relevant named entities, lemmatizing words, and finally performing word sense disambiguation, if necessary.
4. *Transformation:* Given a user-supplied controlled vocabulary of the focal domain, pre-processed text units are mapped onto real-valued text unit vectors. Vector components represent the weight, or importance, of text unit descriptors from the controlled vocabulary within the respective text units.
5. *Text mining:* Clustering algorithms that satisfy framework-specific selection criteria are executed to find clusters of text unit vectors, whose members represent semantically similar text units. Unlike the conventional approach to unsupervised learning, text unit vectors are repeatedly clustered with gradually reduced input data, varying parameter settings, and possibly different clustering algorithms. Thereby, both specific and general, as well as more and less frequent concepts can be discovered.
6. *Interpretation and evaluation:* In each iteration, framework-specific cluster quality criteria are applied to separate qualitatively acceptable text unit clusters, whose text units feature one coherent and frequently occurring semantic concept, from unacceptable ones. Text units assigned to qualitatively unacceptable clusters constitute the input data set for the next iteration, which in turn starts with a transformation step. Knowledge discovery is concluded by semi-automatically labeling acceptable text unit clusters and establishing a conceptual document structure.
7. *Implementation:* Our KDT process outputs classification knowledge in the form of frequently recurring semantic concepts at the text unit level. This discovered knowledge is exploited by means of transforming input text archives into semantically tagged XML documents that conform to a common, concept-based XML

document type definition. In addition, a parameterized KDT process flow is created that supports the fully automated markup of thematically similar text documents in the knowledge application phase of our framework.

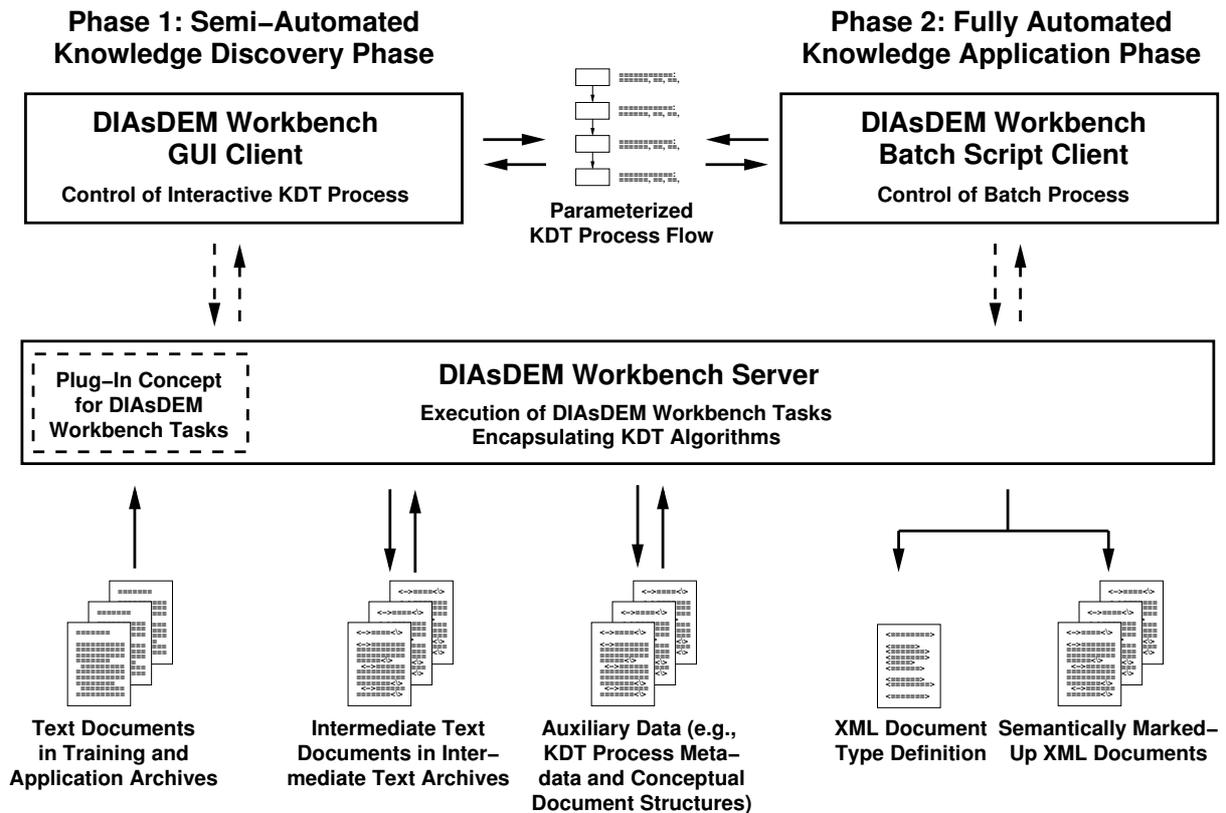
# 5 DIASDEM Workbench

Having introduced the DIASDEM framework for semantic XML tagging of domain-specific text archives in the preceding two chapters, we henceforth focus on the accompanying research prototype DIASDEM WORKBENCH. According to the systems development research methodology underlying this work (cf. Section 1.5), designing and building a research prototype is a prerequisite for the thorough evaluation of the proposed conceptual framework. Since Winkler (2007) described DIASDEM WORKBENCH in detail, we intentionally present an extended overview in this chapter only. To that end, key characteristics and the architecture of our research prototype are outlined in the next section. Subsequently, we describe the core tasks implemented in our prototype workbench in Section 5.2. Finally, this chapter is summarized in Section 5.3.

## 5.1 Key Characteristics and Architecture

This section gives an overview of the key characteristics and the high-level architecture of DIASDEM WORKBENCH. Discussing software engineering or implementation aspects in detail, however, is beyond the scope of this work. The following enumeration outlines the main features of DIASDEM WORKBENCH:

- Our research prototype supports the entire DIASDEM framework for semantic XML tagging of large, domain-specific text archives. In particular, a graphical user interface enables domain and KDT experts to interactively work in phase 1, namely, the semi-automated knowledge discovery phase. Phase 2 of our framework, which is the fully automated knowledge application phase, is additionally supported by a command-line batch client. Our central notion of KDT process flows (cf. Definition 13 on page 61) is operationalized by a workbench-specific scripting language.
- The plug-in concept of DIASDEM WORKBENCH ensures openness and flexibility. By providing a plug-in interface to our research prototype, a variety of pre-processing, pattern discovery, and post-processing algorithms can be easily incorporated into the workbench. Above all, this plug-in concept enables the economical re-use of existing, Java-based implementations of algorithms, such as the ones contained in the WEKA machine learning library (cf. Witten and Frank, 2005).
- DIASDEM WORKBENCH is implemented in Java. This object-oriented programming language facilitates the implementation of our framework-specific abstract data types introduced in the preceding two chapters and defined in Appendix B.

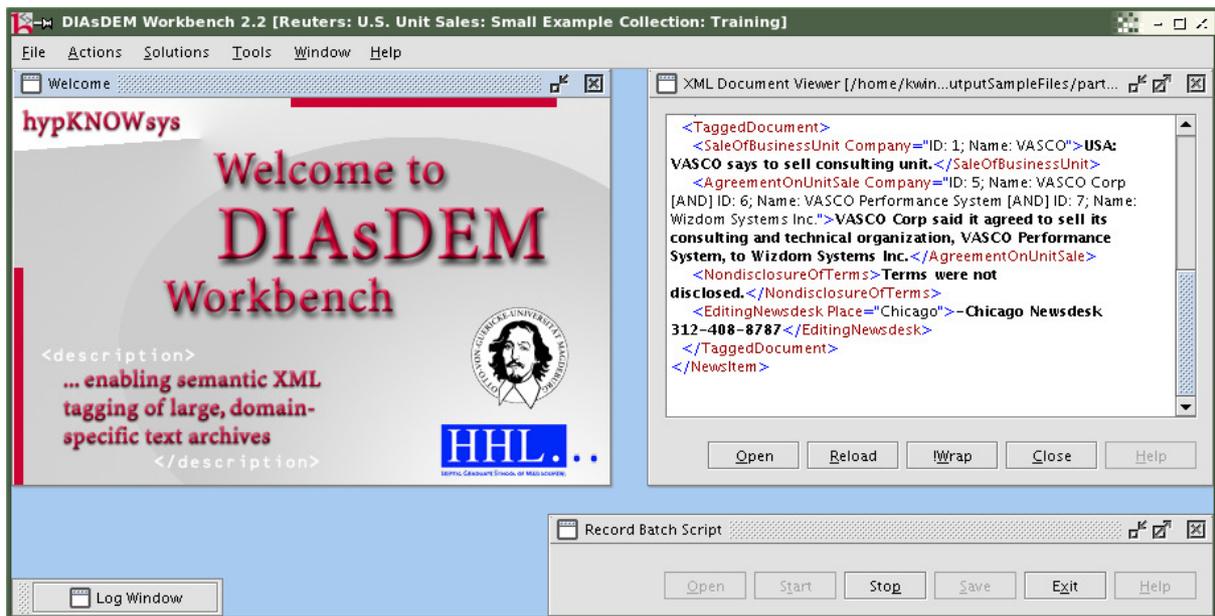


**Figure 5.1:** Architectural Overview of DIASDEM WORKBENCH

- The research prototype, which was designed and predominantly implemented by the author, is open source software. It is available for download via the Web site <http://ka.rsten-winkler.de/thesis> under the GNU general public license.

**Architecture** Figure 5.1 illustrates the high-level architecture of DIASDEM WORKBENCH, which is a Java-based client/server-application. Solid arrows represent the data flow whereas dashed arrows indicate the control flow in this figure. Controlled by DIASDEM WORKBENCH GUI CLIENT or DIASDEM WORKBENCH BATCH SCRIPT CLIENT, DIASDEM WORKBENCH SERVER executes so-called DIASDEM WORKBENCH tasks, each of which encapsulates one dedicated KDT algorithm or a group of related KDT algorithms. Instances of both clients (i) trigger the execution of tasks by submitting parameterized requests, (ii) monitor the progress of task execution, and (iii) finally request metadata describing the results from the respective server instance. By implementing a set of Java interfaces that realize the plug-in concept, new or existing algorithms can be easily converted into DIASDEM WORKBENCH tasks.

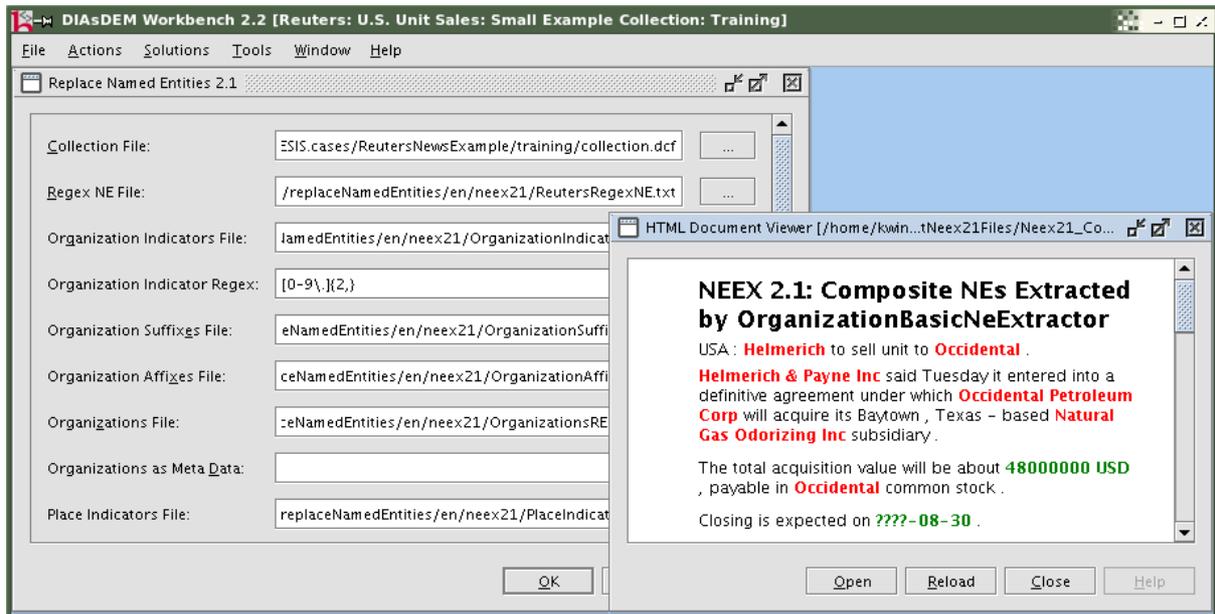
During knowledge discovery, the KDT expert initially creates a new KDT process flow. Subsequently, this process flow is parameterized step-by-step by means of recording all



**Figure 5.2:** Screen Shot of DIASDEM WORKBENCH GUI CLIENT

KDT tasks that are interactively submitted to DIASDEM WORKBENCH SERVER by the expert. For each task, the knowledge discovery specialist inputs required and optional parameters using a graphical user interface prior to submitting the parameterized task for execution. Once a task is completely executed, a result metadata container is returned by the server and displayed for inspection by domain and KDT experts in DIASDEM WORKBENCH GUI CLIENT. Existing KDT process flows can be modified, if necessary, using DIASDEM WORKBENCH GUI CLIENT. KDT process flows are submitted for execution by any client application in the knowledge application phase of our framework. Typically, however, the command-line batch script client is preferred during knowledge application because this client allows for an automated, even scheduled conversion of text documents into semantically marked-up XML documents without any human intervention at all. If DIASDEM WORKBENCH SERVER executes a batch script in the non-interactive way, result metadata are not immediately visualized by the respective client application, but stored in the KDT process flow for later inspection. Nevertheless, human experts can examine the results of any automatically executed process flow by opening the corresponding file in DIASDEM WORKBENCH GUI CLIENT. When processing KDT tasks, DIASDEM WORKBENCH SERVER reads input text archives, stores intermediate text archives and auxiliary data (e.g., KDT process metadata and conceptual document structures) in XML files, and finally outputs semantically marked-up XML documents along with a concept-based XML document type definition.

Figure 5.2 depicts the graphical user interface of DIASDEM WORKBENCH GUI CLIENT. Using this client application, knowledge discovery experts manage the entire

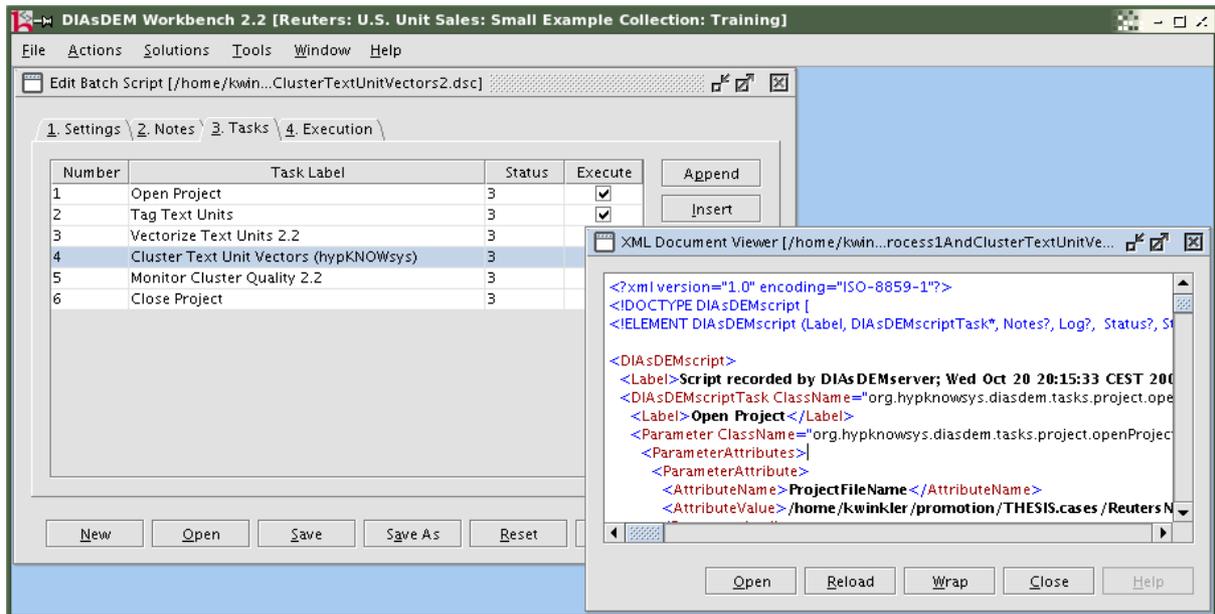


**Figure 5.3:** Screen Shot of the REPLACE NAMED ENTITIES 2.1 Task of DIASDEM WORKBENCH GUI CLIENT

knowledge discovery phase by setting up new KDT projects, importing raw text documents, establishing a domain-specific controlled vocabulary, executing the necessary KDT tasks and recording them in a batch script, assessing the results of KDT tasks (e.g., the quality of semantic markup), and eventually exporting a domain-specific, concept-based XML document type definition and semantically marked-up XML documents. For example, Figure 5.3 shows DIASDEM WORKBENCH GUI CLIENT during the activity of parameterizing the task REPLACE NAMED ENTITIES 2.1. In addition, this figure exemplifies the visualization of results as it also depicts a panel that highlights identified named entities. A recorded KDT process flow and batch script, respectively, can be edited using the BATCH SCRIPT EDITOR tool of DIASDEM WORKBENCH GUI CLIENT, which is depicted in Figure 5.4. This figure also shows that batch scripts are XML documents, which conform to a workbench-specific XML document type definition (see Winkler, 2007, p. 91).

## 5.2 Overview of Core Tasks

In this section, we concisely describe fundamental DIASDEM WORKBENCH TASKS, which are henceforth synonymously referred to as KDT tasks or tasks in shortened form. To that end, essential tasks supported by the research prototype are categorized by the three main phases of our knowledge discovery process (cf. Chapter 4).



**Figure 5.4:** Screen Shot of the BATCH SCRIPT EDITOR Tool of DIASDEM WORKBENCH GUI CLIENT

### 5.2.1 Pre-Processing of Text Documents

The KDT tasks outlined in this subsection correspond to the text pre-processing steps of our knowledge discovery process (cf. Section 4.2). The task that maps text units onto text unit vectors is described in the next subsection on iterative clustering for the reasons discussed in Subsection 4.2.5 and in Section 4.5.

**Text Document Import** The task IMPORT PLAIN TEXT FILES reads text files from a user-specified directory and converts them into a text archive according to Definition 3 on page 58. Technically, text files are imported into an intermediate data representation, the so-called DIASDEM collection, that comprises XML documents conforming to a workbench-specific XML DTD (see Winkler, 2007, p. 90). Each DIASDEM collection is identified by a collection file containing archive metadata and references to intermediate XML documents that constitute the collection. Hence, a new collection must be created using the auxiliary task CREATE DOCUMENT COLLECTION (see Winkler, 2007, pp. 18–19) prior to importing text files into a collection for subsequent processing. The storage is not limited to a file-based representation. Storing a collection, for example, in an XML-enabled database requires the implementation of storage-related Java interfaces.

**Creation of Text Units** As described in Section 4.2.1, the KDT task CREATE TEXT UNITS transforms the initial text archive into an intermediate text archive (cf. Defini-

tion 21 on page 75) by identifying and separating text units. The algorithm HEURISTIC SENTENCE IDENTIFIER initially replaces full stops in abbreviations (e.g., "e.g."), which are listed in the parameter file *Abbreviations File*, with asterisks. Thereafter, regular expressions listed in the parameter file *Full Stop Regex File* are matched against the text. These regular expressions match full stops that are no sentence boundaries (e.g., "01.01.2007") and replace all matches with asterisks. All remaining punctuation marks are heuristically considered to be sentence boundaries that separate text units from each other. Thereby, text documents are split into sentence-based text units.

In general, our framework comprises the notion of structural views on texts, which are referred to as text unit layers (cf. Definition 5 on page 59). For example, one might define three text unit layers to semantically annotate each document at the text level, at the paragraph level, and at the sentence level. However, DIASDEM WORKBENCH 2.2 does not support the hierarchical semantic tagging of text documents. Instead, KDT tasks always process text units that are associated with one user-specified text unit layer (e.g., the sentence level) and output XML documents that are marked up accordingly.

**Tokenization of Text Units** After importing texts and creating text units, tokenizing them using the task TOKENIZE TEXT UNITS constitutes the third pre-processing step (see Subsection 4.2.1). During tokenization, text units are decomposed into individual words and tokens, respectively. In addition, text units are normalized to map, for example, date literals appearing in many different formats (e.g., "1 Jan 2007" and "1.1.2007") onto a canonical representation (e.g., "01.01.2007"). Moreover, multi-token terms that contain blank spaces, such as "for example", are identified to subsequently process them as single tokens.

Firstly, regular expressions listed in the parameter file *Tokenization Regex File* are matched against each processed text unit. For instance, the character subsequence "e." of the string "This is a sentence. There" is matched by the regular expression  $(\S)(\.|\\!|\\?)$ . This matching character subsequence is thus substituted by the accompanying replacement string  $\$1\ \$2$ , which results in the following tokenized text: "This is a sentence . There". Secondly, regular expressions listed in the parameter file *Normalization Regex File* are matched against processed text units. Analogously, matching character subsequences are substituted by the corresponding replacement string. Thirdly, multi-token terms included in the parameter file *Multi-Token Words File* are looked up in text units. Each identified multi-token term is reduced to one single token by replacing blank spaces with underscores, such as "for\_example".

**Extraction of Named Entities** After importing text files, creating and tokenizing text units, identifying named entities and replacing them with placeholders constitutes the fourth pre-processing step, as described in Subsection 4.2.2. Extracted named entities potentially serve as attribute values in semantic XML tags. The task REPLACE NAMED ENTITIES 2.1 can be fully parameterized by editing a large number of parameter files. The heuristic, rule-based named entity extractor is inspired by Feldman et al. (1999), Agichtein

and Gravano (2000), Brüninghaus and Ashley (2001), Volk and Clematide (2001), as well as Sekine et al. (2002). The prototypical extractor works as follows:

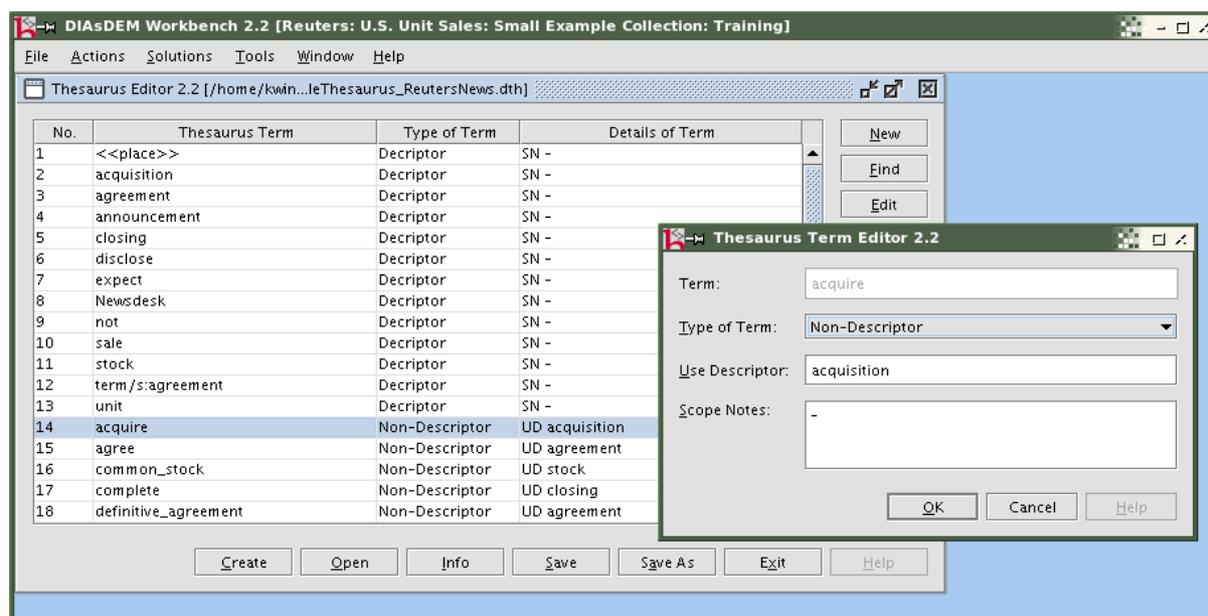
1. Firstly, regular expressions listed in the parameter file *Regex NE File* are matched against intermediate text units to identify instances of the following named entity types, which are referred to as basic named entities: number, date, time, amount\_of\_money, paragraph, email, url, organization\_id, document\_id, court, postal\_code, reference\_number, percentage, newspaper, stock\_exchange, isin (i.e., international securities identification number), wkn (i.e., German securities identification number), number\_of\_shares, and amount\_of\_money\_per\_share.
2. Secondly, instances of the basic named entity type organization are identified by employing the parameter files *Organization Indicators File*, *Organization Suffixes File*, *Organization Affixes File*, and *Organizations File*. The latter file contains a list of complete, tokenized names of important organizations that are extracted in any case. To heuristically extract less important organization names, the algorithm initially searches for known organization name suffix tokens (e.g., "Ltd.") and then looks backwards for valid organization indicator tokens, such as "takeover of", to instantiate a named entity of type organization. Subsequently, identified organization names are extended if they are followed by organization name affix tokens (e.g., "Worldwide") listed in the respective parameter file.
3. Thirdly, tokens or sequences thereof instantiating the basic named entity type place are extracted using the parameter files *Place Indicators File*, *Places File*, and *Place Affixes File*. The algorithm first looks up all tokens in the dictionary of known places comprising, for example, countries like "Germany" and cities like "Berlin". Subsequently, place candidate tokens are extended if they are immediately followed by another known place or a place affix token, such as "Airport". However, even place candidates are only instantiated as named entities of type place if they are directly preceded by a known place indicator token, such as "in" or "to".
4. Thereafter, the parameter files *Person Name Indicators File*, *Titles File*, *Forenames File*, *Middle Initials File*, *Surnames File*, *Surname Suffixes File*, and *Name Affixes File* are used to discover instances of basic named entity type person\_name. Each instance corresponds to a contiguous sequence of tokens, each of which instantiates one of the following basic named entity types: academic title (e.g., "Dr."), forename, middle\_initial, surname, or name\_affix (e.g., "Sen."). Composite forenames and surnames comprising a hyphen are also identified. Person name candidates are extended if they are immediately followed by a known name affix token or a capitalized token ending with a known surname suffix listed in *Surname Suffixes File*. However, single-token person name candidates are only instantiated if they are directly preceded by a person name indicator token contained in the respective parameter file. If the optional parameter *Professions File* is specified, instances of basic named entity type profession are identified in text units that contain at least one instance of named entity type person\_name.

5. Constructor rules, which are specified in a workbench-specific syntax in the parameter file *Composite NE File*, are finally applied to intermediate text units that contain identified basic named entities to discover instances of the following composite named entities: `person`, `company`, `company_relocation`, `date_period`, `equity_stake`, `amount_of_money_range`, `percentage_range`, `unit_of_company`, and `key_figure`. Each composite named entity consists of basic named entities that occur in the context of tokens defined by constructor rules. For instance, composite named entities of type `person` can be constructed from the basic named entities `person_name`, `date`, and `place`. If a composite named entity is identified, both ordinary tokens and basic named entity placeholders matched by the rule are substituted by the corresponding composite named entity placeholder. Adding the exemplary rule `<<person_name>> geb. <<person_name>> , <<date>> , <<place>>` to match tokens and instances of basic named entities along with an appropriate constructor statement to *Composite NE File* maps, for instance, the German token sequence "Marion Marcella Adolph geb. Priester , 22.03.1957 , Offenbach" onto an instance of the composite named entity type `person`.

In general, each token might simultaneously instantiate various basic named entities. For instance, the token "Hagen" could be a German forename or a German city. Thus, various heuristics are used in conjunction with the parameter files *Person Name Indicators File* and *Place Indicators File* to decide whether a particular token probably instantiates a place or is more likely a partial `person_name`.

**Lemmatization of Text Units** In accordance with Subsection 4.2.3, the prototypical task `LEMMATIZE TEXT UNITS` determines grammatical roots of terms (i.e., their lemma forms) and subsequently replaces terms in intermediate text units with their respective lemma forms. For example, inflected verb forms (e.g., "went") are mapped onto their infinite forms (e.g., "go"). This task employs `TREETAGGER` to automatically determine lemma forms. `TREETAGGER` is a multi-lingual part-of-speech tagger developed by Schmid (1994, 1999), which can be used free of charge for research purposes. More specifically, intermediate text units are exported in the file format required by `TREETAGGER`, the part-of-speech tagger is afterwards executed to annotate the generated text file, and the results are finally imported to update the intermediate text archive accordingly. The KDT expert is able to parameterize the temporary input file name, the temporary output file name, and the `TREETAGGER` command to be executed.

**Word Sense Disambiguation** The prototypical task `DISAMBIGUATE WORD SENSES` heuristically determines the sense of ambiguous words (cf. Subsection 4.2.3). Its embedded algorithm is inspired by the method referred to as disambiguation based on sense definitions (cf. Manning and Schütze, 1999, pp. 242-244). Potentially ambiguous tokens, such as "interest" in the mergers and acquisitions domain, are heuristically disambiguated based on the occurrence of contextual sense indicator tokens in the same intermediate



**Figure 5.5:** Screen Shot of the THESAURUS EDITOR Tool of DIASDEM WORKBENCH GUI CLIENT

text unit or in an even smaller, user-specified set of neighboring tokens. For example, the sense identifier `/s:stake` is appended to the token `interest` to form the new token `interest/s:stake` if at least one of the lemmatized sense indicator tokens `acquisition`, `acquire`, or `merge` occurs in the same intermediate text unit. Hence, sense indicator tokens serve as concise sense definitions for their associated token. If additional indicator tokens were defined for the relevant senses `/s:involvement` and `/s:borrowing` of the token `interest`, a voting schema would determine the most likely sense of the ambiguous token. Placeholders for named entity types may serve as sense indicator tokens as well. All tokens to be disambiguated along with the respective senses and their indicator terms are specified in the *Word Senses File*.

**Establishment of the Controlled Vocabulary** Before applying our framework to a new domain, an appropriate controlled vocabulary is semi-automatically created in the pre-processing phase of our knowledge discovery process (see Subsection 4.2.4). To that end, the task `ESTABLISH INITIAL THESAURUS` creates a new thesaurus on the basis of term frequency statistics computed by the auxiliary task `COMPUTE TERM FREQUENCY STATISTICS`. Terms are inserted into the new thesaurus as descriptor terms if their absolute frequency is greater than or equal to a minimum threshold and less than or equal to a maximum threshold. After establishing an initial thesaurus, the domain expert refines this thesaurus by using the `THESAURUS EDITOR` tool depicted in Figure 5.5. In particular, semantically less important terms are deleted from the thesaurus. Furthermore,

the domain expert takes into account the semantics of terms and concepts by defining relations between less important non-descriptors and associated descriptors.

### 5.2.2 Iterative Clustering of Text Unit Vectors

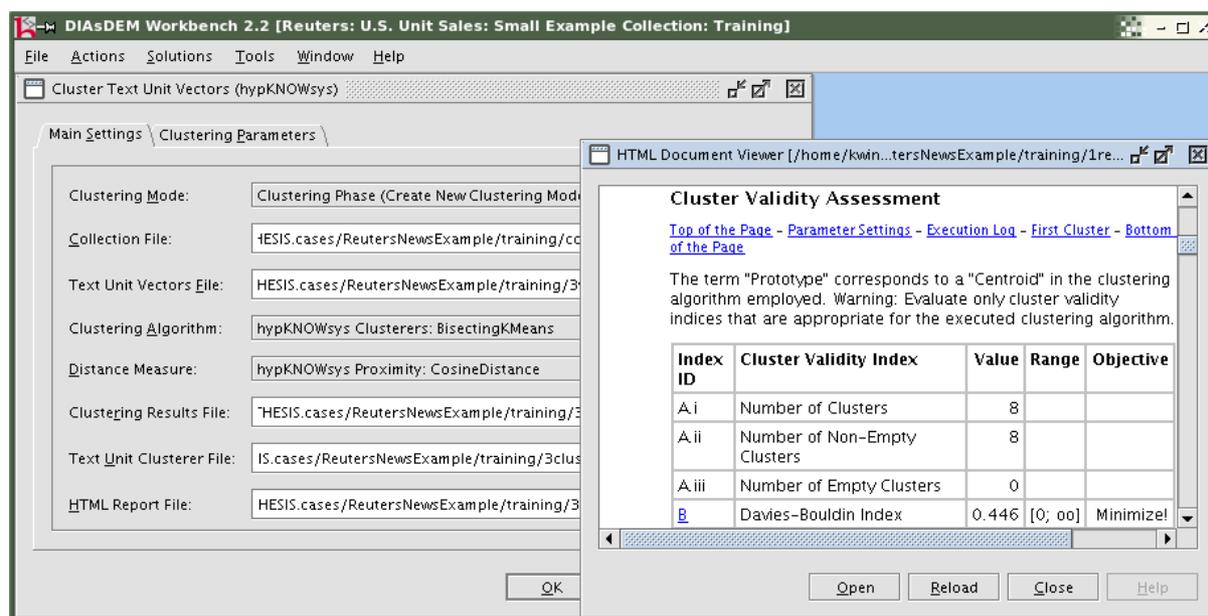
This subsection describes tasks of DIASDEM WORKBENCH that are executed during each iteration of the knowledge discovery phase (cf. Subsection 4.3.4). A few tasks are executed in the knowledge application phase of our framework as well (cf. Section 4.5).

**Vectorization of Text Units** The task VECTORIZE TEXT UNITS 2.2 transforms text units into text unit vectors (cf. Subsection 4.2.5). To that end, the KDT expert specifies the *KDT Process Iteration* number, name and type of the *Text Unit Vectors File* to be exported for consumption by the clustering task, and a domain-specific *Thesaurus File*. A weighting scheme is chosen and parameterized during knowledge discovery. The default weighting scheme, however, corresponds to Expression 4.1 on page 96 with a Boolean term frequency component, an inverse text unit frequency as the collection frequency component, and no vector length normalization component at all. As explained in Subsection 4.2.5, the iteration-specific collection frequency components of descriptor weights are computed once in the interactive KDT phase and merely used in the application phase. During knowledge discovery, computed collection frequencies are thus stored in the user-specified *Collection Frequencies File* as part of the iteration metadata. Collection frequencies are retrieved from this file during knowledge application.

**Selecting, Parameterizing, and Executing a Clustering Algorithm** In accordance with Subsection 4.3.4, the task CLUSTER TEXT UNIT VECTORS (HYPKNOWSYS<sup>1</sup>) allows the KDT expert to select an appropriate clustering algorithm, parameterize it, and execute the algorithm both in discovery and application mode. This task utilizes and extends the Java-based data mining library WEKA (cf. Witten and Frank, 2005). Along with various data pre-processing and machine learning algorithms, WEKA includes three clustering algorithms (i.e., *k*-means, Cobweb, and EM). Due to their poor performance on large data sets in discovery mode, the missing support of common cluster validity indices, and the necessity to process data sets inside the main memory in application mode, however, the author decided to implement the algorithms discussed in Subsection 4.3.2 within the WEKA framework. The bisecting *k*-means algorithm (Steinbach et al., 2000), the BATCH MAP algorithm, (Kohonen, 2001, pp. 139–140), and the SNN clustering algorithm (Ertöz et al., 2004) satisfy most requirements for usage in our framework and simultaneously represent three distinct approaches to clustering textual data. All algorithms are executable in application mode by means of utilizing the Java-inherent object persistence mechanism. For each clustering algorithm, the necessary parameters are entered by the

---

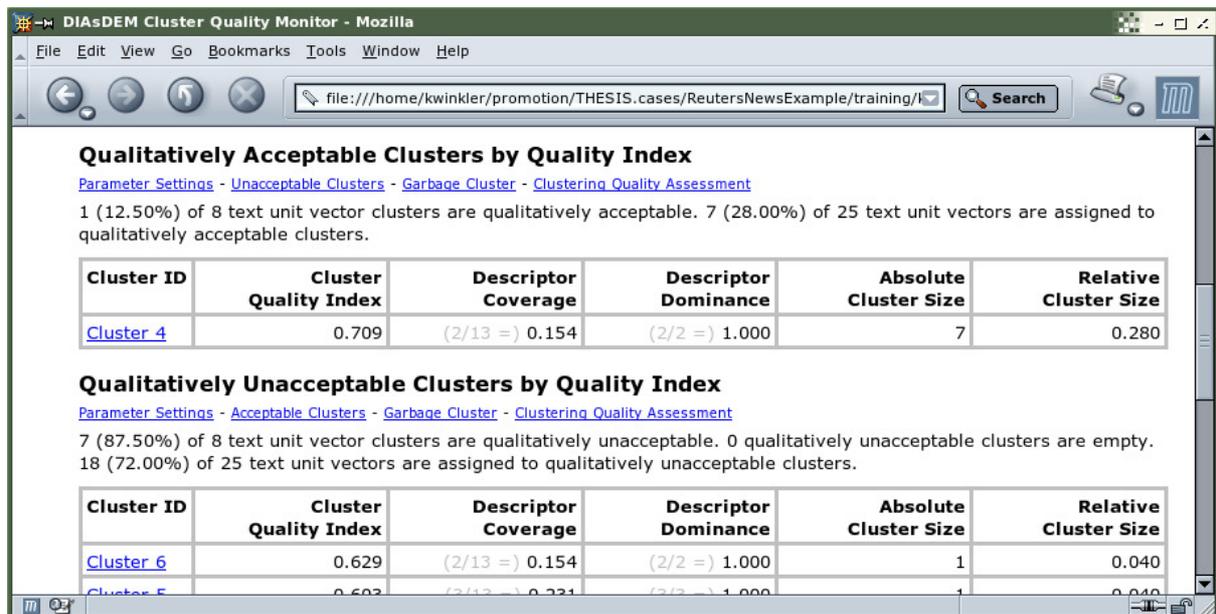
<sup>1</sup>The open source software project hypKNOWsys hosts the development of DIASDEM WORKBENCH.



**Figure 5.6:** Screen Shot of the CLUSTER TEXT UNIT VECTORS (HYPKNOWSYS) Task of DIASDEM WORKBENCH GUI CLIENT

knowledge discovery expert. In addition, the task CLUSTER TEXT UNIT VECTORS (HYPKNOWSYS) encapsulates all proximity measures discussed in Subsection 4.3.1 such that the KDT expert is able to select an appropriate combination of clustering algorithm and proximity measure. Furthermore, this task computes the five common relative cluster validity indices listed in Table 4.12 on page 106. Figure 5.6 illustrates the user interface of this task and the visualization of results.

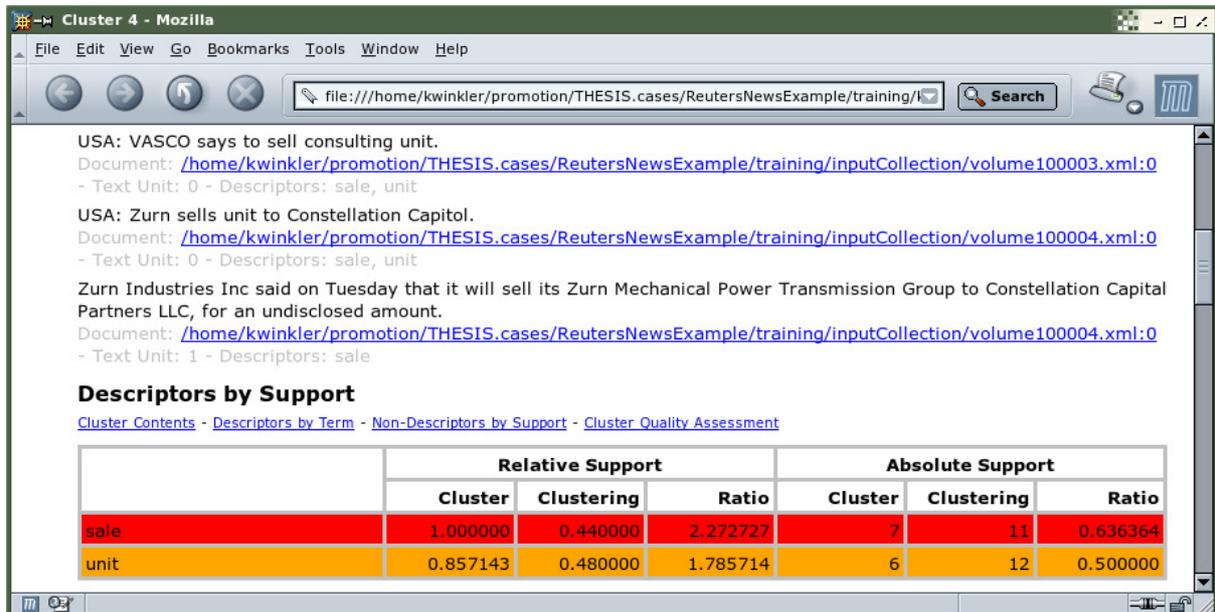
**Automatic Ranking of Text Unit Clusters and Automatic Default Labeling of Acceptable Clusters** Subsequent to clustering text unit vectors during knowledge discovery, the task MONITOR CLUSTER QUALITY 2.2 automatically ranks the resulting clusters by decreasing quality (cf. Subsection 4.3.3) and additionally generates default labels of all qualitatively acceptable clusters (cf. Subsection 4.4.1). For performance reasons, this task executes the operations in line 4 (i.e., automatic creation of cluster ranking) and in line 6 (i.e., automated labeling of acceptable clusters) of Algorithm 4.5 (PerformClusteringIteration) on page 138 in one pass over the intermediate text archive. Thereby, clusters that are considered to be acceptable in the first place are semantically labeled by default. This task is not executed during the knowledge application phase of our framework. Prior to executing this task, the KDT expert inputs, among other parameters, the name of the *Cluster Label File* that is part of the iteration metadata and contains all generated semantic cluster labels. The cluster quality parameters explained in Subsection 4.3.3 are set by the KDT specialist to define the minimum requirements to be met by qualitatively acceptable clus-



**Figure 5.7:** Visualization of Text Unit Clustering Created by the MONITOR CLUSTER QUALITY 2.2 Task of DIASDEM WORKBENCH GUI CLIENT

ters. Besides outputting a quality assessment for each text unit cluster and generating a *Cluster Label File* comprising default labels for acceptable clusters, the task MONITOR CLUSTER QUALITY 2.2 creates HTML-based visualizations of the entire clustering and each individual cluster. Figures 5.7 and 5.8 depict automatically generated visualizations, which enable a quick and intuitive content characterization while interactively reviewing cluster labels.

**Interactive Screening of Cluster Ranking and Interactive Review of Default Cluster Labels** During knowledge discovery, automatically generated cluster quality assessments and default cluster labels can be interactively reviewed and modified, if necessary. To that end, the domain expert employs the CLUSTER LABEL EDITOR tool of DIASDEM WORKBENCH. As discussed in Subsection 4.4.1, the domain specialist is asked to inspect text units assigned to a cluster, or a sample thereof, in combination with cluster-specific frequency statistics of both text unit descriptors and terms not covered by the controlled vocabulary. Supported by cluster visualizations created by the task MONITOR CLUSTER QUALITY 2.2, default cluster labels can be efficiently corrected. Furthermore, the CLUSTER LABEL EDITOR tool enables the human expert to interactively screen the cluster ranking and, if necessary, to manually override automatically created cluster quality assessments (cf. Subsection 4.3.4). The CLUSTER LABEL EDITOR tool is not used in the knowledge application phase of our framework.



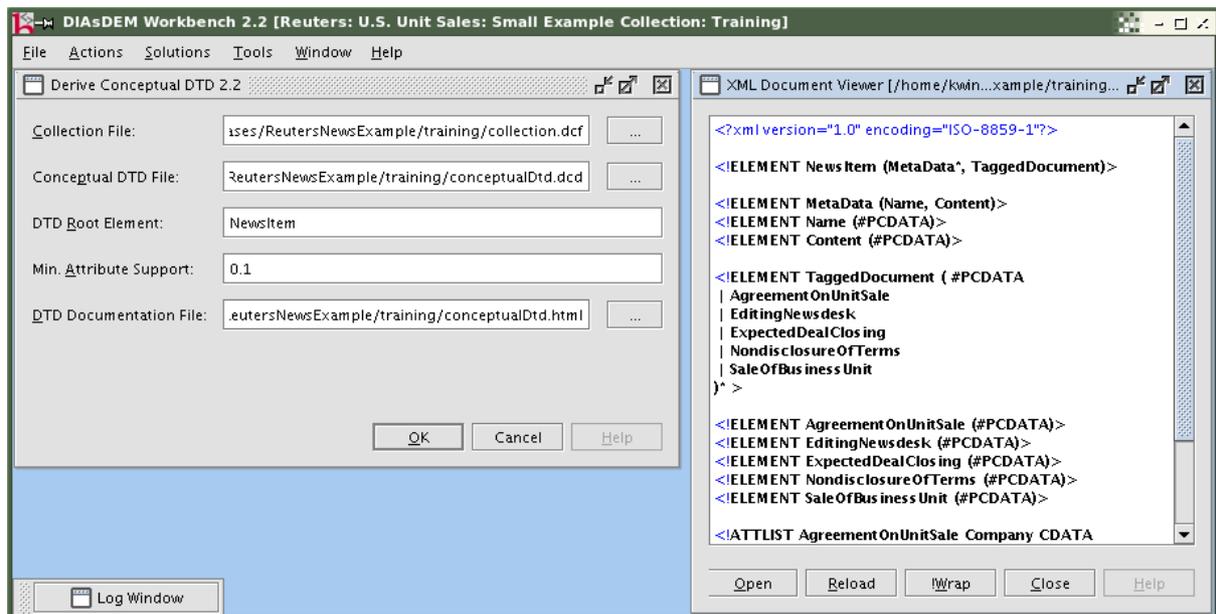
**Figure 5.8:** Visualization of Text Unit Cluster Created by the MONITOR CLUSTER QUALITY 2.2 Task of DIASDEM WORKBENCH GUI CLIENT

**Update of the Intermediate Text Archive** At the end of each clustering and classification iteration, the task TAG TEXT UNITS is executed to persistently store the results in the intermediate text archive. This task updates the iteration identifier, the cluster identifier, as well as the assigned semantic cluster label and concept, respectively, of all intermediate text units that have not yet been assigned to a qualitatively acceptable cluster according to the cluster assignments at the beginning of the current iteration.

### 5.2.3 Post-Processing of Discovered Patterns

This subsection describes two core KDT tasks that post-process discovered patterns.

**Establishment of the Concept-Based XML DTD** Subsequent to the final clustering iteration of the knowledge discovery process, the DIASDEM WORKBENCH task DERIVE CONCEPTUAL DTD 2.2 establishes a concept-based XML document type definition in accordance with Subsection 4.4.2 on page 138. Prior to executing this task, the KDT expert specifies the name of the DTD root element, the required minimum attribute support, as well as the name of the *Conceptual DTD File* and the HTML-based *DTD Documentation File* to be generated. Once a new concept-based XML DTD is generated, the metadata file referred to as *Conceptual DTD File* comprises the domain-specific conceptual document structure, which is utilized when creating semantically marked-up XML documents during both knowledge discovery and knowledge application. The new concept-based



**Figure 5.9:** Screen Shot of the DERIVE CONCEPTUAL DTD 2.2 Task of DIAsDEM WORKBENCH GUI CLIENT

XML document type definition (e.g., *NewsItem.dtd*) is named in correspondence with its root element (e.g., *NewsItem*). Furthermore, the automatically generated DTD documentation template contains an enumeration of valid attributes and five exemplary text units for each DTD element. Figure 5.9 depicts a screen shot of the task DERIVE CONCEPTUAL DTD 2.2 along with a generated XML DTD. The task is not executed in the knowledge application phase.

**Semantic XML Tagging of Text Documents** The final step in both knowledge discovery and knowledge application, namely, the semantic XML tagging of text documents, is performed by the task TAG DOCUMENTS 2.2 (cf. Subsection 4.4.3 on page 143). Given a *Conceptual DTD File* comprising the domain-specific conceptual document structure established by the task DERIVE CONCEPTUAL DTD 2.2, the entire original text archive is converted into semantically marked-up XML documents conforming to the corresponding concept-based XML document type definition. In addition, the supplementary *Random Sample File* contains a random sample of text units for the subsequent evaluation of markup quality (cf. Section 6.1) as a result of executing this task. The random sample size is specified by the knowledge discovery expert.

## 5.3 Summary

The open source research prototype DIASDEM WORKBENCH has been described in this chapter. Our workbench supports the entire DIASDEM framework for semantic tagging of domain-specific text archives: the interactive knowledge discovery phase and the automated, batch-oriented knowledge application phase. Our notion of KDT process flows as a link between both phases is operationalized by a workbench-specific scripting language. The Java-based client/server-application comprises a server component that executes DIASDEM WORKBENCH tasks as requested by client applications. Each task encapsulates a KDT algorithm or a group of similar ones. The implemented plug-in concept for tasks ensures openness, flexibility, and re-usability of existing implementations of algorithms. The graphical user interface client application facilitates the iterative, interactive knowledge discovery process and simultaneously records submitted tasks in editable batch scripts. In addition, the batch script client is capable of automatically executing batch scripts during knowledge application. Consequently, DIASDEM WORKBENCH enables the thorough evaluation of our conceptual framework as part of the systems development research methodology underlying this work.



## 6 Experimental Evaluation

The research methodology underlying this work, systems development as proposed by Nunamaker et al. (1991), has been summarized in Section 1.5. Following the introduction of our conceptual framework for semantic XML tagging and the description of our research prototype in the preceding three chapters, we henceforth elaborate on the experimental evaluation of the DIASDEM framework in the experimentation/observation phase of the generic systems development research process. In the next section, we describe criteria designed to measure the quality of semantic XML markup. Section 6.2 presents the results of using DIASDEM WORKBENCH to semantically enhance two real-word text archives from the competitive intelligence domain. This chapter is summarized in Section 6.3.

### 6.1 Assessing the Quality of Semantic XML Markup

“For prediction problems,” Weiss et al. (2005, p. 123) emphasized in the context of knowledge discovery in textual databases, “evaluation is straightforward.” Given a class-labeled text archive, classifier performance is in principle measured by classifying documents not utilized in the training procedure and comparing the predicted class memberships with their true, a priori known classes. This standard approach to assessing the quality of supervised learning techniques (e.g., cf. Vazirgiannis et al., 2003, pp. 76–82), however, cannot be employed in the DIASDEM framework because unsupervised learning, or clustering, is employed to semantically annotate plain, a priori unlabeled texts.

In information retrieval research, the evaluation of retrieval systems is also an important issue (e.g., see Baeza-Yates and Ribeiro-Neto, 1999, pp. 73–97). The notion of retrieval performance evaluation refers to assessing the relevance of documents retrieved in response to a user’s query. To measure retrieval effectiveness, recall and precision, along with measures derived thereof, are conventionally employed (cf. Grossman and Frieder, 2004, pp. 2–5). Baeza-Yates and Ribeiro-Neto emphasized that assessing retrieval performance typically involves experiments performed in laboratories because the set of truly relevant documents for certain queries is needed in advance. Since costly resources for creating topic-labeled test collections are mostly scarce, research into information retrieval tends to resort to using carefully crafted reference corpora whose texts are already assigned topic identifiers (cf. Baeza-Yates and Ribeiro-Neto, 1999, pp. 84–96). For example, Agrawal et al. (2000), Steinbach et al. (2000), and Weiss et al. (2005, p. 171–178) utilized standard IR collections when addressing knowledge discovery tasks. As the discussion on descriptor weighting in Subsection 4.2.5 has revealed, standard IR collections along with precision and recall measures cannot be used for quality evaluation in our framework because we

strive for exploratively discovering new, a priori unknown semantic concepts at the text unit level. Due to our unsupervised learning approach, an automated computation of recall or precision measures necessitates the manual mark-up of plain texts in conformance with the established concept-based XML DTD.

Without doubt, we cannot assume the availability of costly human markup resources to annotate the entire training collection. Unlike standardized studies in measuring the effectiveness of information retrieval techniques, we cannot leverage common IR reference collections as they are not marked-up at the level of structural text units. Analogous to Castellanos (2004, p. 154), human domain expertise is hence essential in evaluating the quality of semantic XML markup in our framework. Therefore, we suggest drawing a random sample of semantically marked-up text units and ask a domain expert to assess the markup quality, as defined in detail in the next subsection. By limiting the quality evaluation to a random sample of processed text units, we reduce human involvement on the one hand and ensure statistically valid statements concerning markup quality on the other. Requirements for effective software support of domain experts, who perform the quality assessment, are briefly described in Subsection 6.1.2.

In particular, we acknowledge the “inherent degree of psychological subjectiveness” associated with the task of quality evaluation as noticed, for example, by Baeza-Yates and Ribeiro-Neto (1999, p. 84). For instance, the results of experiments on human performance on a clustering task conducted by Macskassy et al. (1998, p. 264) provided a “sobering note on any quest for a single clearly correct clustering method for web pages.” When conducting experiments on marking up resources for the Semantic Web (cf. Section 1.3), Forno (2003) observed that human taggers often assigned different descriptive tags from a finite set to the same resource. In this work, we assume that the markup quality is assessed by exactly one domain expert. Thereby we avoid inconsistencies introduced by asking several humans to evaluate the same processed text unit. Furthermore, we abstract from possible inconsistencies introduced by different assessments of the same processed text unit by one domain expert. These assumptions can be relaxed by incorporating social science techniques designed to measure the so-called inter-tagger agreement (cf. Fellbaum, 1998, pp. 217–237), as well as the so-called inter-coder and intra-coder reliability (cf. Bayerl et al., 2003b).

### 6.1.1 Quality Criteria for Semantic XML Markup

The quality assessment of semantic XML markup involves detecting two distinct issues. Firstly, the derived concept-based XML document type definition might not be appropriate to describe the semantic content of the domain under scrutiny. Secondly, marked-up text units may be erroneously annotated. In this work, however, we focus solely on the evaluation of marked-up content with respect to the latter error. Since our DIASDEM framework intentionally incorporates human domain knowledge (cf. Section 4.6), it is reasonable to assume that a derived concept-based XML document type definition appropriately reflects the most important, frequently recurring concepts at the text unit level.

After all, labels of qualitatively acceptable text unit clusters and names of XML tags, respectively, are semi-automatically assigned and interactively reviewed. Consequently, we advocate a quality assessment from the perspective of users of semantically marked-up XML documents who take the corresponding DTD as given. The distinction between correct and erroneous markup therefore solely requires comparing a random sample of automatically generated markup with the correct markup that would have been assigned by a human annotator, who is only allowed to select tags from the concept-based DTD. We initially elaborate on measuring the quality of XML tag names and concepts, respectively. Subsequently, our notion of markup quality is discussed in the context of evaluating XML tag attributes and extracted named entities, respectively.

**Quality of XML Tag Names** As explained in Subsection 4.4.3, both tagged and plain text units are typically contained in semantically marked-up XML documents. The former are enclosed in semantic XML tags, which optionally comprise attributes, whereas plain text units are not annotated at all. Analogous to text classification (e.g., cf. Sebastiani, 2002, pp. 32–33), the following four mutually exclusive and collectively exhaustive cases are relevant when evaluating the correctness of XML tag names with respect to a given concept-based XML document type definition:

- *True Positive*: Text units enclosed in semantic XML tags are true positives if their tag names represent correct concepts that domain experts typically associate with the respective, marked-up text units.
- *False Positive*: If the semantic XML tag enclosing a text unit does not correspond to the concept that domain experts typically associate with the marked-up text unit, a false positive occurs.
- *True Negative*: A plain text unit not enclosed in a semantic XML tag is a true negative if the concept-based XML document type definition does not comprise a tag for the concept that domain experts typically associate with this text unit.
- *False Negative*: A false negative occurs when a plain text unit, which is not enclosed in a semantic XML tag, in fact represents a semantic concept that is listed in the concept-based XML document type definition.

To summarize, the term positive (and negative, respectively) refers to marked-up (and plain, respectively) text units in semantically tagged XML documents. Furthermore, the term true designates correctly marked-up text units whereas the term false indicates erroneous markup. Here, the notion of markup refers to both positives and negatives.

Considering the concept-based XML document type definition established for our example archive listed in Table 4.28 on page 141, Table 6.1 illustrates these four cases. The first sentence is a true positive because it is marked up with the correct XML tag from the DTD. The tag name `SaleOfBusinessUnit` corresponds to a concept that domain experts typically associate with the exemplary sentence. It is particularly noteworthy that the correctness of XML tag attributes and extracted named entities, respectively, is irrelevant

**Table 6.1:** Four Sentences Illustrating the Types of XML Tag Names w.r.t. Markup Quality

True Positive:	<code>&lt;SaleOfBusinessUnit Company="ID: 1; Name: Modtech"&gt;USA: Miller to sell unit to Modtech.&lt;/SaleOfBusinessUnit&gt;</code>
False Positive:	<code>&lt;SaleOfBusinessUnit&gt;In 1995, the unit's net sales were \$230 million.&lt;/SaleOfBusinessUnit&gt;</code>
True Negative:	The total acquisition value will be about \$48 million, payable in Occidental common stock.
False Negative:	Final settlement is scheduled for October 21, Miller said.

when evaluating the quality of XML tag names. In the first sentence of Table 6.1, failing to extract the company name `Miller` does not change the true positive status at all. The second sentence<sup>1</sup> listed in this table illustrates the false positive case since its XML tag name `SaleOfBusinessUnit` does not correspond to the true concept, as perceived by domain experts. Sentence three in Table 6.1 illustrates the true negative case because the DTD does not contain an XML tag whose name represents an appropriate semantic concept. In contrast, the fourth sentence in this table is a false negative since the DTD contains an appropriate XML tag (i.e., `ExpectedDealClosing`).

Given a random sample comprising  $s \in \mathbb{N}$  semantically marked-up text units drawn without replacement from a semantically marked-up text archive, let  $n_{TP} \in \mathbb{N} \cup 0$  denote the number of true positives in the sample according to the quality assessment performed by a domain expert. Analogously, let  $n_{FP}, n_{TN}, n_{FN} \in \mathbb{N} \cup 0$  denote the number of false positives, true negatives, and false negatives, respectively, identified by a human expert in the random sample such that  $n_{TP} + n_{FP} + n_{TN} + n_{FN} = s$ . In analogy with common practice in text classification as noted, for example, by Sebastiani (2002, pp. 32–33) and Weiss et al. (2005, pp. 77–80), we introduce three measures of semantic XML markup quality and their approximate 95% confidence intervals. The latter are defined in correspondence with Ester and Sander (2000, pp. 110–111) as well as Evert (2004):

- *XML tag name accuracy*, estimated as  $\hat{a}_{\text{TagName}} := (n_{TP} + n_{TN})/s$ , simultaneously considers the markup correctness of both text units enclosed in semantic XML tags and plain text units in a semantically marked-up text archive. For sufficiently large values of  $s \gg 0$ , the approximate 95% confidence interval of XML tag name accuracy is  $\hat{a}_{\text{TagName}} \pm 1.96 \cdot \sqrt{\hat{a}_{\text{TagName}} \cdot (1 - \hat{a}_{\text{TagName}})/s}$ .
- *XML tag name precision*, estimated as  $\hat{p}_{\text{TagName}} := n_{TP}/(n_{TP} + n_{FP})$ , focuses on the markup correctness of text units enclosed in semantic XML tags in a semantically marked-up text archive. Given a text unit is enclosed in semantic XML tags, it denotes the probability that the respective XML tag is correct. For sufficiently

<sup>1</sup>Since the running example does not contain any false positives at all, the second sentence in Table 6.1 is not part of the exemplary training archive shown in Table 4.1. It is sentence 7 of news item 127801 of Reuters Corpus, Volume 1, English Language, 1996-08-20 to 1997-08-19 (cf. Rose et al., 2002).

large values of  $(n_{\text{TP}} + n_{\text{FP}}) \gg 0$ , the approximate 95% confidence interval of XML tag name precision is  $\hat{p}_{\text{TagName}} \pm 1.96 \cdot \sqrt{\hat{p}_{\text{TagName}} \cdot (1 - \hat{p}_{\text{TagName}}) / (n_{\text{TP}} + n_{\text{FP}})}$ .

- *XML tag name recall*, estimated as  $\hat{r}_{\text{TagName}} := n_{\text{TP}} / (n_{\text{TP}} + n_{\text{FN}})$ , measures the success in identifying concepts defined by the corresponding concept-based XML DTD. It is the ratio of text units enclosed in correct semantic XML tags to the total number of text units representing concepts that are defined in the XML DTD. For sufficiently large values of  $(n_{\text{TP}} + n_{\text{FN}}) \gg 0$ , the approximate 95% confidence interval of XML tag name recall is  $\hat{r}_{\text{TagName}} \pm 1.96 \cdot \sqrt{\hat{r}_{\text{TagName}} \cdot (1 - \hat{r}_{\text{TagName}}) / (n_{\text{TP}} + n_{\text{FN}})}$ .

As each measure emphasizes a specific aspect of markup quality, an overall assessment requires considering them in combination. In this work, we intentionally define XML tag name accuracy, precision, and recall with respect to the overall population of text units in a semantically marked-up text archive. Therefore, these three measures are not specific to certain XML tag names, but rather reflect their average quality. If necessary, this approach may of course be extended to assess the quality of particular text unit sub-populations featuring, for example, a specific semantic concept.

**Quality of XML Tag Attributes** Analogous to evaluating the quality of XML tag names, the correctness of XML tag attributes and extracted named entities, respectively, is assessed against the concept-based XML document type definition. Unlike research in general information extraction (cf. Subsection 2.1.3), the DIASDEM framework thus only requires determining the quality of named entity extraction within text units enclosed in semantic XML tags. To that end, we furthermore restrict the assessment to named entities in marked-up text units whose types correspond to tag-specific attributes defined in the corresponding concept-based XML DTD. Reconsidering the exemplary XML DTD depicted in Table 4.28 on page 141, the XML tag `AgreementOnUnitSale` has two optional attributes named `Company` and `Place`. Consequently, the quality assessment of named entities occurring within text units marked up with the tag `AgreementOnUnitSale` is limited to companies and places only.

Analogous to the assessment of information extraction as performed in the seminal GATE project (cf. Maynard et al., 2001; Cunningham et al., 2002, pp. 114–115), we distinguish four mutually exclusive and collectively exhaustive types of identified named entities within XML tag attributes.

- *Completely Correct Named Entity*: If an attribute value comprises all components of a named entity without any extraction flaws, it is considered to be completely correct. In line 7 of Table 4.31 on page 146, for instance, the value `ID: 6; Name: Miller Building Systems Inc` of attribute `Company` illustrates a completely correct named entity since the company-specific information is properly extracted.
- *Partially Correct Named Entity*: If an attribute value does not contain a few tokens of the corresponding, typically complex named entity without conveying erroneous information (e.g., concerning its type), it is partially correct. The value `ID: 7;`

**Name:** Miller Structures Inc of attribute **Company** in line 7 of Table 4.31 serves as an example of this case. Although the mentioned location of this company (i.e., California) is not extracted, the identified information about its name is correct.

- *Incorrect Named Entity:* If an attribute value conveys erroneous information about a named entity, it represents an incorrect named entity. In line 7 of Table 4.31, the attribute **Place="California"** represents an incorrect named entity. This place refers to a named entity that is not a stand-alone place, but rather an integral part of the partially correct company **ID: 7; Name: Miller Structures Inc.**
- *Missing Named Entity:* An existing named entity, whose type is listed in the tag-specific XML DTD attribute list, is considered to be a missing one if this entity is not referenced by an attribute. For example, the missing named entity **Miller** is mentioned in line 6 of Table 4.31, but it is not contained in the attribute **Company** of the XML tag `<SaleOfBusinessUnit Company="ID: 1; Name: Modtech">`.

In a random sample comprising  $s \in \mathbb{N}$  semantically marked-up text units drawn to assess the quality of XML tag names, let  $n_{NE} \in \mathbb{N} \cup 0$  denote the true number of relevant named entities occurring in the subset of text units enclosed in semantic XML tags. When determining  $n_{NE}$ , a named entity occurring in marked-up content is considered only if its named entity type corresponds to a valid XML attribute name of the enclosing semantic XML tag. Furthermore, let  $n_{CC}, n_{PC}, n_I, n_M \in \mathbb{N} \cup 0$  denote the number of completely correct, partially correct, incorrect, and missing, respectively, named entities in the sample according to the quality assessment performed by a domain expert such that  $n_{NE} \leq n_{CC} + n_{PC} + n_I + n_M \leq n_{NE} + n_I$ .

Again following Maynard et al. (2001) and Cunningham et al. (2002, pp. 114–115), the following three measures and their approximate 95% confidence intervals are employed to assess the quality of named entity extraction within our framework. Confidence intervals are defined in analogy with Ester and Sander (2000, pp. 110-111) as well as Evert (2004).

- *XML attribute accuracy*, estimated as  $\hat{a}_{\text{Attribute}} := (n_{CC} + 1/2 \cdot n_{PC})/n_{NE}$ , is defined as the proportion of accurately extracted named entities over all relevant named entities. Thereby, partially correct named entities are assigned a correctness weight of  $1/2$ , which of course is arguable. For sufficiently large values of  $n_{NE} \gg 0$ , the approximate 95% confidence interval of XML attribute accuracy is  $\hat{a}_{\text{Attribute}} \pm 1.96 \cdot \sqrt{\hat{a}_{\text{Attribute}} \cdot (1 - \hat{a}_{\text{Attribute}})/n_{NE}}$ .
- *XML attribute precision*, estimated as  $\hat{p}_{\text{Attribute}} := (n_{CC} + 1/2 \cdot n_{PC})/(n_{CC} + 1/2 \cdot n_{PC} + n_I)$ , is the ratio of correctly extracted named entities over all identified ones. Hence, it measures the accuracy of extracted named entities without considering any missed ones. For sufficiently large values of  $(n_{CC} + 1/2 \cdot n_{PC} + n_I) \gg 0$ , the approximate 95% confidence interval of XML attribute precision is  $\hat{p}_{\text{Attribute}} \pm 1.96 \cdot \sqrt{\hat{p}_{\text{Attribute}} \cdot (1 - \hat{p}_{\text{Attribute}})/(n_{CC} + 1/2 \cdot n_{PC} + n_I)}$ .
- *XML attribute recall*, estimated as  $\hat{r}_{\text{Attribute}} := (n_{CC} + 1/2 \cdot n_{PC})/(n_{CC} + 1/2 \cdot n_{PC} + n_M)$ , is defined as the proportion of correctly extracted named entities over all existing ones. The higher this measure, the better the system in not missing relevant

named entities. For sufficiently large values of  $(n_{CC} + 1/2 \cdot n_{PC} + n_M) \gg 0$ , the approximate 95% confidence interval of XML attribute recall is  $\hat{r}_{\text{Attribute}} \pm 1.96 \cdot \sqrt{\hat{r}_{\text{Attribute}} \cdot (1 - \hat{r}_{\text{Attribute}}) / (n_{CC} + 1/2 \cdot n_{PC} + n_M)}$ .

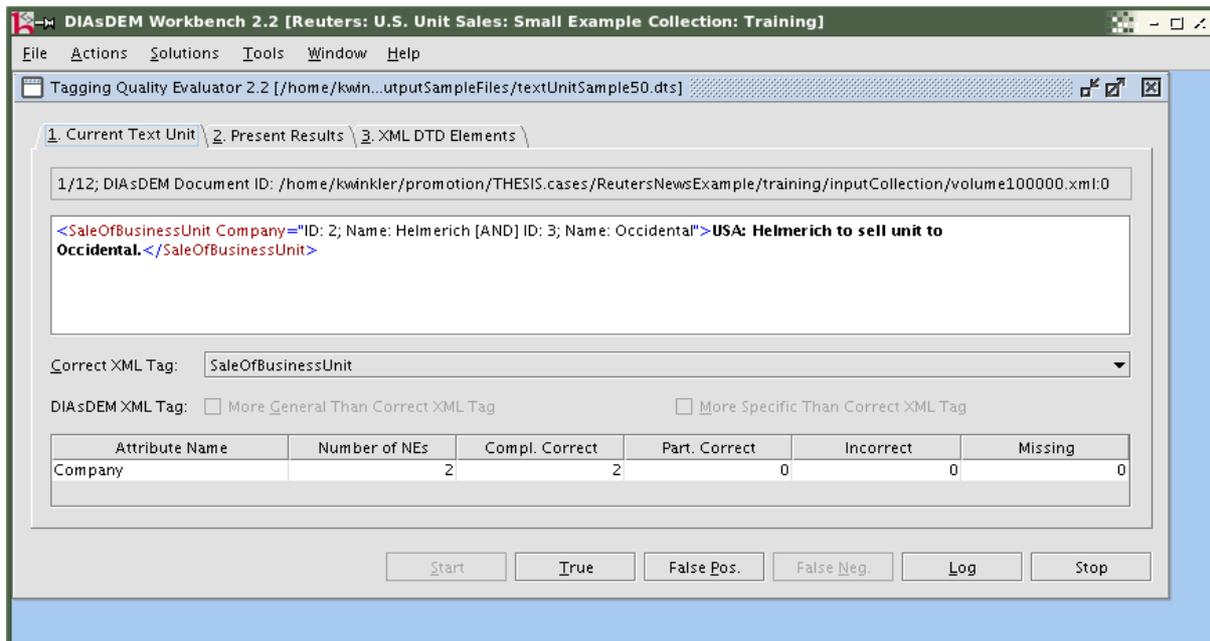
Each measure emphasizes a different aspect of XML tag attribute quality. Therefore, they shall be analyzed in combination to ensure a thorough assessment. Analogous to evaluating the quality of XML tag names, the quality of attributes is not assessed in a type-specific or even tag-specific way. According to the above definitions, XML attribute accuracy, precision, and recall are rather average measures of quality taking all relevant, identified named entity types in consideration. Computing type-specific or tag-specific measures of XML attribute quality, however, and thereby overcoming this intentional simplification is straightforward.

### 6.1.2 Extending DIAsDEM Workbench

Due to the necessary incorporation of human domain expertise into our markup quality assessment procedure, an effective software support of the entire evaluation process is paramount to minimize the required human efforts and to ensure statistically valid results. Therefore, DIASDEM WORKBENCH facilitates the experimental evaluation of our framework for semantic XML tagging by means of two extensions.

Firstly, a random sample of semantically marked-up text units is created during the process of transforming an intermediate text archive into a semantically marked-up one. Given the desired sample size, each semantically marked-up text unit, whether or not it is actually enclosed in semantic XML tags, has the same chance of being drawn into the sample without replacement. If requested, a random sample of semantically marked-up text units is drawn during knowledge application as well.

Having drawn a random text unit sample, the workbench tool TAGGING QUALITY EVALUATOR 2.2 secondly facilitates the interactive assessment procedure performed by a domain expert. Depicted in Figure 6.1, this tool effectively supports an interactive quality assessment of semantic XML markup. More specifically, the human expert is asked to evaluate the markup of one semantically marked-up text unit after the other. Each text unit in the random sample is displayed exactly as contained in the corresponding semantically marked-up XML document. Based on his or her expertise, the domain expert evaluates the correctness of the (possibly non-existing) XML tag name with respect to the accompanying concept-based DTD, which can be viewed instantaneously. If the automatically assigned XML tag name is incorrect, the correct one from the finite tag set in the concept-based DTD is input as well. For text units enclosed in semantic XML tags, the domain expert is further asked to assess the performance in extracting the relevant named entities. To that end, the true number of relevant named entities, as well as the number of correct, partially correct, incorrect, and missing named entities is input for each XML attribute that is listed in the tag-specific DTD attribute definition. In addition, all evaluation results are persistently stored, the entire process can be stopped and resumed at



**Figure 6.1:** Screen Shot of the TAGGING QUALITY EVALUATOR 2.2 Tool of DIAsDEM WORKBENCH GUI CLIENT

any time, all assessment decisions are documented in a protocol, and the markup quality measures defined in this section are finally computed for the entire random sample.

## 6.2 Real-World Applications of the DIAsDEM Framework

This section describes in detail the design and the results of two case studies conducted to evaluate our framework for semantic XML tagging of large, domain-specific text collections. Since both real-world case studies aim at semantically enhancing publicly available text archives to acquire actionable knowledge for competitive intelligence applications, we initially outline this application domain in Subsection 6.2.1. The case study on transforming a large collection of German Commercial Register entries into semantically tagged XML documents is presented in the subsequent subsection. In Subsection 6.2.3, we discuss the second experiment conducted to semantically mark up English news items about United States mergers and acquisitions.

### 6.2.1 Semantic XML Markup for Competitive Intelligence

In his seminal work on competitive strategy, Porter (1980b, p. 47) emphasized “that a central aspect of strategy formulation is perceptive competitor analysis.” In the past decades, Porter’s competitor analysis has gradually evolved into the more holistic notion

of competitive intelligence (e.g., cf. Kahaner, 1997; West, 1999; Vedder et al., 1999). This term denotes both the process of conducting a competitive analysis and the resulting actionable information (Herring, 1988, p. 5). As an integral phase in the strategic management process, a thorough competitive analysis encompasses the systematic, legal and ethical collection, storage, and analysis of publicly available information about current and potential competitors, other stakeholders, and the environment to anticipate trends and strategic moves (Winkler, 2003, pp. 4–5).

“To be effective,” Porter (1980b, pp. 71–74) noted, “there is the need for an organized mechanism—some sort of competitor intelligence system—to insure the process is efficient.” Taking an information systems perspective, Winkler (2003) established a conceptual framework for competitive intelligence information systems that considers their users, the respective process, and necessary data. Due to the large number of publicly available and highly relevant textual resources (e.g., cf. Sullivan, 2001, pp. 413–416), a competitive analysis strongly depends on leveraging computer-accessible texts about trends and stakeholders. Consequently, a competitive intelligence information system shall adequately support the analysis of text archives. In this context, Sullivan (2001, pp. 407–436), Chen et al. (2002), Gerdes Jr. (2003), Weiss et al. (2005, pp. 157–162), Glance et al. (2005), as well as Feldman and Sanger (2007, pp. 279–295) successfully employed knowledge discovery methods to respond to competitive intelligence challenges.

Many large archives, which are highly relevant for competitive intelligence purposes (e.g., business news stories or quarterly reports to shareholders), do not contain text documents covering a variety of topics, but instead comprise domain-specific text documents of relatively homogeneous content. Although focusing on the same central topic, documents contained in these kinds of thematic collections often feature many different specialized, but nonetheless frequently recurring, subtopics at the sentence or paragraph level. If a text archive indeed exhibits a thematic clustering tendency at the sentence or paragraph level, our proposed DIAsDEM framework for semantic XML tagging is applicable to semantically enhance the collection. To experimentally assess our framework, two real-world text archives were chosen that exhibit a clustering tendency at the sentence level. Commercial Register entries and news items on mergers and acquisitions are of great importance when conducting a thorough competitive analysis. Since these sources of information about current and potential competitors are continuously published on a large scale, automatically converting them into semantically marked-up XML documents provides the benefits outlined in Section 1.3. Besides enabling concept-based information retrieval, semantic XML markup facilitates the integration of textual content with related, structured and semi-structured data (cf. Winkler, 2003, pp. 24–25).

### **6.2.2 German Commercial Register Entries**

This subsection summarizes a case study in which our framework was employed to semantically annotate a large collection of German Commercial Register entries. To that end, we briefly describe this domain and characterize the document collection. Subsequently,

we present the approach taken in the knowledge discovery phase and in the knowledge application phase of our framework. Finally, we outline a potential use case for semantically marked-up Commercial Register entries and discuss the lessons learned.

**Application Domain and Text Archive** In Germany, the Commercial Register comprises information of legal significance for commercial transactions (see Peltzer et al., 2000, pp. 6–7). Pursuant to the German Commercial Code, merchants and businesses of particular legal forms are required to file specific facts for registration in the competent Commercial Register. Although these registers are maintained by district courts for businesses domiciled in their respective area, all Commercial Register entries are additionally published by an official federal gazette to ensure nation-wide publicity. The register entries are made available to the general public since up-to-date knowledge about a company’s legal affairs is essential to its current and prospective competitors as well as stakeholders including, but not limited to, suppliers and business customers.

Three major categories of Commercial Register entries can be distinguished: initial entries of newly established business, update entries, and entries announcing the deletion of businesses from the register. Depending on the legal form of the registering business, initial entries comprise form-specific information about, for example, the firm name and Commercial Register number, the domicile including the address, the business purpose indicating the industry a business operates in, the amount of capital, as well as the names of partners and managing directors, respectively. Due to their very nature, update entries reflect changes and modifications of legal facts (e.g., capital increases or reductions, appointments of new managing directors, or changes of the legal form and the business purpose). Finally, deletion entries contain detailed information concerning, for instance, the dissolution and liquidation of partnerships or companies with limited liability, as well as the commencement and conclusion of insolvency proceedings. Consequently, the Commercial Register constitutes a vast and continuously updated source of legally reliable information that is highly relevant in a competitive intelligence context.

Although Commercial Register entries comprise unstructured textual content, they are written by civil servants in a highly controlled language for a specific purpose (cf. Bowker and Pearson, 2002) whose scope is defined by German commercial law. Hence, they do not comprise narrative text, but expository content (cf. Hearst, 1997, p. 35) about merchants and businesses. When reading multiple entries, one undoubtedly observes a natural, thematic clustering tendency (cf. Jain and Dubes, 1988, pp. 201–202) at the sentence level. For example, appointments of managing directors are typically announced using one sentence. Since the German alphabet is finite and texts are interpreted from left to right, all four requirements for applying our DIAsDEM framework (see Subsection 3.2 on page 62) are fulfilled by German Commercial Register entries.

Table 6.2 lists three exemplary Commercial Register update entries and their English translations. Each entry starts with the Commercial Register number of the respective company (e.g., HRA 13 870) and its effective publication date. Subsequently, the official firm name is typically listed along with the place of domicile. The registered legal content

**Table 6.2:** Exemplary German Commercial Register Entries and English Translations

---

HRA 13 870 – 12. Oktober 2001: zeitlos schlafen Behringer und Eckert OHG, Leipzig (Gottschedstr. 12, 04109 Leipzig). Persönlich haftender Gesellschafter: Wilhelm Behringer, geb. 12.06.1959, Berlin; Sabine Eckert, geb. 12.06.1961, Berlin. Offene Handelsgesellschaft. Die Gesellschaft hat am 1. Dezember 1994 begonnen. Der Sitz der Gesellschaft ist von Berlin (bisher Amtsgericht Charlottenburg, HRA 26 745) nach Leipzig verlegt.

HRA 13 870 – October 12, 2001: zeitlos schlafen Behringer und Eckert OHG, Leipzig (Gottschedstr. 12, 04109 Leipzig). General partners: Wilhelm Behringer, born 1959/06/12, Berlin; Sabine Eckert, born 1961/06/12, Berlin. General commercial partnership. The partnership has commenced operations on December 1, 1994. The domicile of the partnership has been relocated from Berlin (previously registered at district court Charlottenburg, HRA 26 745) to Leipzig.

---

4 HRB 4970 – 21. März 2001: Fuchs Gesellschaft mit beschränkter Haftung, Bensheim. Die Prokura Joachim Jäger ist erloschen. Joachim Jäger, geb. 18.08.1967, Wald-Michelbach, ist zum Geschäftsführer bestellt. Er vertritt die Gesellschaft mit einem anderen Geschäftsführer oder einem Prokuristen.

4 HRB 4970 – March 21, 2001: Fuchs Gesellschaft mit beschränkter Haftung, Bensheim. The Prokura conferred to Joachim Jäger has been terminated. Joachim Jäger, born 1967/08/18, Wald-Michelbach, has been appointed managing director. He represents the company jointly with another managing director or Prokura holder.

---

90 HRB 113 – 13. Dezember 2000: 'S I E G E' Siedlungsgesellschaft für das Verkehrspersonal gemeinnützige Gesellschaft mit beschränkter Haftung, Mainz. Der Prokurist Wolfgang Weber ist zur Veräußerung und Belastung von Grundstücken und grundstücksgleichen Rechten berechtigt. Die Prokura des Herbert Schmeiser ist erloschen. Peter Kiesewetter ist nicht mehr Geschäftsführer.

90 HRB 113 – December 13, 2000: 'S I E G E' Siedlungsgesellschaft für das Verkehrspersonal gemeinnützige Gesellschaft mit beschränkter Haftung, Mainz. The Prokura holder Wolfgang Weber has been authorized to transfer and encumber real property and rights equivalent to real property. The Prokura conferred to Herbert Schmeiser has been terminated. Peter Kiesewetter is no longer managing director.

---

follows this entry header. All texts listed in Table 6.2 are update entries that announce legally relevant changes, such as a domicile relocation in the first one. The natural clustering tendency at the sentence level is, for example, illustrated by texts 2 and 3 that both announce the termination of a once-granted Prokura<sup>2</sup>. To facilitate the understanding of the highly regulated Commercial Register domain, Appendix C comprises the most important German terms along with their English translations, which are mostly based on Peltzer et al. (2000).

Descriptive properties of the text archives marked up during knowledge discovery and knowledge application are summarized in Table 6.3. It is gratefully acknowledged that both text archives were provided for research purposes by Heins + Partner GmbH, Bielefeld, Germany. The training text archive consists of 23,668 Commercial Register entries published between 1997 and 2001. This collection was carefully assembled to include all

---

<sup>2</sup>Prokura holders are legal representatives of the business they are employed by. The scope of the representative power referred to as the Prokura is regulated by the German Commercial Code (cf. Peltzer et al., 2000, pp. 10–11).

**Table 6.3:** Overview of Text Archives Used in the Commercial Register Case Study

Text Archive Property	Knowledge Discovery Phase	Knowledge Application Phase
Language	German	German
Text Unit Granularity	Sentence	Sentence
Number of Documents	23,668	35,881
Number of Text Units	115,647	165,330
Publication Date of Documents	1997/06/25–2001/12/01	2004/01/01–2004/01/31

three entry types discussed above for all possibly occurring legal forms, such as sole proprietorship, general partnership, limited liability company, and stock corporation. This stratified random sampling approach (e.g., cf. Hand et al., 2001, pp. 135-136) was necessary to facilitate the discovery of concepts and semantic XML tags, respectively, that are published less frequently (e.g., foundation entries of non-profit limited liability companies or stock corporations). The text archive processed in the knowledge application phase, in contrast, contains all 35,881 Commercial Register entries published in Germany in January 2004. Thereby, the classification knowledge discovered in a stratified sample was applied to a real-world scenario of sequentially incoming text documents. The author is not aware of any substantial structural changes of the relevant Commercial Code sections between the years 1997 and 2004.

This case study was performed using DIASDEM WORKBENCH 2.2 introduced in Chapter 5. Core results of this case study, such as the concept-based XML DTD, the tagging quality assessment protocol, and a sample of marked-up XML documents, are electronically accessible (see Appendix A on page 217).

**Knowledge Discovery Phase** Based on the introduction of the DIASDEM knowledge discovery process in Chapter 4, we henceforth outline the KDT steps performed in the Commercial Register case study. The fully parameterized KDT process flow (cf. Definition 13 on page 61) was interactively created to conform with Algorithm 4.9 (see page 149) by means of selecting, parameterizing, and executing appropriate KDT tasks within DIASDEM WORKBENCH. Since each KDT task execution was recorded in a batch script by our workbench, the resulting fully parameterized KDT process flow was represented by a DIASDEM batch script.

#### 1. *Pre-processing of text documents*

- a) *Text document import:* Heins + Partner GmbH provided a comma-separated input file that comprised the actual Commercial Register entries in plain text format and descriptive metadata (e.g., competent district court, entry type, firm name, and Commercial Register number of the merchant or company). This input file was transformed into a text archive (cf. Definition 3 on page 58) suitable for applying our framework.

- b) *Creation of text units*: Subsequently, the original text documents were decomposed into sentences and text units, respectively, as introduced in Subsection 4.2.1. To identify sentence boundaries, the heuristic sentence identification algorithm outlined in Subsection 5.2.1 was employed. The input parameters to this algorithm comprised approx. 3,500 German, mostly domain-independent abbreviations containing periods (e.g., "z.B.") and 46 mostly domain-independent regular expressions masking periods that do not serve as sentence boundaries (e.g., "http://www.my-domain.de", "1.000,00 EUR", or "01.01.2007"). By creating text units, the initially imported text archive was converted into an intermediate text archive (cf. Definition 21 on page 75).
- c) *Tokenization of text units*: In this task, sentences were tokenized to separate words from punctuation marks (cf. Subsection 4.2.1). In addition, 47 regular expressions were designed to convert, for example, date and time expressions into their canonical forms suitable for named entity extraction. The input parameters also included 96 domain-specific, frequently occurring multi-token terms (e.g., legal forms like "Limited Company"). Finally, 18 search and replace statements were utilized to break relevant composite nouns into their main components and to deal with frequent hyphenation patterns.
- d) *Extraction of named entities*: Using the prototypical named entity extractor of DIASDEM WORKBENCH (cf. Subsection 5.2.1), instances of the named entity types listed in Table 6.4 were identified and replaced by placeholders, as discussed in Subsection 4.2.2. Many look-up lists (e.g., German forenames, surnames, cities, and villages) were provided by Heins + Partner GmbH for the purpose of this case study. A considerable effort was put into designing the domain-specific composite named entity patterns to identify mentioned persons, companies, and company relocations in text units.
- e) *Lemmatization of text units*: TREETAGGER (cf. Schmid, 1994, 1999) was employed without any parameter modifications via the DIASDEM WORKBENCH task LEMMATIZE TEXT UNITS to determine the lemma forms of all natural language words, as explained in Subsection 4.2.3.
- f) *Word sense disambiguation*: Since the language used in the rather narrow Commercial Register domain is highly controlled by the German Commercial Code, disambiguating the sense of lemmatized tokens (cf. Subsection 4.2.3) was of minor importance in this case study. In particular, only 13 tokens were disambiguated by appropriate contextual terms and sense definitions, respectively (cf. Manning and Schütze, 1999, pp. 242-244).
- g) *Establishment of the controlled vocabulary*: Following the pre-processing steps, the intermediate training archive comprised 23,635 distinct terms not counting named entity placeholders and numeric tokens. An auxiliary DIASDEM WORKBENCH task created an initial, domain-specific controlled vocabulary containing all terms with collection frequency greater than 49. Subsequently, this vocabulary was manually refined into a thesaurus by defining synonyms,

**Table 6.4:** Named Entity Types Extracted in the Commercial Register Case Study

Named Entity Type	Description, Example, and Resources Utilized in Extraction
document_id	Headers of Commercial Register entries (e.g., "HRA 13 870 - 12. Oktober 2001:") were extracted using one regular expression.
amount_of_money	German canonical forms (e.g., "25000,00 EUR") of amounts of money, such as "25 000,- Euro", were extracted using one regular expression.
date	German canonical forms (e.g., "31.12.2000") of date expressions, like "31. Dezember 2000", were extracted using one regular expression.
paragraph	Section headings of articles of association, as well as laws and regulations (e.g., "§ 201 UmwG") were extracted using three regular expressions.
organization_id	Commercial Register numbers that uniquely identify merchants and businesses in the area of one district court (e.g., "Amtsgericht Charlottenburg, HRA 26 745") were extracted using three regular expressions.
place	German cities (e.g., "Leipzig"), and villages (e.g., "Wald-Michelbach"), as well as large international cities were extracted using 40 place indicator terms, 22,852 names of places, and 53,726 place name affixes or district names.
street	Street names (e.g., "Gottschedstr. 12") were extracted using 22 street name suffixes, 194,918 non-standard street names, and three regular expressions.
profession	A list of 867 frequently occurring job or occupational titles (e.g., "Dipl.-Ingenieur") was utilized to extract them from Commercial Register entries.
person	Person names (e.g., "Sabine Eckert") were initially identified using 51 person name indicator terms, 1,856 academic titles, 143,562 forenames, 51 middle initials, 430,689 surnames, 356 surname suffixes, and 63 name affixes. They were subsequently combined with nearby occurring dates, places, street names, and job titles into composite named entities of type person (e.g., "Sabine Eckert, geb. 12.06.1961, Berlin") using 175 composite named entity patterns.
company	Company names (e.g., "Fuchs Gesellschaft mit beschränkter Haftung") were initially identified using 648 organization indicator terms, 259 organization name suffixes (e.g., "GmbH" and "Ltd."), 118 organization affixes, one regular expression, and the registered firm name contained in the metadata of each entry. They were subsequently combined with nearby occurring places, street names, district courts, and Commercial Register numbers into composite named entities of type company (e.g., "Fuchs Gesellschaft mit beschränkter Haftung, Bensheim") using 305 composite named entity patterns.
company_relocation	Identified as composite named entities by means of 29 composite named entity patterns, company relocations (e.g., "von Berlin (bisher Amtsgericht Charlottenburg, HRA 26 745) nach Leipzig") comprise available information about the origin and destination domicile of relocating companies (i.e., place, street name, district court, and Commercial Register number).

removing irrelevant terms, and adding new terms. The final thesaurus comprised 1,366 terms, 158 of which served as text unit descriptors in this case study. By far the largest number of 1,208 non-descriptors were mapped onto the descriptor representing the purpose, or line of business, of companies.

2. *Iterative clustering of text unit vectors*: Table 6.5 lists parameter settings and results for all 12 clustering iterations performed in this case study. In each clustering

iteration, the following five knowledge discovery tasks were performed:

- a) *Vectorization of text units*: Using a DIASDEM WORKBENCH task, we applied the text unit descriptor weighting scheme (cf. Subsection 4.2.5 on page 94) to map text units onto text unit vectors.
- b) *Selecting, parameterizing, and executing the clustering algorithm in discovery mode*: Although DIASDEM WORKBENCH supports all algorithms discussed in Subsection 4.3.2, we deliberately selected the bisecting  $k$ -means algorithm (Steinbach et al., 2000) for execution. The purpose of this case study is not to compare the performance of different clustering algorithms, but to demonstrate the applicability of our framework to mark up real-world text archives. One advantageous property of the bisecting  $k$ -means is that it requires only one input parameter, namely, the desired number of clusters  $k \in \mathbb{N}$ . The bisecting  $k$ -means was employed in combination with the cosine similarity due to the characteristics of this proximity measure (cf. Subsection 4.3.1). Initially, the number of desired text unit clusters was set to  $k = 800$ . Preliminary experiments, which involved clustering a 10% random text unit vector sample for  $k = 200, 400, \dots, 1600$  and assessing the corresponding cluster validity using the indices surveyed in Table 4.12 on page 106, indicated that there exists a large number of clusters. However, the overall cluster separation tends to worsen for  $k > 800$ . In addition, increasing  $k$  resulted in a growing number of text unit clusters that were assigned only one member. Hence, we started with  $k = 800$  in iteration 1 to find both compact and well separated clusters. As listed in Table 6.5, the number of desired text unit clusters was raised to  $k = 1600$  to discover more specific semantic concepts in iterations 7 through 9. Finally, the number of text unit clusters was decreased to  $k = 400$  to identify even the most general remaining concepts in the last iteration 12.
- c) *Automatic ranking of text unit clusters and automatic default labeling of acceptable clusters*: The workbench task MONITOR CLUSTER QUALITY 2.2 automatically ranked the resulting clusters by decreasing quality (cf. Subsection 4.3.3 on page 115) and additionally generated default labels of all qualitatively acceptable clusters (cf. Subsection 4.4.1 on page 133). Table 6.5 lists the user-specified, iteration-specific settings of our five DIASDEM cluster quality criteria along with the number of resulting acceptable and unacceptable clusters. This table reveals that our cluster quality requirements were step-by-step relaxed to discover more specific concepts in earlier iterations and rather general concepts in later iterations.
- d) *Interactive screening of cluster ranking and interactive review of default cluster labels*: Using the CLUSTER LABEL EDITOR tool of DIASDEM WORKBENCH, the automatically generated cluster quality decisions and the default cluster labels were manually reviewed and modified, if necessary. If text units assigned to an automatically accepted cluster did not adhere to a single and overwhelmingly pure semantic concept, the corresponding cluster was typically rejected

**Table 6.5:** Summary of Pattern Discovery in 12 Clustering Iterations in the Commercial Register Case Study

Clustering Iteration	Input Text Units	Parameter of Bisecting $k$ -Means	Dominant Descriptor Threshold	Rare Descriptor Threshold	Maximum Descriptor Coverage	Minimum Descriptor Dominance	Minimum Cluster Size
1	115,647	$k = 800$	0.8	0.0025	0.10	0.50	50
2	66,853	$k = 800$	0.8	0.0050	0.20	0.40	45
3	44,598	$k = 800$	0.8	0.0075	0.20	0.35	40
4	34,120	$k = 800$	0.8	0.0075	0.25	0.30	35
5	30,201	$k = 800$	0.8	0.0100	0.30	0.25	30
6	24,606	$k = 800$	0.8	0.0100	0.35	0.20	25
7	19,990	$k = 1,600$	0.8	0.0100	0.35	0.20	20
8	14,531	$k = 1,600$	0.8	0.0200	0.35	0.20	15
9	11,828	$k = 1,600$	0.8	0.0200	0.35	0.20	10
10	9,256	$k = 800$	0.8	0.0200	0.35	0.33	10
11	7,655	$k = 800$	0.8	0.1000	0.35	0.20	10
12	6,306	$k = 400$	0.8	0.1000	0.35	0.20	10

Clustering Iteration	Clustering Quality Index	Number of Clusters (Number of Text Units Therein) after				Stop Iterative Clustering
		Automated Ranking		Interactive Screening		
		Acceptable	Unacceptable	Acceptable	Unacceptable	
1	0.505	52 (52,269)	748 (63,378)	39 (48,794)	761 (66,853)	No
2	0.427	75 (24,731)	725 (42,122)	61 (22,255)	739 (44,598)	No
3	0.414	76 (12,459)	724 (32,139)	55 (10,478)	745 (34,120)	No
4	0.370	95 (7,224)	705 (26,896)	52 (3,919)	748 (30,201)	No
5	0.376	137 (8,699)	663 (21,502)	90 (5,595)	710 (24,606)	No
6	0.370	175 (8,794)	625 (15,812)	96 (4,616)	704 (19,990)	No
7	0.445	281 (8,302)	1,319 (11,688)	193 (5,459)	1,407 (14,531)	No
8	0.452	237 (4,976)	1,363 (9,555)	147 (2,703)	1,453 (11,828)	No
9	0.460	370 (5,342)	1,230 (6,486)	207 (2,572)	1,394 (9,256)	No
10	0.471	243 (5,383)	557 (3,873)	93 (1,601)	707 (7,655)	No
11	0.454	281 (4,603)	519 (3,052)	96 (1,349)	704 (6,306)	No
12	0.418	169 (4,024)	231 (2,282)	33 (828)	367 (5,478)	Yes

by the domain expert to ensure a high markup quality. To discover the most specific semantic markup possible, text unit clusters representing a mixture of specific concepts under a common, more general concept were also manually rejected as unacceptable in earlier iterations. Facing a very large number of clusters, this review was intentionally limited to qualitatively acceptable clusters in the first 11 iterations. In the final iteration 12, however, all remaining 400 clusters were carefully inspected. Due to the large number of discovered semantic concepts in the Commercial Register domain, all final cluster labels were additionally prefixed by category identifiers chosen by the human expert (e.g., the prefix CB04 of CB04\_GeneralPartner\_Joining). Category identifiers group cluster labels and semantic XML tags, respectively, in a meaningful way to facilitate querying the semantically marked-up XML documents. The

**Table 6.6:** Semantically Marked-Up XML Document Comprising the First Exemplary German Commercial Register Entry (cf. Table 6.2 on Page 183)

---

1:	<?xml version="1.0" encoding="ISO-8859-1"?>
2:	<!DOCTYPE CommercialRegisterEntry SYSTEM "CommercialRegisterEntry.dtd">
3:	
4:	<CommercialRegisterEntry>
5:	<TaggedDocument>
6:	<JA01_EntryHeader Company="ID: 5; Name: zeitlos schlafen Behringer und Eckert OHG, Leipzig; Place: 04109 Leipzig; Street: Gottschedstr. 12" DocumentID="HRA 13870 - 12.10.2001">HRA 13 870 - 12. Oktober 2001: zeitlos schlafen Behringer und Eckert OHG, Leipzig (Gottschedstr. 12, 04109 Leipzig).</JA01_EntryHeader>
7:	<CB01_GeneralPartner Person="ID: 12; Name: Wilhelm Behringer; DateOfBirth: 12.06.1959; Place: Berlin [AND] ID: 13; Name: Sabine Eckert; DateOfBirth: 12.06.1961; Place: Berlin">Persönlich haftender Gesellschafter: Wilhelm Behringer, geb. 12.06.1959, Berlin; Sabine Eckert, geb. 12.06.1961, Berlin.</CB01_GeneralPartner>
8:	<EA01_LegalForm>Offene Handelsgesellschaft.</EA01_LegalForm>
9:	<EB01_CommencementOfPartnership Date="01.12.1994">Die Gesellschaft hat am 1. Dezember 1994 begonnen.</EB01_CommencementOfPartnership>
10:	<IC04_Domicile_Change CompanyRelocation="ID: 18; OriginCommercialRegisterID: Amtsgericht Charlottenburg, HRA 26745; OriginPlace: Berlin; DestinationPlace: Leipzig">Der Sitz der Gesellschaft ist von Berlin (bisher Amtsgericht Charlottenburg, HRA 26 745) nach Leipzig verlegt.</IC04_Domicile_Change>
11:	</TaggedDocument>
12:	</CommercialRegisterEntry>

---

results of this interactive review are also listed in Table 6.5.

- e) *Update of the intermediate text archive:* At the end of each clustering iteration, the task TAG TEXT UNITS of DIASDEM WORKBENCH was executed to persistently store the results in the intermediate text archive. After iteration 12, only 5,478 (4.7%) of altogether 115,647 text units remained assigned to qualitatively unacceptable clusters. Hence, the iterative clustering phase of our knowledge discovery process was terminated by the expert.

### 3. Post-processing of discovered patterns

- a) *Establishment of the concept-based XML DTD:* Subsequent to the final clustering iteration of the knowledge discovery process, a concept-based XML document type definition was derived by a DIASDEM WORKBENCH task in accordance with Subsection 4.4.2 on page 138. Generated with a minimum attribute support of 10%, this concept-based XML document type definition was named `CommercialRegisterEntry.dtd` in correspondence with its root element. It comprises 242 elements representing a variety of semantic concepts from specific ones (e.g., `DA01_ManagingDirector_Appointment`) to broader ones (e.g., `FA01_Merger`). For each DTD element, the automatically generated DTD documentation template contains an enumeration of valid attributes and

**Table 6.7:** Semantically Marked-Up XML Document Comprising the Second Exemplary German Commercial Register Entry (cf. Table 6.2 on Page 183)

---

1:	<?xml version="1.0" encoding="ISO-8859-1"?>
2:	<!DOCTYPE CommercialRegisterEntry SYSTEM "CommercialRegisterEntry.dtd">
3:	
4:	<CommercialRegisterEntry>
5:	<TaggedDocument>
6:	<JA01_EntryHeader Company="ID: 3; Name: Fuchs Gesellschaft mit beschränkter Haftung; Place: Bensheim" DocumentID="4 HRB 4970 - 21.03.2001">4 HRB 4970 - 21. März 2001: Fuchs Gesellschaft mit beschränkter Haftung, Bensheim.
7:	</JA01_EntryHeader>
7:	<DC03_Prokura_Termination Person="ID: 5; Name: Joachim Jäger">Die Prokura Joachim Jäger ist erloschen.</DC03_Prokura_Termination>
8:	<DA01_ManagingDirector_Appointment Person="ID: 9; Name: Joachim Jäger; DateOfBirth: 18.08.1967; Place: Wald-Michelbach">Joachim Jäger, geb. 18.08.1967, Wald-Michelbach, ist zum Geschäftsführer bestellt.
9:	</DA01_ManagingDirector_Appointment>
9:	<BG04_SpecificProvision_PersonalPronoun_JointPowerOfRepresentation>Er vertritt die Gesellschaft mit einem anderen Geschäftsführer oder einem Prokuristen.
10:	</BG04_SpecificProvision_PersonalPronoun_JointPowerOfRepresentation>
10:	</TaggedDocument>
11:	</CommercialRegisterEntry>

---

five exemplary text units. Appendix A on page 217 includes details on how to electronically access the concept-based XML DTD and the corresponding DTD documentation template.

- b) *Semantic XML tagging of text documents*: As explained in Subsection 4.4.3, the entire original text archive was converted into 23,668 semantically marked-up XML documents conforming to the concept-based XML document type definition `CommercialRegisterEntry.dtd` derived beforehand. Table 6.6, Table 6.7, and Table 6.8 contain semantically marked-up XML documents that correspond to the three exemplary Commercial Register entries listed in Table 6.2 on page 183. A 5% random sample of semantically marked-up XML documents is electronically accessible (see Appendix A on page 217).

To assess the quality of the generated semantic XML markup (cf. Section 6.1), a random sample comprising 1,156 (1%) of altogether 115,647 semantically marked-up text units was drawn. The author served as a domain expert and assessed the markup quality of this text unit sample using the TAGGING QUALITY EVALUATOR 2.2 tool of DIAS-DEM WORKBENCH. Columns 2 and 3 of Table 6.9 on page 192 list the results of quality assessment in the knowledge discovery phase. The quality measures for XML tag names and XML tag attributes are indeed very promising. With 0.95 confidence, for example, the XML tag name accuracy is in the interval [0.9411;0.9655]. Focusing on the markup correctness of text units enclosed in semantic XML tags further improves the results. For

**Table 6.8:** Semantically Marked-Up XML Document Comprising the Third Exemplary German Commercial Register Entry (cf. Table 6.2 on Page 183)

---

1:	<?xml version="1.0" encoding="ISO-8859-1" ?>
2:	<!DOCTYPE CommercialRegisterEntry SYSTEM "CommercialRegisterEntry.dtd">
3:	
4:	<CommercialRegisterEntry>
5:	<TaggedDocument>
6:	<JA01_EntryHeader Company="ID: 3; Name: 'S I E G E' Siedlungsgesellschaft für das Verkehrspersonal gemeinnützige Gesellschaft mit beschränkter Haftung; Place: Mainz" DocumentID="90 HRB 113 - 13.12.2000">90 HRB 113 - 13. Dezember 2000: 'S I E G E' Siedlungsgesellschaft für das Verkehrspersonal gemeinnützige Gesellschaft mit beschränkter Haftung, Mainz.
7:	</JA01_EntryHeader>
8:	Der Prokurist Wolfgang Weber ist zur Veräußerung und Belastung von Grundstücken und grundstücksgleichen Rechten berechtigt.
9:	<DC03_Prokura_Termination Person="ID: 7; Name: Herbert Schmeiser">Die Prokura des Herbert Schmeiser ist erloschen.</DC03_Prokura_Termination>
10:	<DA02_ManagingDirector_Removal Person="ID: 9; Name: Peter Kiesewetter">Peter Kiesewetter ist nicht mehr Geschäftsführer.</DA02_ManagingDirector_Removal>
11:	</TaggedDocument>
12:	</CommercialRegisterEntry>

---

instance, the XML tag name precision is in the interval [0.9663; 0.9846] with 0.95 confidence. The protocol of the entire quality assessment procedure is electronically available (see Appendix A on page 217).

**Knowledge Application Phase** Applying the classification knowledge acquired in phase 1 of our framework to mark up all Commercial Register entries published in January 2004 required only slight modifications of the recorded DIAsDEM batch script. In particular, the following tasks were edited and deleted, respectively, in the batch script to bridge knowledge discovery and knowledge application (cf. Section 4.5 on page 148):

1. *Pre-processing of text documents*

- a) *Text document import:* The January 2004 input file, which was also provided by Heins + Partner GmbH, was imported instead of the training input file.

2. *Iterative classification of text unit vectors:* All tasks recorded in 12 clustering iterations of the knowledge discovery phase were edited and removed, respectively, to transform them into 12 corresponding classification iterations, as introduced in Section 4.5 on page 148:

- a) *Vectorization of text units:* Text units were mapped onto text unit vectors in application mode (cf. Subsection 4.2.5 on page 94). Hence, the iteration-specific collection frequency components of descriptor weights were not computed, but retrieved from the respective clustering iteration metadata items.

**Table 6.9:** Results of the Semantic XML Markup Quality Assessment in the Commercial Register Case Study

Quality Measure for XML Tag Names	Knowledge Discovery Phase		Knowledge Application Phase	
	Value in Sample	Approximate 95% Confidence Interval	Value in Sample	Approximate 95% Confidence Interval
Number of Text Units	1,156		1,239	
Proportion of True Positives	0.9282	[0.9133; 0.9431]	0.9177	[0.9024; 0.9330]
Proportion of False Positives	0.0234	[0.0146; 0.0321]	0.0242	[0.0157; 0.0328]
Proportion of True Negatives	0.0251	[0.0161; 0.0341]	0.0307	[0.0211; 0.0403]
Proportion of False Negatives	0.0234	[0.0146; 0.0321]	0.0274	[0.0183; 0.0365]
XML Tag Name Accuracy	0.9533	[0.9411; 0.9655]	0.9483	[0.9360; 0.9607]
XML Tag Name Precision	0.9755	[0.9663; 0.9846]	0.9743	[0.9652; 0.9834]
XML Tag Name Recall	0.9755	[0.9663; 0.9846]	0.9710	[0.9613; 0.9806]

Quality Measure for XML Tag Attributes	Knowledge Discovery Phase		Knowledge Application Phase	
	Value in Sample	Approximate 95% Confidence Interval	Value in Sample	Approximate 95% Confidence Interval
True Number of Named Entities	1,266		1,271	
XML Attribute Accuracy	0.9198	[0.9049; 0.9348]	0.9359	[0.9224; 0.9493]
XML Attribute Precision	0.9741	[0.9651; 0.9831]	0.9859	[0.9793; 0.9926]
XML Attribute Recall	0.9724	[0.9632; 0.9817]	0.9604	[0.9496; 0.9713]

- b) *Execution of the clustering algorithm in application mode:* In each classification iteration, the clustering algorithm parameterized and executed in the corresponding clustering iteration was employed in application mode (cf. Subsection 4.3.4 on page 124). Hence, text unit vectors were assigned to clusters identified and semantically labeled in the knowledge discovery phase.
- c) *Automatic ranking of text unit clusters and automatic default labeling of acceptable clusters:* The corresponding tasks were deleted in the batch script.
- d) *Interactive screening of cluster ranking and interactive review of default cluster labels:* The related, interactive workbench tasks were not performed.

### 3. *Post-processing of discovered patterns*

- a) *Establishment of the concept-based XML DTD:* This task was removed from the batch script as well.

The modified batch script, which conformed to Algorithm 4.11 on page 151, was executed without any human intervention by DIASDEM WORKBENCH BATCH SCRIPT CLIENT. To evaluate the markup quality, as discussed in Section 6.1, we subsequently drew a random sample comprising 1,239 (0.75%) of altogether 165,330 semantically marked-up text units in the application text archive. Analogous to the knowledge discovery phase, the author assessed the markup quality of this sample. The results are listed in columns 4 and 5 of Table 6.9. Furthermore, Appendix A contains information on how to electronically access the quality assessment protocol. It is noteworthy that the quality measures

for XML tag names and XML tag attributes are approx. equal in both knowledge application and knowledge discovery. The XML attribute accuracy achieved in the knowledge application sample (93.59%) is even better than the value obtained from the knowledge discovery sample (91.98%). This rather unusual fact can be attributed to the stratified sampling approach for Commercial Register entries in the knowledge discovery phase. Unlike the regularly published entries marked up during knowledge application, the stratified sample was designed to comprise infrequently occurring concepts and named entities as well.

**Usage Scenario and Lessons Learned** The results of this case study were truly encouraging. In particular, the high precision and recall values, which were obtained for both XML tag names and their attributes in the knowledge application phase, paved the way for applying the DIAsDEM framework in a productive environment. Concretely, one usage scenario for semantically marked-up Commercial Register entries was realized at Heins + Partner GmbH, Bielefeld, Germany, by employing our framework for semantic XML tagging of large text archives. Incoming Commercial Register entries are now automatically transformed into XML documents in a daily batch process. The major business driver to employ our framework was the necessity of updating a complex database of German businesses without the delays triggered by manually or semi-automatically extracting the relevant information from Commercial Register entries. Combining the benefits of semantic XML markup and named entity extraction enabled Heins + Partner to update its database with legally reliable information published by German district courts. By performing additional clustering iterations, extending the domain-specific thesaurus, and fine-tuning the named entity extraction parameters, Heins + Partner achieved even better results than published in this work. After all, ensuring the highest possible markup quality was the primary objective of this project to reduce the risk of inserting false information (e.g., regarding the commencement of insolvency proceedings) into the database.

When recapitulating the Commercial Register case study, the following lessons learned are particularly noteworthy and shall thus be summarized:

- Think big, start small! The iterative aspect of knowledge discovery emphasized in Subsection 2.1.1 is of paramount importance to achieve quick wins. When starting a semantic XML tagging project in a new domain, we suggest adopting a step-by-step approach to quickly gain insights into the particularities of the focal text archive. For example, we initially parameterized a KDT process flow using default parameters and an automatically generated controlled vocabulary. Subsequently, the markup quality was step-by-step improved by modifying input parameters.
- Be precise! The domain-specific controlled vocabulary and the input parameters of our named entity extraction module need to be continuously refined by carefully defining synonyms, adding new relevant terms, or precisely defining pattern extraction rules. Achieving a high markup quality in the knowledge application phase thus requires precise and accurate human efforts during knowledge discovery. For

instance, the false negative rate can be reduced by updating the thesaurus if new synonyms of existing text unit descriptors occur in Commercial Register entries.

- Be patient! Our iterative clustering phase is designed to discover both specific and general, as well as more and less frequent concepts. As discussed above, manually rejecting certain, qualitatively acceptable clusters (e.g., representing a mixture of specific concepts under a common, more general concept) in earlier iterations facilitates the discovery of more specific text unit clusters in subsequent iterations.
- Combine domain and knowledge discovery expertise! To successfully employ our framework for semantic XML tagging, both types of knowledge are required. For example, domain expertise is necessary to establish or refine an effective thesaurus. Selecting and parameterizing appropriate clustering algorithms, however, requires rather generic knowledge discovery experience. In this case study, the author took on both roles due to his interdisciplinary background.
- Structure discovered concepts! When discovering many semantic concepts, structuring them by prefixing concept names with category identifiers (e.g., the prefix CB04 of CB04\_GeneralPartner\_Joining) improves the lucidity and therefore the usability of the generated concept-based XML document type definition.

Furthermore, important issues to be resolved by future research and future versions of DIASDEM WORKBENCH are discussed in Section 7.2.

### 6.2.3 News about U.S. Mergers and Acquisitions

The second case study in the competitive intelligence domain is described in detail in this subsection. Analogous to the approach adopted in presenting the Commercial Register case study, we initially outline the application domain and characterize the text collection. Subsequently, we describe both phases of our framework, concisely present a potential use case for semantically marked-up news stories about mergers and acquisitions, and finally discuss the lessons learned in this case study.

**Application Domain and Text Archive** According to Gaughan (2002, p. xi), the “field of mergers and acquisitions continues to experience dramatic growth.” In November 1999, for example, British Vodafone AirTouch PLC announced the hostile acquisition of German Mannesmann AG in a transaction valued at \$202.8 billion (cf. Gaughan, 2002, pp. 3–8). In principle, a merger is a combination of two companies such that only one company survives and the merged one ceases to exist. However, the author defined a variety of specific merger types and additionally differentiated mergers from consolidations (i.e., two or more companies form an entirely new company) and takeovers, which sometimes refer to hostile transactions. In general, the term acquisition denotes both the purchase of an entire company (i.e., a merger) or the controlling interest in a company (cf. Gaughan, 2002, p. 595).

Without doubt, the timely collection and analysis of information concerning merger and acquisition (M&A) activities of both current and potential competitors are of great importance to obtain actionable competitive intelligence. “One of the most difficult searches” to gain competitive intelligence, as emphasized by Fuld (1995, p. 353), “involves information on a merger or acquisition.” Kahaner (1997, pp. 138–142) even devoted an entire chapter to the benefits of using competitive intelligence in M&A transactions. When it comes to forecasting a competitor’s likely actions in the future, publicly available information about mergers and acquisitions is capable of providing highly valuable insights (cf. Kahaner, 1997, pp. 102–106). In particular, M&A analysis helps companies to discover new and potential competitors, learn from the success and failure of others, and increase the range and quality of one’s own acquisition targets (cf. Kahaner, 1997, pp. 22–28). Due to the great business value inherent in textual documents focusing on M&A deals, this application domain has attracted many information extraction (cf. Subsection 2.1.3) researchers in recent decades (e.g., see Rau, 1888; Soderland, 1999; Freitag, 2000; Moens, 2006, pp. 213–214).

Fuld (1995, p. 353) collected a number of merger and acquisition news sources and directories. Another important, publicly available resource for gathering information on M&A talks and rumors, as well as announced, closed, and completed transactions are news stories continuously published by local and international news agencies. Published by Reuters Limited, the Reuters Corpus, Vol. 1, English Language, 1996-08-20 to 1997-08-19, comprises 806,791 news stories (cf. Rose et al., 2002; Lewis et al., 2004). This text archive, which is freely available to the research community, contains all English language news items produced by Reuters journalists between August 20, 1996, and August 19, 1997. The news stories of this archive are formatted in an early version of NewsML, the XML-based News Markup Language (cf. Geroimenko, 2004, p. 103). To enhance the value of this collection for research purposes, each news item is annotated with category codes for featured topics, regions, and industry sectors. Rose et al. (2002) discussed the annotation, or metadata generation, process in great depth, which involved a combination of text categorization, manual editing, and manual correction.

In our second case study, we focused on news items annotated with topic code C181, which is assigned to news about mergers and acquisitions, and region code USA, which is assigned to news items related to the United States of America. Table 6.10 lists three exemplary news stories about U.S. mergers and acquisitions in the original spelling, which includes some mistakes (e.g., “acquisiton” and “subisidiary”). By limiting our input documents to news items assigned the topic code C181 and region code USA, we extracted a thematic sub-corpus that fulfills all four requirements for using the DIAsDEM framework (see Subsection 3.2 on page 62). In particular, this sub-collection solely consists of domain-specific documents of relatively homogeneous and expository content, namely, M&A news related to transactions in the United States. Unlike the highly controlled language used by German civil servants to compose Commercial Register entries, however, the language used by Reuters journalists to report on M&A issues is largely varying in grammar, vocabulary, and style. An average news story, which consists of approx. ten

**Table 6.10:** Exemplary News Stories about U.S. Mergers and Acquisitions (Source: Reuters Corpus, Volume 1, English Language, 1996-08-20 to 1997-08-19; cf. Rose et al., 2002)

---

USA: Gillette to acquire Duracell - WSJ. Gillette Co is near an agreement to buy Duracell International Inc in a stock swap worth over \$7 billion, the Wall Street Journal reported Thursday. The newspaper, quoting anonymous sources, said the deal could be announced Thursday, adding that neither company would comment on the story. Duracell's largest shareholder is the New York buyout firm Kohlberg Kravis Roberts & Co, which holds 34 percent of the company, the article said. KKR declined to comment when contacted by Reuters late Wednesday. The deal values Duracell shares at \$58.85 each, or 0.904 share of Gillette. The Journal said the transaction includes the assumption of debt and is expected to immediately be accretive to Gillette's earnings. Neither company was immediately available to comment on the article. –New York Newsdesk (212) 859-1610.

---

USA: Culbro discusses acquiring cigar maker. Culbro Corp said Tuesday the company and its General Cigar unit are in preliminary talks that could lead to the acquisition of privately held Villazon & Co Inc which makes premium cigars in Florida and Honduras. The Culbro statement gave no terms for the acquisition, but did say it anticipates completing the transaction by year end. – New York Newsdesk 212-859-1610.

---

USA: Houghten target of complaint out of merger. Houghten Pharmaceuticals Inc said Friday that a complaint has been filed against the company and its officers in a dispute over its acquisition of ChromaXome Corp. The complaint was filed by Michael Dickman and Katie Thompson, the founders of ChromaXome, who accuse Houghten of denying them access to and employment with ChromaXone. The company said the complaint also accuses it of preventing Dickman and Thompson from obtaining certain consideration in connection with the acquisition. The complaint was filed in Superior Court in San Diego. Houghten said the claims are without merit. Houghten develops and uses combinatorial chemistry and other molecular diversity technologies to pursue the discovery of small-molecule drug therapies. ChromaXome, now a wholly owned subsidiary, was a privately held combinatorial company.

---

sentences, is longer than an average Commercial Register entry.

Table 6.11 lists descriptive properties of the text sub-collection used in the knowledge discovery and knowledge application phase. In contrast to the first case study, we did not pursue a stratified sampling approach because a priori knowledge concerning the existence of specific types of M&A news stories was not available. Instead, we pragmatically divided the entire Reuters corpus into two distinct parts based on the publication date of news items. All 12,187 training news items were published prior to 1997/04/01. Consequently, we validated the classification knowledge acquired during knowledge discovery using 8,408 news stories published between 1997/04/01 and 1997/08/19. Analogous to the first case study, we employed DIASDEM WORKBENCH 2.2 introduced in Chapter 5 and published core results of this case study, such as the concept-based XML DTD and a sample of marked-up news items, on the Web site <http://ka.rsten-winkler.de/thesis>.

**Knowledge Discovery Phase** Subsequently, we describe the KDT steps performed in this case study along the DIASDEM knowledge discovery process introduced in Chap-

**Table 6.11:** Overview of Text Archives Used in the Reuters News Case Study

Text Archive Property	Knowledge Discovery Phase	Knowledge Application Phase
Language	English	English
Text Unit Granularity	Sentence	Sentence
Number of Documents	12,187	8,408
Number of Text Units	122,256	86,238
Publication Date of Documents	1996/08/20–1997/03/31	1997/04/01–1997/08/19

ter 4. Analogous to the first case study, we interactively created a fully parameterized KDT process flow (cf. Definition 13 on page 61) in conformance with Algorithm 4.9 on page 149. To that end, we selected, parameterized, executed, and recorded appropriate KDT tasks within DIASDEM WORKBENCH. The resulting case-specific, fully parameterized KDT process flow was represented by a DIASDEM batch script.

#### 1. *Pre-processing of text documents*

- a) *Text document import:* As indicated above, the Reuters corpus is formatted in an early version of NewsML, the XML-based News Markup Language (see Geroimenko, 2004, p. 103). Using a dedicated DIASDEM WORKBENCH task, we transformed all relevant news stories, namely, the ones annotated with both topic code C181 and region code USA, from their NewsML representation into a text archive (cf. Definition 3 on page 58).
- b) *Creation of text units:* Analogous to the first case study, the initial news stories were decomposed into sentences and text units, respectively, as introduced in Subsection 4.2.1. Thereby, the initially imported text archive was converted into an intermediate text archive (cf. Definition 21 on page 75). The input parameters to this task comprised 1,886 English, mostly domain-independent abbreviations containing periods and 29 mostly domain-independent regular expressions masking periods not serving as sentence boundaries.
- c) *Tokenization of text units:* All intermediate text units were tokenized to separate words from punctuation marks (cf. Subsection 4.2.1). Furthermore, 185 regular expressions were designed to convert instances of relevant named entity types into their canonical forms suitable for named entity extraction. Moreover, the input parameters of this task included 691 domain-specific, frequently occurring multi-token terms, such as names of important organizations, stock exchanges, and news papers.
- d) *Extraction of named entities:* As in the first case study, the named entity extractor of DIASDEM WORKBENCH was employed to identify instances of the named entity types listed in Table 6.12, as discussed in Subsection 4.2.2. Again, a considerable effort was put into designing domain-specific composite named entity patterns to identify persons, companies, units of companies,

**Table 6.12:** Named Entity Types Extracted in the Reuters News Case Study

Named Entity Type	Description, Example, and Resources Utilized in Extraction
amount_of_money	Canonical forms (e.g., "107100000 USD") of amounts of money, like "\$107.1 million", were extracted by two regular expressions.
amount_of_money_per_share	Amounts of money per share, such as "134 USD per share", were extracted using two regular expressions.
number_of_shares	Expressions denoting numbers of shares, like "820000 common shares" or "34500 shares", were extracted using one regular expression.
percentage	Canonical forms (e.g., "8.5%" or "51.8%") of percentages, like "8.5 pct" or "51.8 percent", were extracted using one regular expression.
date	Canonical forms (e.g., "????-07-02" or "1996-??-??") of dates, like "July 2" or "1996", were extracted using one regular expression. Missing date components are represented by question marks.
stock_exchange	154 major stock exchanges (e.g., "London Stock Exchange" or "NASDAQ") were extracted by means of one regular expression.
date_period	Extracted dates were combined into date periods (e.g., "between June 25 and August 20") using four composite named entity patterns.
place	Country names, which occur in a particular context, and large cities from all over the world (e.g., "Quito") were extracted using 30 place indicator terms, 24,009 names of places, and 659 place name affixes.
profession	A list of 397 frequently occurring job titles (e.g., "President and CEO" or "Board Member") was utilized to extract them from news stories.
person	Person names (e.g., "Don Box") were initially identified using 190 person name indicator terms, 43 academic titles, 10,737 forenames, 49 middle initials, 97,296 surnames, 372 surname suffixes, and 25 name affixes. They were subsequently combined with nearby occurring places and job titles into composite named entities of type person (e.g., "President and Chief Executive Officer Don Box") using 81 composite named entity patterns.
company	Company names (e.g., "KKR", "Gillette", or "Elbit") were initially identified using 1,458 organization indicator terms, 302 organization name suffixes (e.g., "Corp." and "Ltd."), 418 organization affixes, one regular expression, and 2,294 names of large international corporations. They were subsequently combined with nearby occurring places into composite named entities of type company (e.g., "Israel's Elbit") using 49 composite named entity patterns.
unit_of_company	Extracted company names, places, and percentages were combined into named entities of type unit_of_company, such as "Batus Inc., a unit of BAT Industries Inc.", using 66 composite named entity patterns.
equity_stake	Extracted company names, places, number of shares, and percentages were combined into named entities of type equity_stake, such as "stake in Centennial Cellular Corp to 4.91 percent or 453,900 common shares", using 84 composite named entity patterns.

and equity stakes. In contrast to the Commercial Register domain, identifying company names posed a particular challenge because many company names

occurring in news stories were not suffixed with abbreviated legal forms. To address this issue, we adopted the two stage, so-called learn - filter - apply approach to recognizing company names (Volk and Clematide, 2001). In step 1, company names were identified based on legal form suffixes (e.g., "Corp") to discover, for example, "Culbro Corp [said]" in the second exemplary news item of Table 6.10. All identified names were tokenized and heuristically filtered to recognize informal, abbreviated names of companies, like "Culbro [discusses]", in step 2.

- e) *Lemmatization of text units*: We employed TREE TAGGER (cf. Schmid, 1994, 1999) without any parameter modifications to lemmatize natural language words (cf. Subsection 4.2.3). In addition, 90 search and replace statements were utilized to pragmatically deal with frequently occurring English clitics (e.g., replacing "won ' t" with "will not") and important phrasal verbs (e.g., combining the two tokens "close at" into a single token "close\_at").
  - f) *Word sense disambiguation*: Despite the relatively narrow M&A domain, disambiguating the sense of lemmatized tokens (see Subsection 4.2.3) was necessary because a few terms were highly ambiguous when considered as an out-of-context token. For example, the term "interest" has at least three relevant domain-specific senses (i.e., borrowing, stake, and involvement). Hence, we defined contextual terms and sense definitions, respectively, to disambiguate 18 important tokens (cf. Manning and Schütze, 1999, pp. 242-244).
  - g) *Establishment of the controlled vocabulary*: Subsequent to the preceding pre-processing steps, the intermediate training archive comprised 24,313 distinct terms not counting named entity placeholders and numeric tokens. Analogous to the first case study, an auxiliary DIASDEM WORKBENCH task created an initial, domain-specific controlled vocabulary containing terms with a collection frequency of 50 or greater. This vocabulary was manually refined into a thesaurus by defining synonyms, removing irrelevant terms, and adding new terms. The final thesaurus comprised 734 terms including 227 text unit descriptors. 205 of 607 non-descriptors were mapped onto the descriptor representing various lines of businesses (e.g., apparel, automotive, and banking).
2. *Iterative clustering of text unit vectors*: The parameter settings and results for the eight clustering iterations performed in this case study are listed in Table 6.13. We executed the following five knowledge discovery tasks in each clustering iteration:
- a) *Vectorization of text units*: The text unit descriptor weighting scheme introduced in Subsection 4.2.5 was used to map text units onto text units vectors.
  - b) *Selecting, parameterizing, and executing the clustering algorithm in discovery mode*: For the reasons discussed in the context of the Commercial Register case study, we deliberately selected the bisecting  $k$ -means algorithm (Steinbach et al., 2000) in combination with the cosine similarity. To appropriately set the input parameter  $k$  in the first iteration, we conducted the preliminary

**Table 6.13:** Summary of Pattern Discovery in Eight Clustering Iterations

Clustering Iteration	Input Text Units	Parameter of Bisecting $k$ -Means	Dominant Descriptor Threshold	Rare Descriptor Threshold	Maximum Descriptor Coverage	Minimum Descriptor Dominance	Minimum Cluster Size
1	122,256	$k = 800$	0.8	0.005	0.10	0.25	50
2	92,509	$k = 800$	0.8	0.010	0.15	0.25	30
3	87,193	$k = 1,600$	0.8	0.010	0.20	0.20	25
4	73,877	$k = 1,600$	0.8	0.020	0.25	0.20	20
5	65,517	$k = 1,600$	0.8	0.020	0.25	0.15	15
6	56,336	$k = 800$	0.8	0.050	0.30	0.15	15
7	45,646	$k = 800$	0.8	0.050	0.35	0.15	10
8	38,863	$k = 400$	0.8	0.100	0.35	0.15	10

Clustering Iteration	Clustering Quality Index	Number of Clusters (Number of Text Units Therein) after				Stop Iterative Clustering
		Automated Ranking		Interactive Screening		
		Acceptable	Unacceptable	Acceptable	Unacceptable	
1	0.313	57 (26,548)	743 (95,708)	47 (29,747)	753 (92,509)	No
2	0.291	35 (5,909)	765 (86,600)	31 (5,316)	769 (87,193)	No
3	0.312	141 (14,295)	1,459 (72,898)	126 (13,316)	1,474 (73,877)	No
4	0.320	138 (9,790)	1,462 (64,087)	117 (8,360)	1,483 (65,517)	No
5	0.312	217 (10,470)	1,383 (55,047)	192 (9,181)	1,408 (56,336)	No
6	0.340	260 (17,977)	540 (38,359)	157 (10,690)	643 (45,646)	No
7	0.340	239 (12,794)	561 (32,852)	131 (6,783)	669 (38,863)	No
8	0.377	254 (24,100)	146 (14,763)	108 (10,390)	292 (28,473)	Yes

experiments described in the preceding case study and obtained similar results. Hence, the number of desired text unit clusters was initially set to  $k = 800$  to find both compact and well separated clusters. In analogy with the iterative clustering approach taken in the Commercial Register case study, the number of desired text unit clusters was raised to  $k = 1600$  to discover more specific semantic concepts in iterations 3 through 5. Finally, the cluster number was decreased to  $k = 400$  to identify even the most general remaining concepts in the final iteration 8.

- c) *Automatic ranking of text unit clusters and automatic default labeling of acceptable clusters:* The DIASDEM WORKBENCH task MONITOR CLUSTER QUALITY 2.2 was employed to automatically rank the resulting clusters by decreasing quality and to generate default labels of qualitatively acceptable clusters. Table 6.13 summarizes the user-specified, iteration-specific settings of our DIASDEM cluster quality criteria as well as the resulting number of acceptable and unacceptable clusters. We relaxed the cluster quality requirements step-by-step to discover more specific concepts in earlier iterations and rather general concepts in later iterations.
- d) *Interactive screening of cluster ranking and interactive review of default cluster labels:* All automatically generated cluster quality decisions and their de-

fault cluster labels were manually reviewed and modified, if necessary, using the CLUSTER LABEL EDITOR tool of DIASDEM WORKBENCH. In general, we pursued the same interactive screening strategy as motivated in the first case study to discover the most specific semantic markup possible. Furthermore, cluster labels were also prefixed by category identifiers chosen by the human expert (e.g., the prefix A110 of A110\_AcquisitionOfCompany) to facilitate querying the semantically marked-up XML documents. Moreover, our interactive review was intentionally limited to qualitatively acceptable clusters in the first seven iterations. In addition, the largest qualitatively unacceptable cluster comprising 5,715 text units featuring no text unit descriptor at all was inspected in iteration 1. In the final iteration 8, all remaining 400 clusters were carefully inspected. Unlike Commercial Register entries, however, M&A news stories typically comprise noisy sentences representing neither frequently occurring nor relevant concepts. In particular, certain statements issued by spokespersons, analysts, or other stakeholders (e.g., "I think it's definitely a possibility," Paris said.) are meaningless when focusing on the text unit level and thereby – intentionally (cf. Definition 1) – ignoring the context. To avoid passing members of clusters comprising only noisy text units or text units without any descriptor occurrence to the next iteration, they were temporarily marked as acceptable and assigned the label Z999\_IrrelevantTopic. Prior to establishing the concept-based XML DTD in the post-processing phase, however, the corresponding 16,753 text units were eventually marked as qualitatively unacceptable to prevent the useless semantic XML tag <Z999\_IrrelevantTopic> from inclusion in the DTD. The results of this interactive review are listed in Table 6.13.

- e) *Update of the intermediate text archive:* The task TAG TEXT UNITS of DIASDEM WORKBENCH was executed at the end of each clustering iteration to persistently store the results in the intermediate text archive. After the eighth iteration, 28,473 (23.3%) of altogether 122,256 text units remained assigned to qualitatively unacceptable clusters. Nevertheless, the iterative clustering phase of our knowledge discovery process was terminated by the KDT expert because the perceived progress in discovering new semantic concepts was only marginal in this iteration.

### 3. Post-processing of discovered patterns

- a) *Establishment of the concept-based XML DTD:* Subsequent to clustering iteration 8, a concept-based XML document type definition was derived (cf. Subsection 4.4.2 on page 138). Generated with a minimum attribute support of 10%, this concept-based XML document type definition is named in correspondence with its root element NewsItemOnMergersAndAcquisitions. It comprises 217 elements that represent a variety of semantic concepts, from specific ones (e.g., C500\_EquityStakeIsHoldForInvestmentPurposes) to broader ones (e.g.,

**Table 6.14:** Semantically Marked-Up XML Document Comprising the First Exemplary Reuters News Item (cf. Table 6.10 on Page 196)

---

1:	<?xml version="1.0" encoding="ISO-8859-1"?">
2:	<!DOCTYPE NewsItemOnMergersAndAcquisitions SYSTEM "NewsItemOnMergersAndAcquisitions.dtd">
3:	
4:	<NewsItemOnMergersAndAcquisitions>
5:	<TaggedDocument>
6:	<A110_AcquisitionOfCompany Company="ID: 2; Name: Gillette [AND] ID: 3; Name: Duracell">USA: Gillette to acquire Duracell - WSJ. </A110_AcquisitionOfCompany>
7:	Gillette Co is near an agreement to buy Duracell International Inc in a stock swap worth over \$7 billion, the Wall Street Journal reported Thursday.
8:	The newspaper, quoting anonymous sources, said the deal could be announced Thursday, adding that neither company would comment on the story.
9:	<O230_OwnershipOfShares Company="ID: 13; Name: Duracell [AND] ID: 14; Name: Kohlberg Kravis Roberts & Co" Percentage="34%" Place="New York">Duracell's largest shareholder is the New York buyout firm Kohlberg Kravis Roberts & Co, which holds 34 percent of the company, the article said. </O230_OwnershipOfShares>
10:	<E800_NoComments Company="ID: 16; Name: KKR">KKR declined to comment when contacted by Reuters late Wednesday.</E800_NoComments>
11:	<B003_DealValue_StockDeal AmountOfMoney="58.85 USD" Company="ID: 21; Name: Duracell">The deal values Duracell shares at \$58.85 each, or 0.904 share of Gillette.</B003_DealValue_StockDeal>
12:	<K510_ExpectedEffectOfDeal_FinancialPerformance Company="ID: 24; Name: Gillette">The Journal said the transaction includes the assumption of debt and is expected to immediately be accretive to Gillette's earnings. </K510_ExpectedEffectOfDeal_FinancialPerformance>
13:	<E800_NoComments>Neither company was immediately available to comment on the article.</E800_NoComments>
14:	<P010_ContactInformation_Newsroom Place="New York">-New York Newsdesk (212) 859-1610 </P010_ContactInformation_Newsroom>
15:	</TaggedDocument>
16:	</NewsItemOnMergersAndAcquisitions>

---

R100\_StatementOrCommentAboutDeal). In addition, a DTD documentation template was automatically generated that contains a list of valid attributes and five exemplary text units for each DTD element. Both the concept-based XML DTD and the corresponding DTD documentation template are electronically accessible (see Appendix A on page 217).

- b) *Semantic XML tagging of text documents:* The input text archive was converted into 12,187 semantically marked-up XML documents conforming to the concept-based XML DTD `NewsItemOnMergersAndAcquisitions.dtd` derived beforehand. Three semantically marked-up XML documents, which correspond to the exemplary news stories listed in Table 6.10 on page 196, are listed in

**Table 6.15:** Semantically Marked-Up XML Document Comprising the Second Exemplary Reuters News Item (cf. Table 6.10 on Page 196)

---

1:	<?xml version="1.0" encoding="ISO-8859-1"?>
2:	<!DOCTYPE NewsItemOnMergersAndAcquisitions SYSTEM "NewsItemOnMergersAndAcquisitions.dtd">
3:	
4:	<NewsItemOnMergersAndAcquisitions>
5:	<TaggedDocument>
6:	<E100_Talks Company="ID: 1; Name: Culbro">USA: Culbro discusses acquiring cigar maker.</E100_Talks>
7:	Culbro Corp said Tuesday the company and its General Cigar unit are in preliminary talks that could lead to the acquisition of privately held Villazon & Co Inc which makes premium cigars in Florida and Honduras.
8:	<J210_ExpectationsAboutCompletionOfAcquisition Company="ID: 10; Name: Culbro">The Culbro statement gave no terms for the acquisition, but did say it anticipates completing the transaction by year end. </J210_ExpectationsAboutCompletionOfAcquisition>
9:	<P010_ContactInformation_Newsroom Place="New York">- New York Newsdesk 212-859-1610.</P010_ContactInformation_Newsroom>
10:	</TaggedDocument>
11:	</NewsItemOnMergersAndAcquisitions>

---

Table 6.14, Table 6.15, and Table 6.16, respectively. Analogous to the first case study, a 5% random sample of semantically marked-up XML documents is electronically accessible (see Appendix A on page 217).

Finally, we drew a random sample comprising 1,222 (1%) of altogether 122,256 semantically marked-up text units to evaluate the quality of the generated semantic XML markup (cf. Section 6.1). Due to his background in business administration, the author was capable of serving again as a domain expert and assessed the markup quality of this text unit sample using the TAGGING QUALITY EVALUATOR 2.2 tool of DIASDEM WORKBENCH. Columns 2 and 3 of Table 6.17 on page 205 summarize the results of the quality assessment in the knowledge discovery phase. The quality of XML tag names and XML tag attributes in marked-up M&A news stories is certainly not as high as the results achieved in our Commercial Register case study. With 0.95 confidence, for example, the XML tag name accuracy is in the interval [0.7856; 0.8298]. Focusing on the markup correctness of text units enclosed in semantic XML tags slightly improves the results. For instance, the XML tag name precision and recall are in the intervals [0.8029; 0.8566] and [0.8280; 0.8793], respectively, with 0.95 confidence. A protocol of the entire quality assessment procedure is electronically available (see Appendix A on page 217).

**Knowledge Application Phase** In the second phase of our framework, we applied the classification knowledge discovered in phase 1 to semantically annotate 8,408 news items contained in the application text archive. To that end, the following tasks were edited and

**Table 6.16:** Semantically Marked-Up XML Document Comprising the Third Exemplary Reuters News Item (cf. Table 6.10 on Page 196)

---

1:	<?xml version="1.0" encoding="ISO-8859-1"?>
2:	<!DOCTYPE NewsItemOnMergersAndAcquisitions SYSTEM "NewsItemOnMergersAndAcquisitions.dtd">
3:	
4:	<NewsItemOnMergersAndAcquisitions>
5:	<TaggedDocument>
6:	USA: Houghten target of complaint out of merger.
7:	Houghten Pharmaceuticals Inc said Friday that a complaint has been filed against the company and its officers in a dispute over its acquisition of ChromaXome Corp.
8:	The complaint was filed by Michael Dickman and Katie Thompson, the founders of ChromaXome, who accuse Houghten of denying them access to and employment with ChromaXone.
9:	The company said the complaint also accuses it of preventing Dickman and Thompson from obtaining certain consideration in connection with the acquisition.
10:	The complaint was filed in Superior Court in San Diego.
11:	Houghten said the claims are without merit.
12:	<0100_LineOfBusiness Company="ID: 22; Name: Houghten">Houghten develops and uses combinatorial chemistry and other molecular diversity technologies to pursue the discovery of small-molecule drug therapies.</0100_LineOfBusiness>
13:	ChromaXome, now a wholly owned subsidiary, was a privately held combinatorial company.
14:	</TaggedDocument>
15:	</NewsItemOnMergersAndAcquisitions>

---

deleted, respectively, in the DIASDEM batch script recorded during knowledge discovery, as explained in Section 4.5 on page 148:

1. *Pre-processing of text documents:* 8,408 Reuters news items assigned the topic code C181 and region code USA published between 1997/04/01 and 1997/08/19 were imported into a text archive.
2. *Iterative classification of text unit vectors:* All tasks recorded when performing eight clustering iterations in the knowledge discovery phase were edited and removed, respectively, to transform them into eight corresponding classification iterations, as introduced in Section 4.5 on page 148. The original batch script was edited exactly as explained in the context of the knowledge application phase of the Commercial Register case study in the preceding subsection. In particular, we edited the tasks related to vectorization of text units and execution of the clustering algorithm in application mode. All workbench tasks responsible for automatic ranking of text unit clusters, and automatic default labeling of acceptable clusters, as well as interactive screening of cluster ranking, and interactive review of default cluster labels were deleted in the batch script.

**Table 6.17:** Results of the Semantic XML Markup Quality Assessment in the Reuters News Case Study

Quality Measure for XML Tag Names	Knowledge Discovery Phase		Knowledge Application Phase	
	Value in Sample	Approximate 95% Confidence Interval	Value in Sample	Approximate 95% Confidence Interval
Number of Text Units	1,222		862	
Proportion of True Positives	0.5106	[0.4826; 0.5387]	0.4919	[0.4585; 0.5253]
Proportion of False Positives	0.1047	[0.0876; 0.1219]	0.1276	[0.1053; 0.1499]
Proportion of True Negatives	0.2971	[0.2714; 0.3227]	0.2610	[0.2317; 0.2903]
Proportion of False Negatives	0.0876	[0.0717; 0.1034]	0.1195	[0.0978; 0.1411]
XML Tag Name Accuracy	0.8077	[0.7856; 0.8298]	0.7529	[0.7241; 0.7817]
XML Tag Name Precision	0.8298	[0.8029; 0.8566]	0.7940	[0.7597; 0.8283]
XML Tag Name Recall	0.8536	[0.8280; 0.8793]	0.8046	[0.7707; 0.8384]

Quality Measure for XML Tag Attributes	Knowledge Discovery Phase		Knowledge Application Phase	
	Value in Sample	Approximate 95% Confidence Interval	Value in Sample	Approximate 95% Confidence Interval
True Number of Named Entities	1,441		1,036	
XML Attribute Accuracy	0.7946	[0.7737; 0.8154]	0.7712	[0.7457; 0.7968]
XML Attribute Precision	0.9030	[0.8867; 0.9193]	0.9142	[0.8956; 0.9328]
XML Attribute Recall	0.8161	[0.7958; 0.8364]	0.7934	[0.7684; 0.8185]

3. *Post-processing of discovered patterns:* The task that established the concept-based XML DTD was removed from the original knowledge discovery batch script.

Conforming to Algorithm 4.11 on page 151, the modified batch script was executed without any human intervention by DIASDEM WORKBENCH BATCH SCRIPT CLIENT. Subsequently, we drew a random sample comprising 862 (1%) of altogether 86,238 semantically marked-up text units in the application text archive to evaluate the markup quality, as discussed in Section 6.1. Analogous to the knowledge discovery phase, the author evaluated the markup quality of this random text unit sample. The assessment results are listed in columns 4 and 5 of Table 6.17. As expected when applying classification knowledge to test data, the quality measures for XML tag names and XML tag attributes are slightly worse than the results obtained in the knowledge discovery phase. With 0.95 confidence, for example, the knowledge application XML tag name accuracy is in the interval [0.7241; 0.7817] whereas this quality measure is in the interval [0.7856; 0.8298] during knowledge discovery. The entire quality assessment protocol can be electronically accessed (see Appendix A on page 148).

**Usage Scenario and Lessons Learned** Although the markup quality achieved in the Reuters M&A news case study is not as high as in the Commercial Register entries case, organizations in the finance and banking industry may clearly benefit from semantically marked-up news stories in the scenarios discussed in Section 1.3 on page 9. In particular,

the activity of retrieving information about competitors from the ever growing archive of M&A news items is largely facilitated by semantic XML markup along with appropriate XML query languages. Another case of semantic markup usage was presented by Schulz et al. (2003). Ultimately, this study aimed at predicting the effect of news stories on the share prices of the affected companies. As part of the text pre-processing phase, Schulz et al. (2003) employed the DIASDEM framework to semantically annotate the input news stories. For each discovered semantic concept, a new binary input variable was subsequently derived to make explicit the occurrence of the respective concept in individual news stories for the purpose of training a text classifier.

In addition to the five generic lessons learned in the Commercial Register case study, we finally summarize two noteworthy lessons learned in the context of marking up Reuters news items on U.S. mergers and acquisitions:

- Extract relevant domain-specific named entities! When starting a new semantic XML tagging project, we strongly recommend identifying all named entity types of particular importance. The benefits of augmenting semantic concepts and XML tag names, respectively, with truly relevant named entity types (e.g., `unit_of_company` and `equity_stake` in the M&A domain) shall nevertheless outweigh the efforts required to parameterize and perhaps to extend the named entity extractor accordingly. In general, however, solely relying on the extraction of domain-independent named entity types is an inferior approach to semantic text annotation.
- Remove noisy text units in iteration one! Unlike the highly controlled language of Commercial register entries, news stories typically do contain noisy text units that represent neither frequent nor relevant concepts. Based on the experiences gained in this case, we recommend temporarily marking noisy text units as acceptable, assigning them a label like `Z999_IrrelevantTopic`, and thereby excluding them from the input data of subsequent clustering iterations. Prior to establishing the concept-based DTD, however, noisy text units must be marked as unacceptable.

Additionally, issues to be addressed by future research and future development of DIASDEM WORKBENCH are outlined in Section 7.2.

### 6.3 Summary

This chapter has been devoted to the third phase of our specific research process described in Section 1.5. By processing two distinct real-world text archives, we have evaluated both the DIASDEM framework for semantic XML tagging of domain-specific text archives and the research prototype DIASDEM WORKBENCH. To that end, we have initially established a set of criteria that reflect different aspects of semantic XML markup quality. In particular, we have distinguished between the quality of XML tag names and the quality of XML tag attributes by defining specific measures of accuracy, precision, and recall. Finally, we have presented the entire process and the results of semantically enhancing

two text archives from the competitive intelligence domain: German Commercial Register entries and English Reuters news stories featuring U.S. mergers and acquisitions. Although both archives fulfill the prerequisites of applying our framework, they differ in the scope of vocabulary used, the inherent semantic diversity, and the variation in phrasing. Consequently, the case study results vary between excellently marked-up Commercial Register entries and acceptably tagged Reuters M&A news stories. Nevertheless, we have successfully demonstrated the applicability of our conceptual framework and the research prototype to transform large, domain-specific text archives into semantically tagged XML archives.



## 7 Conclusions

Our research on semantic XML tagging of domain-specific text archives is summarized in the next section. Section 7.1 also emphasizes the contribution of this work along the research questions posed in the introduction. To envision valuable extensions and enhancements of the DIASDEM framework, three important areas of future research are identified in Section 7.2. Finally, Section 7.3 comprises a few concluding remarks.

### 7.1 Summary and Contribution

Facing ever increasing volumes of computer-accessible textual data, semantic XML tagging of text archives creates value by providing embedded metadata. In this work, names of semantic XML tags explicitly convey informal metadata by concisely describing concepts that domain experts typically associate with marked-up text units, which represent structural document parts (e.g., sentences or paragraphs). Optionally, attributes of semantic XML tags make explicit named entities (e.g., names of companies) occurring in marked-up text units. The markup syntax is defined in a concept-based XML document type definition, and its meaning is informally specified in the DTD documentation.

Organizations benefit from exploiting semantic metadata in two broad ways. Firstly, semantic XML markup tends to improve the effectiveness and efficiency of information retrieval. Secondly, semantic XML markup facilitates applications that leverage semantic metadata in texts, such as document warehousing and information integration.

*Can techniques for knowledge discovery in textual databases be employed to convert large archives comprising domain-specific text documents of homogeneous content into semantically marked-up XML documents?*

This primary research objective, as posed in Section 1.4, has been accomplished by (i) establishing a conceptual framework for semantic XML tagging of domain-specific text archives, (ii) developing a research prototype that implements the entire framework, and (iii) evaluating the markup quality by processing real-world text archives. Collectively, the new conceptual framework, the research prototype system, and the conducted experiments constitute the contribution of this work to research in information systems.

In phase one of the two-phase framework, an interactive knowledge discovery process discovers and semantically labels concepts that occur at the text unit level of plain texts, derives a concept-based XML document type definition based on identified, frequently occurring concepts and prevailing named entity types, as well as semantically tags text

documents to enable quality assessment. In the second phase termed knowledge application, large volumes of thematically similar text documents are automatically tagged by utilizing the classification knowledge acquired in phase one.

As introduced in Section 1.4, the primary research question stated above entails the following six secondary research questions. For each of them, we henceforth outline our proposed solutions:

1. *How to discover frequently recurring thematic concepts at the text unit level?* Unsupervised learning is utilized in the pattern discovery step of our DIASDEM knowledge discovery process. Subsequent to extensive text pre-processing, clustering algorithms that satisfy framework-specific selection criteria are iteratively executed to find clusters of text unit vectors, whose members represent semantically similar text units. Unlike the conventional approach to unsupervised learning, text unit vectors are repeatedly clustered with gradually reduced input data, varying parameter settings, and possibly different clustering algorithms. In each iteration, a clustering algorithm is selected, parameterized, and executed. Framework-specific cluster quality criteria are employed to distinguish between qualitatively acceptable and unacceptable text unit clusters. Members of acceptable ones do not constitute input data to the next iteration. Due to this iterative clustering approach, both specific and general, as well as more and less frequent concepts can be discovered.
2. *How to assign semantic, content-descriptive labels to identified thematic concepts?* A semantic cluster label concisely describes the concept that domain experts typically associate with text units assigned to the corresponding cluster. To facilitate the creation of high-quality semantic markup, we intentionally take a semi-automated approach to cluster labeling. To automatically generate the default label for a qualitatively acceptable text unit cluster, all dominant descriptors occurring therein are identified, ordered by decreasing descriptor dominance, concatenated using the underscore character, and prefixed with "DEFAULT\_". Subsequently, label suggestions are interactively approved or rejected (i.e., corrected) by the domain expert. Ultimately, labels of acceptable clusters serve as names of semantic XML tags.
3. *How to extract domain-specific named entities from text units?* Subsection 2.2.2 has revealed that this research question can be considered solved by the information extraction research community for the scope of this work. Machine learning and fully automated techniques significantly reduced the necessary efforts of knowledge engineers to establish and maintain extensive repositories of information extraction rules. Consequently, we have not addressed named entity extraction in this work. Instead, we have proposed utilizing existing techniques to extract named entities.
4. *How to establish a domain-specific, concept-based XML document type definition that reflects the thematic structure on the text unit level by enumerating frequently occurring thematic concepts and associated named entity types?* During knowledge

discovery, a conceptual document structure is established in the post-processing step by (i) enumerating the distinct semantic concepts identified during pattern discovery, (ii) collecting the concept-specific set of extracted named entity types for each semantic concept, and finally (iii) applying a user-supplied attribute support threshold to remove noise from the concept-specific lists of named entity types. Creating the concept-based XML DTD from a conceptual document structure involves five major steps, each of which performs basic string concatenations.

5. *How to automatically convert text documents into semantically annotated XML documents by marking up text units according to the concept-based XML DTD?* Our KDT process discovers classification knowledge in the form of frequently occurring semantic concepts at the text unit level. This classification knowledge is exploited to transform input text archives into semantically tagged XML documents that conform to a common, concept-based XML document type definition. In addition, a parameterized KDT process flow is created in the knowledge discovery phase. If this sequence of parameterized, non-interactive KDT algorithms is executed in the knowledge application phase, thematically similar text documents are automatically marked up without any human intervention at all.
6. *How to evaluate the quality of automatically created semantic XML markup?* Quality assessment is performed from the perspective of users of semantically marked-up XML documents who take the concept-based DTD as given. The distinction between correct and erroneous markup requires comparing a random sample of automatically generated markup with the correct markup that would have been assigned by a human annotator, who is only allowed to select tags from the concept-based DTD. To account for different aspects of semantic XML markup quality, framework-specific measures of accuracy, precision, and recall have been proposed to assess the quality of XML tag names and XML tag attributes.

Our research prototype system DIASDEM WORKBENCH supports the entire framework for semantic tagging of domain-specific text archives, namely, the interactive knowledge discovery phase and the automated, batch-oriented knowledge application phase. The notion of KDT process flows as a link between these two phases is operationalized by a workbench-specific scripting language. The Java-based client/server-application comprises a server component that executes DIASDEM WORKBENCH tasks, each of which encapsulates a KDT algorithm or a group of similar ones, as requested by client applications. A graphical user interface enables domain and KDT experts to interactively work in the semi-automated knowledge discovery phase. The knowledge application phase of our framework is also supported by a command-line batch client.

We conducted experiments with two real-world text archives from the competitive intelligence domain: German Commercial Register entries and English Reuters news stories featuring U.S. mergers and acquisitions. These archives fulfill the prerequisites of applying our framework, but they differ in the scope of vocabulary used, the inherent semantic

diversity, and the variation in phrasing. Hence, the case study results vary between excellently tagged Commercial Register entries and acceptably marked-up Reuters M&A news stories. To sum up, we successfully demonstrated the applicability of our framework and the research prototype DIASDEM WORKBENCH to transform domain-specific text archives into semantically marked-up XML documents. Partial results of the experimental evaluation are electronically accessible, as detailed in Appendix A.

## 7.2 Future Research

In this section, recommendations are given to guide forthcoming research on semantic XML tagging of domain-specific text archives. The areas of future research considered worth exploring are (i) methods of structuring the concept-based XML document type definition, (ii) temporal aspects of discovered knowledge, and (iii) approaches to automating the knowledge discovery phase of our framework. These areas of future research are motivated in the following three subsections.

### 7.2.1 Structuring the Concept-Based XML DTD

Our proposed framework is intentionally focused on creating semantically marked-up XML documents with non-overlapping tags. In addition, the concept-based XML DTD does not impose any structural constraints on the sequence of XML tags within marked-up documents. To overcome this initial limitation, two questions are worth being addressed by future research. Firstly, how to further structure the currently enumerative, concept-based XML document type definition? Secondly, how to address the shortcomings inherent to XML document type definitions by replacing them, for instance, by XML Schema (cf. World Wide Web Consortium, 2001) definitions.

Concerning the first issue of structure discovery, nested XML tags can, for example, be introduced by relaxing the restriction of one text unit layer per document and introducing the notion of multi-level annotation (cf. Bayerl et al., 2003a). One way of creating nested semantic XML tags is to decompose texts into multiple text unit layers and to semantically tag them on different levels of structural granularity (e.g., sentences, paragraphs, and sections). This approach requires modifying our notion of conceptual document structures. We have discussed approaches to schema discovery in marked-up texts in Section 2.4. To incorporate a suitable method into our framework, one needs to assess the applicability of existing schema discovery techniques to infer a structured DTD from semantically marked-up XML documents with ordered and/or nested tags.

Furthermore, knowledge discovery techniques can be utilized to discover, for example, groups of structurally similar XML documents (e.g., cf. Helmer, 2007) or frequently occurring sequences of semantic XML tags. To make explicit the most likely ordering of semantic tags, Winkler and Spiliopoulou (2002) outlined initial thoughts on establishing a probabilistic DTD as part of the DIASDEM framework. A probabilistic DTD takes the

fact into consideration that DTD elements are derived by data mining techniques, which are subject to errors, and thus includes statistical properties of DTD elements.

With respect to the second issue of structure representation, XML document type definitions “have some deficiencies that may prevent certain important classes of applications from accomplishing their goals” (Melton and Buxton, 2006, p. 100). These deficiencies include the inability to specify certain types of limitations on the content (e.g., the sequence of child elements in an element defined to have mixed content) as well as the inability to define the data types required for both attribute values and element content. XML Schema (see World Wide Web Consortium, 2001) addresses these limitations, which are particularly relevant in the context of information integration, by supporting restrictive structural and data type constraints on XML documents. Incorporating the notion of concept-based XML Schema definitions in our framework requires a modification of conceptual document structures and research into deriving a concept-based XML Schema definition from semantically marked-up XML documents (e.g., cf. Hegewald et al., 2006; Bex et al., 2007) that comprise ordered and/or nested XML tags.

Finally, combining our notion of informal semantic annotation and research into learning ontologies as formal representations of semantics from text (Fensel, 2004; Cimiano, 2006) is another promising research direction. This research direction aims at providing explicit and formal semantic XML markup to facilitate automated knowledge sharing and reuse between various human and artificial agents (cf. Fensel, 2004, p. 4).

### 7.2.2 Temporal Aspects of Discovered Knowledge

Our framework is designed to identify a priori unknown, frequently occurring concepts at the text unit level by means of iterative clustering in the initial knowledge discovery phase. Subsequently, the discovered classification knowledge is utilized for semantic XML tagging in the automated knowledge application phase. In classification theory, it is typically assumed that labeled training cases are randomly, independently, and identically drawn samples from a stationary test distribution (cf. Weiss et al., 2005, p. 53). As we employ unsupervised learning to initially discover both concepts and labeled training cases, the training text archive is implicitly assumed to be a randomly, independently, and identically drawn sample from the stationary application archive.

As Weiss et al. (2005) emphasized, however, real-world text collections tend to change over time. Moreover, commercial applications, such as semantic XML markup to gain competitive intelligence, often require to apply a trained classifier to make predictions on a stream of future samples that may vary over time (Forman, 2006, p. 252). This time-varying component of classification tasks is often referred to as concept drift (e.g., see references of Forman, 2006). Nevertheless, as Weiss et al. concluded, it can often be expected that the nature of documents remains relatively stable over a time frame, which is sometimes relatively short, such that classification methods are applicable. Over longer periods, the learning process needs to be repeated on the basis of an updated random sample of training documents, which adequately reflects the topic changes over time.

Kriegel et al. (2007, pp. 90–91) discussed future trends in data mining and identified, among other focus research areas, temporal aspects of discovered knowledge as an important challenge to the research community. Incorporating the notion of concept drift into our framework for semantic XML tagging constitutes a promising research direction as well. Potential research questions include, but are not limited to, (i) the identification of the presence of concept drift in domain-specific text archives during knowledge discovery and application, (ii) the design of methods to measure the effect of drifting concepts on markup quality during knowledge application, and (iii) the development of techniques to automatically trigger an update of classification knowledge instead of repeating the entire knowledge discovery phase.

Aimed at modeling and tracking temporal cluster transitions, Spiliopoulou et al. (2006) presented the framework MONIC for monitoring cluster transitions. The authors employed DIASDEM WORKBENCH in their case study.

### 7.2.3 Towards Automated Knowledge Discovery

The DIASDEM framework comprises an intentionally semi-automated and interactive knowledge discovery phase. As detailed in Section 4.6, six types of human domain knowledge are recommended for incorporation into the KDT process. In particular, three text pre-processing tasks and three iterative text unit clustering tasks require human expertise to ensure a high markup quality during the automated knowledge application phase. To overcome the necessity to allocate often scarce and mostly costly human resources to semantic XML tagging projects, substituting human efforts by automated pre-processing and iterative clustering techniques is one important direction for forthcoming research. The quality of semantic XML markup, however, must not be worse than the quality obtained when employing the semi-automated KDT approach.

When discussing future trends in data mining, Kriegel et al. (2007, p. 92) concluded that there will be “a lot to gain and a lot to study in preprocessing for data mining in the next years.” In our framework, establishing and maintaining a controlled vocabulary typically requires domain-specific expertise. Hence, one important aspect of automating the text pre-processing step during knowledge discovery is related to solving the open research challenge (cf. Zhou, 2007, pp. 247–249) of automatically learning and/or updating high-quality taxonomies, thesauri, and ontologies that are qualitatively comparable to their human-built counterparts. Another promising direction of future research aims at enhancing the text pre-processing by means of anaphora resolution as part of named entity extraction, automated and highly precise word sense disambiguation, as well as the correction of spelling errors and line-breaking hyphens. Given the results of our case studies, the quality of semantic XML tag names and XML tag attributes is likely to increase when advanced, automated NLP methods are incorporated into our framework.

Finally, reducing the human expert efforts in the iterative clustering step of the knowledge discovery phase constitutes another interesting research area. One future data mining trend, as stated by Kriegel et al. (2007, p. 93), is “increased usability” of knowledge discov-

ery algorithms. In our context, research on automating the selection and parameterization of clustering algorithms in each clustering iteration is a rewarding challenge. This involves the automated discovery of both specific and general, as well as more and less frequent concepts. In particular, this research challenge is framed by the prerequisite that results of the iterative text unit clustering have to be meaningful in the eyes of human domain experts. Besides automatically identifying qualitatively acceptable text unit clusters, overcoming the suggested interactive review of automatically generated semantic cluster labels poses another research question: How to automatically generate truly meaningful labels, like the ones created by humans, instead of merely extracting and concatenating content-descriptive terms from text units assigned to a cluster? Newman et al. (2007) and Mei et al. (2007) presented initial thoughts on this issue. After all, cluster labels serve as semantic XML tag names. Therefore, they have to be meaningful, comprehensible, and self-explanatory to humans in describing the aboutness, the topicality, or the subject of semantically marked-up text units.

### **7.3 Concluding Remarks**

When researching the effect of text search and analytics on corporate business intelligence systems, Russom (2007, p. 31) concluded that “semi-structured and unstructured data sources for data warehouses will increase dramatically in the next three years (up 21–81% depending on source type).” In this context, our work on utilizing knowledge discovery techniques to transform domain-specific text archives into semantically marked-up XML documents contributes towards adding value to huge volumes of textual data. Thus, our approach makes a modest contribution to alleviate the hunger for knowledge of today’s information society in times of information overload (cf. Naisbitt, 1982, pp. 11–38).

Based on the systems development research methodology proposed by Nunamaker et al. (1991), this work was intentionally conducted in a highly multi-disciplinary way to take both an information systems and a knowledge discovery perspective on semantic text annotation. Consequently, we preferred research breadth over depth when establishing a conceptual framework for semantic XML tagging of domain-specific texts, developing a research prototype, and evaluating the markup quality in two case studies. Diving into the depth of specific topics, such as improving named entity recognition or including advanced NLP pre-processing, is one of many starting points for future research.



# A Contents of the Supplementary Web Site

Visit <http://ka.rsten-winkler.de/thesis> to access the supplementary Web site. Firstly, the research prototype DIASDEM WORKBENCH 2.2 is available for download via this Web site under the GNU general public license. Secondly, the most important results of both case studies described in detail in Section 6.2 are provided. For each case study, the following documents are accessible using the user name `guest` and the password `Welcome`:

## 1. Results of the knowledge discovery phase

- Automatically generated documentation of the concept-based XML document type definition
- Automatically generated concept-based XML document type definition
- Protocol of the manual evaluation of tagging quality
- Results of the manual evaluation of tagging quality
- Random sample of semantically marked-up XML documents and the corresponding auxiliary DIASDEM documents (cf. Winkler, 2007, pp. 18–19)

## 2. Results of the knowledge application phase

- Protocol of the manual evaluation of tagging quality
- Results of the manual evaluation of tagging quality
- Random sample of semantically marked-up XML documents and the corresponding auxiliary DIASDEM documents (cf. Winkler, 2007, pp. 18–19)



# B Specifications of Abstract Data Types

## B.1 ADT Notation, Primitive Data Types, and Arrays

In the remainder of this appendix chapter, abstract data types are specified using abstract models (Dale and Walker, 1996, pp. 13–14). Abstract models specify the semantics of the abstract data type being defined in terms of operations provided by primitive data types or other ADTs. Concretely, operations are defined by specifying the signatures<sup>1</sup> of the respective procedures or functions. For each operation, preconditions state what the respective procedure or functions can assume on entry. If the preconditions hold before the execution, the respective operation will terminate and all operation-specific postconditions will hold after the execution (Loeckx et al., 1996, pp. 13–14).

Let  $S := \{s_1, s_2, \dots, s_{|S|}\}$  denote an arbitrary, finite alphabet of symbols and characters, respectively. In algorithms and specifications of abstract data types, we utilize the primitive data types Boolean, Integer, Real, and Char that are defined as follows:

- $x$ : Boolean  $\leftrightarrow x \in \{\text{true}, \text{false}\}$
- $x$ : Integer  $\leftrightarrow x \in \mathbb{N} \cup \{0, -1, -2, \dots\}$
- $x$ : Real  $\leftrightarrow x \in \mathbb{R}$
- $x$ : Char  $\leftrightarrow x \in S$

Built into most programming languages, arrays<sup>2</sup> are named collections of typed components, each of which can be accessed by specifying its position in the collection (Dale and Walker, 1996, p. 194). For reasons of clarity and simplicity, arrays are considered as primitive data types in this work, independent of whether their components are instances

---

<sup>1</sup>Notation: Let  $T_i$ , where  $i \in \mathbb{N}$ , denote a primitive or abstract data type defined in this appendix chapter. The signature of function  $f$ , which has  $j \in \mathbb{N} \cup \{0\}$  input parameters  $x_1, x_2, \dots, x_j$  and returns a value of type  $T_{j+1}$ , is denoted by  $f(x_1: T_1, x_2: T_2, \dots, x_j: T_j): T_{j+1}$ . Analogously,  $p(x_1: T_1, x_2: T_2, \dots, x_j: T_j)$  denotes the signature of procedure  $p$ , which has  $j$  input parameters  $x_1, x_2, \dots, x_j$ .

<sup>2</sup>Notation: Let  $T$  denote a primitive or abstract data type defined in this appendix chapter. The primitive data type that represents an array comprising components of type  $T$  is denoted by  $T[]$ . Let  $x: T[]$  be a variable that represents an array of type  $T$ . A new array of size  $i \in \mathbb{N}$  is instantiated by the expression  $x := T[i]$ . Individual array components are denoted by  $x[j]$ , where  $j \in \mathbb{N}$  and  $j \leq i$ . The constant  $x.\text{size}$  denotes the invariant size of the instantiated array  $x$ .

of primitive or abstract data types. The size of an array and the number of components, respectively, cannot be altered once it is instantiated.

## B.2 ADT for Strings

**Abstract Data Type:** String

**Data:** String  $s$  represents either an empty sequence  $s := \varepsilon$  or a non-empty sequence of characters  $s := \langle s_1, s_1, \dots, s_{|s}| \rangle$ , where  $s_i: \text{Char}$  and  $i = 1, 2, \dots, |s|$ .

**Operations:**

- $\text{create}(s_{\text{init}}: \langle s_1, s_1, \dots, s_i \rangle)$  constructs a new string from a character literal. Preconditions:  $i \in \mathbb{N}$  and  $s_j: \text{Char}$ , where  $j = 1, 2, \dots, i$ . Postcondition:  $s := s_{\text{init}}$ . Note: To simplify notation, strings can also be constructed by assigning a character literal to a String-typed variable like  $s: \text{String} := \text{"character literal"}$ .
- $\text{char}(i: \text{Integer}): \text{Char}$  returns the  $i$ th character of the string. Precondition:  $i = 1, 2, \dots, |s|$ . Postcondition:  $\text{char}(i) := s_i$ .
- $\text{chars}(i: \text{Integer}, j: \text{Integer}): \text{String}$  returns a substring. Preconditions:  $i = 1, 2, \dots, |s|$  and  $j = i, i + 1, \dots, |s|$ . Postcondition:  $\text{chars}(i, j) := \langle s_i, s_{i+1}, \dots, s_j \rangle$ .
- $\text{length}(): \text{Integer}$  returns the length of the string. Precondition: None. Postcondition: If the string is an empty sequence,  $\text{length}() := 0$ . Otherwise,  $\text{length}() := |s|$ .

## B.3 ADT for Vectors

**Abstract Data Type:** Vector

**Data:** Vector  $\mathbf{v} := [v_1, v_2, \dots, v_n]^T$  is an  $n$ -dimensional vector, where  $\mathbf{v} \in \mathbb{R}^n$  and  $n \in \mathbb{N}$ .

**Operations:**

- $\text{create}(v_{\text{init}}: \text{Real}[])$  constructs a new vector. Precondition:  $v_{\text{init}}.\text{size} > 0$ . Postcondition:  $n := v_{\text{init}}.\text{size}$  and  $v_1 := v_{\text{init}}[1], v_2 := v_{\text{init}}[2], \dots, v_n := v_{\text{init}}[n]$ .
- $\text{component}(i: \text{Integer}): \text{Real}$  returns the  $i$ th vector component. Precondition:  $i = 1, 2, \dots, n$ . Postcondition:  $\text{component}(i) := v_i$ .
- $\text{dimensions}(): \text{Integer}$  returns the number of vector dimensions. Precondition: None. Postcondition:  $\text{dimensions}() := n$ .

## B.4 ADT for Text Documents

**Abstract Data Type:** TextDocument

**Data:** Text document  $\check{t}: \text{String}$  is a sequence of characters.

**Operations:**

- `create(̃tinit: String)` constructs a new text document. Precondition: None. Postcondition:  $\check{t} := \check{t}_{init}$ .
- `char(i: Integer): Char` returns the  $i$ th character of this text document. Precondition:  $i = 1, 2, \dots, \check{t}.length()$ . Postcondition:  $\text{char}(i) := \check{t}.char(i)$ .
- `chars(i: Integer, j: Integer): String`. Preconditions:  $i = 1, 2, \dots, \check{t}.length()$  and  $j = i, i + 1, \dots, \check{t}.length()$ . Postcondition:  $\text{chars}(i, j) := \check{t}.chars(i, j)$ .
- `content(): String` returns the textual content of this text document. Precondition: None. Postcondition:  $\text{content}() := \check{t}$ .
- `length(): Integer` returns the length of this text document. Precondition: None. Postcondition:  $\text{length}() := \check{t}.length()$ .

## B.5 ADT for Text Archives

**Abstract Data Type:** TextArchive

**Data:** Text archive  $\check{a} := \langle \check{t}_1, \check{t}_2, \dots, \check{t}_{|\check{a}|} \rangle$  is a sequence of not necessarily distinct text documents such that  $\check{t}_i: \text{TextDocument}$ , where  $i = 1, 2, \dots, |\check{a}|$ .

**Operations:**

- `create()` constructs a new, empty text archive. Precondition: None. Postcondition:  $\check{a} := \varepsilon$ .
- `appendTextDocument(̃tnew: TextDocument)` appends a text document to the end of this sequence of text documents. Precondition: None. Postcondition: Let  $i := |\check{a}|$  denote the length of this sequence prior to appending  $\check{t}_{new}$  such that  $\check{t}_{i+1} := \check{t}_{new}$ . If  $i = 0$ ,  $\check{a} := \langle \check{t}_{i+1} \rangle$ . Otherwise,  $\check{a} := \langle \check{t}_1, \check{t}_2, \dots, \check{t}_i, \check{t}_{i+1} \rangle$ .
- `size(): Integer` returns the number of text documents in this text archive. Precondition: None. Postcondition:  $\text{size}() := |\check{a}|$ .
- `textDocument(i: Integer): TextDocument` returns the  $i$ th text document of this text archive. Precondition:  $i = 1, 2, \dots, |\check{a}|$ . Postcondition:  $\text{textDocument}(i) := \check{t}_i$ .

## B.6 ADT for Text Units

**Abstract Data Type:** TextUnit

**Data:** Text unit  $\check{u}: \text{String}$  is a sequence of contiguous characters  $\check{u} := \check{t}.chars(i, j)$ , where  $\check{t}: \text{TextDocument}$  is the parent text document,  $i = 1, 2, \dots, \check{t}.length()$ , and  $j = i, i + 1, \dots, \check{t}.length()$ .

**Operations:**

- `create(̃tinit: TextDocument, iinit: Integer, jinit: Integer)` constructs a new text unit. Preconditions:  $i = 1, 2, \dots, \check{t}_{init}.length()$  and  $j = i, i + 1, \dots, \check{t}_{init}.length()$ . Postconditions:  $\check{u} := \check{t}_{init}.chars(i_{init}, j_{init}), i := i_{init}, j := j_{init}$ .

- `char(i: Integer): Char` returns the  $i$ th character of this text unit. Precondition:  $i = 1, 2, \dots, \check{u}.length()$ . Postcondition: `char(i) :=  $\check{u}.char(i)$` .
- `chars(i: Integer, j: Integer): String`. Preconditions:  $i = 1, 2, \dots, \check{u}.length()$  and  $j = i, i + 1, \dots, \check{u}.length()$ . Postcondition: `chars(i, j) :=  $\check{u}.chars(i, j)$` .
- `content(): String` returns the textual content of this text unit. Precondition: None. Postcondition: `content() :=  $\check{u}$` .
- `endIndex(): Integer` returns the index  $j$  of the last character of this text unit in its parent text document. Precondition:  $\check{u} \neq \varepsilon$ . Postcondition: `endIndex() :=  $j$` .
- `length(): Integer` returns the length of this text unit. Precondition: None. Postcondition: `length() :=  $\check{u}.length()$` .
- `startIndex(): Integer` returns the index  $i$  of the first character of this text unit in its parent text document. Precondition:  $\check{u} \neq \varepsilon$ . Postcondition: `startIndex() :=  $i$` .

## B.7 ADT for Text Unit Layers

**Abstract Data Type:** TextUnitLayer

**Data:** Text unit layer  $\check{r} := \langle \check{u}_1, \check{u}_2, \dots, \check{u}_{|\check{r}|} \rangle$  is a sequence of not necessarily contiguous, non-overlapping text units in the parent text document such that  $\check{u}_i: \text{TextUnit}$ , where  $i = 1, 2, \dots, |\check{r}|$ .

**Operations:**

- `create()` constructs a new, empty text unit layer. Precondition: None. Postcondition:  `$\check{r} := \varepsilon$` .
- `create( $\check{t}_{\text{init}}: \text{TextDocument}$ )` constructs a new text unit layer by decomposing text document  $\check{t}_{\text{init}}$  into  $|\check{r}|$  structural text units  $\langle \check{u}_1, \check{u}_2, \dots, \check{u}_{|\check{r}|} \rangle$  whose granularity depends on the concrete implementation of this constructor. Precondition: None. Postcondition:  `$\check{r} := \langle \check{u}_1, \check{u}_2, \dots, \check{u}_{|\check{r}|} \rangle$` .
- `size(): Integer` returns the number of text units in this text unit layer. Precondition: None. Postcondition: `size() :=  $|\check{r}|$` .
- `textUnit(i: Integer): TextUnit` returns the  $i$ th text unit of this text unit layer. Precondition:  $i = 1, 2, \dots, |\check{r}|$ . Postcondition: `textUnit(i) :=  $\check{u}_i$` .

## B.8 ADT for Concepts

**Abstract Data Type:** Concept

**Data:** Represented by the symbol  $\check{o}: \text{String}$ , a concept is a mental reference or thought that domain experts typically associate with a group of semantically similar text units. A concept is associated with the alphanumeric label  $\check{l}: \text{String}$  that concisely describes the mental reference or thought.

**Operations:**

- `create(ōinit: String, l̃init: String)` constructs a new concept. Precondition: None. Postconditions:  $\check{o} := \check{o}_{init}$  and  $\check{l} := \check{l}_{init}$ .
- `label(): String` returns the alphanumeric label of this concept. Precondition: None. Postcondition:  $label() := \check{l}$ .
- `symbol(): String` returns the symbol of this concept. Precondition: None. Postcondition:  $symbol() := \check{o}$ .

## B.9 ADT for Sets of Concepts

**Abstract Data Type:** SetOfConcepts

**Data:** Set of concepts  $O := \{o_1, o_2, \dots, o_{|O|}\}$  contains concepts such that  $o_i: \text{Concept}$ , where  $i = 1, 2, \dots, |O|$ .

**Operations:**

- `create()` constructs a new, empty set of concepts. Precondition: None. Postcondition:  $O = \emptyset$ .
- `add(onew: Concept)` adds a concept to this set of concepts. Precondition: None. Postcondition:  $O := O \cup o_{new}$ .
- `elements(): Concept[]` returns an array comprising all concepts contained in this set of concepts or null if this set is empty. Precondition: None. Postcondition: If  $O = \emptyset$ , `elements() := null`. Otherwise, `elements() := x`, where  $x: \text{Concept}[] := \text{Concept}[|O|]$  and  $x[1] := o_1, x[2] := o_2, \dots, x[|O|] := o_{|O|}$ .
- `size(): Integer` returns the number of concepts in this set of concepts. Precondition: None. Postcondition:  $size() := |O|$ .

## B.10 ADT for Named Entity Types

**Abstract Data Type:** NamedEntityType

**Data:** Represented by the symbol  $\check{p}: \text{String}$ , a named entity type represents an abstract class or a generic numerical expression. A named entity type is associated with an alphanumeric label  $\check{l}: \text{String}$  that concisely describes the abstract class or a generic numerical expression.

**Operations:**

- `create(p̃init: String, l̃init: String)` constructs a new named entity type. Precondition: None. Postconditions:  $\check{p} := \check{p}_{init}$  and  $\check{l} := \check{l}_{init}$ .
- `label(): String` returns the alphanumeric label of this named entity type. Precondition: None. Postcondition:  $label() := \check{l}$ .
- `symbol(): String` returns the symbol of this named entity type. Precondition: None. Postcondition:  $symbol() := \check{p}$ .

## B.11 ADT for Sets of Named Entity Types

**Abstract Data Type:** SetOfNamedEntityType

**Data:** Set of named entity types  $P := \{p_1, p_2, \dots, p_{|P|}\}$  contains named entity types such that  $p_i: \text{NamedEntityType}$ , where  $i = 1, 2, \dots, |P|$ .

**Operations:**

- `create()` constructs a new, empty set of named entity types. Precondition: None. Postcondition:  $P = \emptyset$ .
- `create(Pinit : NamedEntityType[])` constructs a new set of named entity types. Precondition: None. Postcondition:  $P = \{p \mid p: \text{NamedEntityType} := P_{\text{init}}[i] \ \forall i = 1, 2, \dots, P_{\text{init}}.\text{size}\}$ .
- `add(pnew: NamedEntityType)` adds a named entity type to this set of named entity types. Precondition: None. Postcondition:  $P := P \cup p_{\text{new}}$ .
- `elements(): NamedEntityType[]` returns an array comprising all named entities types contained in this set of named entities types or null if this set is empty. Precondition: None. Postcondition: If  $P = \emptyset$ , `elements()` := null. Otherwise, `elements()` :=  $x$ , where  $x: \text{NamedEntityType}[] := \text{NamedEntityType}[|P|]$  and  $x[1] := p_1, x[2] := p_2, \dots, x[|P|] := p_{|P|}$ .
- `size(): Integer` returns the number of named entities types in this set of named entities types. Precondition: None. Postcondition: `size()` :=  $|P|$ .

## B.12 ADT for Named Entities

**Abstract Data Type:** NamedEntity

**Data:** Named entity  $e := (p: \text{NamedEntityType}, \check{e}: \text{String})$  is a 2-tuple comprising named entity type  $p$  and the canonical form  $\check{e}$  of the instantiated named entity type  $p$ .

**Operations:**

- `create(pinit: NameEntityType, einit: String)` constructs a new named entity. Precondition: None. Postconditions:  $p := p_{\text{init}}$  and  $\check{e} := \check{e}_{\text{init}}$ .
- `label(): String` returns the alphanumeric label of named entity type  $p$ . Precondition: None. Postcondition: `label()` :=  $p.\text{label}()$ .
- `type(): NamedEntityType` returns the type of this named entity. Precondition: None. Postcondition: `type()` :=  $p$ .
- `value(): String` returns the canonical form of this named entity. Precondition: None. Postcondition: `value()` :=  $\check{e}$ .

## B.13 ADT for Sets of Named Entities

**Abstract Data Type:** SetOfNamedEntities

**Data:** Set of named entities  $E := \{e_1, e_2, \dots, e_{|E|}\}$  contains named entities such that  $e_i$ : NamedEntity, where  $i = 1, 2, \dots, |E|$ .

**Operations:**

- `create()` constructs a new, empty set of named entities. Precondition: None. Postcondition:  $E = \emptyset$ .
- `add( $e_{\text{new}}$ : NamedEntity)` adds a named entity to this set of named entities. Precondition: None. Postcondition:  $E := E \cup e_{\text{new}}$ .
- `elements(): NamedEntity[]` returns an array comprising all named entities contained in this set of named entities or null if this set is empty. Precondition: None. Postcondition: If  $E = \emptyset$ , `elements()` := null. Otherwise, `elements()` :=  $x$ , where  $x$ : `NamedEntity[]` := `NamedEntity[ |E| ]` and  $x[1] := e_1, x[2] := e_2, \dots, x[|E|] := e_{|E|}$ .
- `size(): Integer` returns the number of named entities in this set of named entities. Precondition: None. Postcondition: `size()` :=  $|E|$ .

## B.14 ADT for Semantically Marked-Up Text Units

**Abstract Data Type:** SmuTextUnit

**Data:** Semantically marked-up text unit  $\hat{u} := (\check{u}: \text{TextUnit}, o: \text{Concept}, E: \text{SetOfNamedEntities})$  is a 3-tuple comprising the original text unit  $\check{u}$ , the domain-specific concept  $o$  that domain experts typically associate with text units similar to text unit  $\check{u}$ , and the set of named entities  $E$  extracted from text unit  $\check{u}$ .

**Operations:**

- `create( $\check{u}_{\text{init}}$ : TextUnit,  $o_{\text{init}}$ : Concept,  $E_{\text{init}}$ : SetOfNamedEntities)` constructs a new semantically marked-up text unit. Precondition: None. Postconditions:  $\check{u} := \check{u}_{\text{init}}$ ,  $o := o_{\text{init}}$ , and  $E := E_{\text{init}}$ .
- `concept(): Concept` returns the concept associated with this text unit. Precondition: None. Postcondition: `concept()` :=  $o$ .
- `distinctNamedEntityType(): NamedEntityType[]` returns an array containing the distinct named entity types identified in this text unit. Precondition: None. Postcondition: `distinctNamedEntityType()` := `namedEntityType().elements()`.
- `namedEntities(): SetOfNamedEntities` returns the set of named entities extracted from this text unit. Precondition: None. Postcondition: `namedEntities()` :=  $E$ .
- `namedEntityType(): SetOfNamedEntityType` returns the set of distinct named entity types extracted from this text unit. Precondition: None. Postcondition: If  $P = \emptyset$ , `namedEntityType()` := null. Otherwise, let  $x$ : `SetOfNamedEntityType` as well as  $y$ : `NamedEntityType[]` := `NamedEntityType[ |P| ]` and  $y[1] := p_1, y[2] := p_2, \dots, y[|P|] := p_{|P|}$  such that `namedEntityType()` :=  $x.create(y)$ .

- `textUnit()`: `TextUnit` returns the marked-up text unit. Precondition: None. Postcondition: `textUnit() := ũ`.

## B.15 ADT for Semantically Marked-Up Text Unit Layers

**Abstract Data Type:** `SmuTextUnitLayer`

**Data:** Given text unit layer  $\check{r}$ : `TextUnitLayer`, a semantically marked-up text unit layer  $\hat{r} := \langle \hat{u}_1, \hat{u}_2, \dots, \hat{u}_{|\hat{r}|} \rangle$  is a sequence of semantically marked-up text units such that  $\hat{u}_i$ : `SmuTextUnit` and  $\hat{u}_i.\text{textUnit}() = \check{r}.\text{textUnit}(i)$ , where  $i = 1, 2, \dots, \check{r}.\text{size}()$ .

**Operations:**

- `create()` constructs a new, empty semantically marked-up text unit layer. Precondition: None. Postcondition:  $\hat{r} := \varepsilon$ .
- `appendSmuTextUnit( $\hat{u}_{\text{new}}$ : SmuTextUnit)` appends a semantically marked-up text unit to the end of this sequence of semantically marked-up text units. Precondition: None. Postcondition: Let  $i := |\hat{r}|$  denote the length of this sequence prior to appending  $\hat{u}_{\text{new}}$  such that  $\hat{u}_{i+1} := \hat{u}_{\text{new}}$ . If  $i = 0$ ,  $\hat{r} := \langle \hat{u}_{i+1} \rangle$ . Otherwise,  $\hat{r} := \langle \hat{u}_1, \hat{u}_2, \dots, \hat{u}_i, \hat{u}_{i+1} \rangle$ .
- `size()`: Integer returns the number of semantically marked-up text units in this semantically marked-up text unit layer. Precondition: None. Postcondition: `size() := |\hat{r}|`.
- `smuTextUnit( $i$ : Integer)`: `SmuTextUnit` returns the  $i$ th semantically marked-up text unit of this semantically marked-up text unit layer. Precondition:  $i = 1, 2, \dots, |\hat{r}|$ . Postcondition: `smuTextUnit( $i$ ) := \hat{u}_i`.

## B.16 ADT for Semantically Marked-Up Text Documents

**Abstract Data Type:** `SmuTextDocument`

**Data:** Given the text document  $\check{t}$ : `TextDocument`, a semantically marked-up text document  $\hat{t} := (\check{t}: \text{TextDocument}, \hat{r}: \text{SmuTextUnitLayer})$  is a 2-tuple comprising the text document  $\check{t}$  and the semantically marked-up text unit layer  $\hat{r}$  that represents a decomposition of text document  $\check{t}$  into semantically marked-up text units.

**Operations:**

- `create( $\check{t}_{\text{init}}$ : TextDocument,  $\hat{r}_{\text{init}}$ : SmuTextUnitLayer)` constructs a new semantically marked-up text document. Precondition: None. Postconditions:  $\check{t} := \check{t}_{\text{init}}$  and  $\hat{r} := \hat{r}_{\text{init}}$ .
- `smuTextUnitLayer()`: `SmuTextUnitLayer` returns the semantically marked-up text unit layer. Precondition: None. Postcondition: `smuTextUnitLayer() := \hat{r}`.
- `textDocument()`: `TextDocument` returns the semantically marked-up text document. Precondition: None. Postcondition: `textDocument() := \check{t}`.

## B.17 ADT for Semantically Marked-Up Text Archives

**Abstract Data Type:** SmuTextArchive

**Data:** Given the text archive  $\check{a} : \text{TextArchive}$ , a semantically marked-up text archive  $\hat{a} := \langle \hat{t}_1, \hat{t}_2, \dots, \hat{t}_{|\hat{a}|} \rangle$  is a sequence of not necessarily distinct semantically marked-up text documents such that  $\hat{t}_i : \text{SmuTextDocument}$  and  $\hat{t}_i.\text{textDocument}() = \check{a}.\text{textDocument}(i)$ , where  $i = 1, 2, \dots, \check{a}.\text{size}()$ .

**Operations:**

- `create()` constructs a new, empty semantically marked-up text archive. Precondition: None. Postcondition:  $\hat{a} := \varepsilon$ .
- `appendSmuTextDocument( $\hat{t}_{\text{new}} : \text{SmuTextDocument}$ )` appends a semantically marked-up text document to the end of this sequence of semantically marked-up text documents. Precondition: None. Postcondition: Let  $i := |\hat{a}|$  denote the length of this sequence prior to appending  $\hat{t}_{\text{new}}$  such that  $\hat{t}_{i+1} := \hat{t}_{\text{new}}$ . If  $i = 0$ ,  $\hat{a} := \langle \hat{t}_{i+1} \rangle$ . Otherwise,  $\hat{a} := \langle \hat{t}_1, \hat{t}_2, \dots, \hat{t}_i, \hat{t}_{i+1} \rangle$ .
- `size()`: Integer returns the number of semantically marked-up text documents in this semantically marked-up text archive. Precondition: None. Postcondition: `size()` :=  $|\hat{a}|$ .
- `smuTextDocument( $i : \text{Integer}$ )`: `SmuTextDocument` returns the  $i$ th semantically marked-up text document of this semantically marked-up text archive. Precondition:  $i = 1, 2, \dots, |\hat{a}|$ . Postcondition: `smuTextDocument( $i$ )` :=  $\hat{t}_i$ .
- `smuTextUnit( $i : \text{Integer}, j : \text{Integer}$ )`: `SmuTextUnit` returns the  $j$ th semantically marked-up text unit of the  $i$ th semantically marked-up text document of this semantically marked-up text archive. Preconditions:  $i = 1, 2, \dots, |\hat{a}|$  and  $j = 1, 2, \dots, \hat{t}_i.\text{smuTextUnitLayer}().\text{size}()$ . Postcondition: `smuTextUnit( $i, j$ )` :=  $\hat{t}_i.\text{smuTextUnitLayer}().\text{smuTextUnit}(j)$ .
- `smuTextUnitLayerSize( $i : \text{Integer}$ )`: Integer returns the number of semantically marked-up text units in the semantically marked-up text unit layer of the  $i$ th semantically marked-up text document of this semantically marked-up text archive. Precondition:  $i = 1, 2, \dots, |\hat{a}|$ . Postcondition: `smuTextUnitLayerSize( $i$ )` :=  $\hat{t}_i.\text{smuTextUnitLayer}().\text{size}()$ .

## B.18 ADT for Conceptual Document Structures

**Abstract Data Type:** ConceptualDocumentStructure

**Data:** Conceptual document structure  $\hat{s} := (\check{l}, o, p)$  is a 3-tuple comprising the alphanumeric label  $\check{l} : \text{String}$  that concisely describes the common theme of text documents covered by  $\hat{s}$ , the sequence  $o := \langle o_1, o_2, \dots, o_i \rangle$  of  $i \in \mathbb{N}$  concepts such that  $o_j : \text{Concept}$ , where  $j = 1, 2, \dots, i$ , and the sequence of sets  $p := \langle P_1, P_2, \dots, P_i \rangle$ , where  $P_j : \text{SetOfNamedEntityTypeTypes}$ , that represent the types of named entities extracted from text units associated with concept  $o_j$ .

**Operations:**

- `create( $\check{l}_{init}$ : String,  $\bar{a}$ : IntTextArchive,  $p_{AS}$ : Real)` constructs a new conceptual document structure from the intermediate text archive  $\bar{a}$ . This operation enumerates all distinct semantic concepts in  $\bar{a}$  that are represented by labels of qualitatively acceptable text unit clusters, constructs the concept-specific set of extracted named entity types  $P_j$  for each semantic concept  $o_j$ , and applies the attribute support threshold  $p_{AS}$  to remove noise from the concept-specific lists of named entity types  $P_j$ . Precondition:  $\check{l}_{init}, \bar{a} \neq \text{null}$ ,  $0 \leq p_{AS} \leq 1$ . Postconditions:  $\check{l} := \check{l}_{init}$ . The sequences  $o$  and  $p$  are constructed in compliance with Definition 12 on page 61.
- `contains( $o_{input}$ : Concept)`: Boolean checks whether concept  $o_{input}$  is contained in this conceptual document structure. Precondition: None. Postcondition: If  $\exists j$  such that  $o_j = o_{input}$ , `contains( $o_{input}$ ) := true`. Otherwise, `contains( $o_{input}$ ) := false`.
- `validNamedEntities( $o_{input}$ : Concept,  $E_{input}$ : SetOfIntNamedEntities)`: SetOfNamedEntities returns the subset of  $E_{input}$  as a set of named entities, whose type is contained in the set of named entity types associated with concept  $o_{input}$ . Precondition: `contains( $o_{input}$ ) := true`. Postcondition: `validNamedEntities( $o_{input}, E_{input}$ )` removes each intermediate named entities from  $E_{input}$  whose named entity type is not contained in the set of named entity types associated with concept  $o_{input}$ , converts the resulting set of intermediate named entities into the set of corresponding named entities, and returns the latter set.
- `xmlDocuments( $\hat{a}$ : SmuTextArchive)`: String[] returns the collection of XML documents that corresponds to the contents of the semantically marked-up archive  $\hat{a}$  such that each XML document conforms to the XML document type definition returned by `xmlDtd()`. Precondition: None. Postcondition: `xmlDocuments( $\hat{a}$ ) := x`, where  $x$ : String[] denotes the collection of semantically marked-up XML documents.
- `xmlDtd()`: String returns the concept-based XML document type definition that corresponds to conceptual document structure  $\hat{s}$ . Precondition: None. Postcondition: `xmlDtd() := x`, where  $x$ : String denotes the concept-based XML document type definition.

## B.19 ADT for KDT Algorithms

**Abstract Data Type:** `KdtAlgorithm`

**Data:** KDT algorithm  $g := (\check{n}: \text{String}, \check{p}: \text{String})$  is a 2-tuple comprising the unique algorithm name  $\check{n}$  and the parameters  $\check{p}$  as strings.

**Operations:**

- `create( $\check{n}_{init}$ : String,  $\check{p}_{init}$ : String)` constructs a new KDT algorithm. Precondition: None. Postconditions:  $\check{n} := \check{n}_{init}$  and  $\check{p} := \check{p}_{init}$ .
- `name()`: String returns the name of this KDT algorithm. Precondition: None. Postcondition: `name() :=  $\check{n}$` .
- `parameters()`: String returns the parameters of this KDT algorithm. Precondition: None. Postcondition: `parameters() :=  $\check{p}$` .

## B.20 ADT for KDT Process Flows

**Abstract Data Type:** KdtProcessFlow

**Data:** KDT process flow  $f := \langle g_1, g_2, \dots, g_{|f|} \rangle$  is a sequence of KDT algorithms such that  $g_i: \text{KdtAlgorithm}$ , where  $i = 1, 2, \dots, |f|$ .

**Operations:**

- `create()` constructs a new, empty KDT process flow. Precondition: None. Postcondition:  $f := \varepsilon$ .
- `appendKdtAlgorithm(gnew: KdtAlgorithm)` appends a KDT algorithm to the end of this sequence of KDT algorithms. Precondition: None. Postcondition: Let  $i := |f|$  denote the length of this sequence prior to appending  $g_{\text{new}}$  such that  $g_{i+1} := g_{\text{new}}$ . If  $i = 0$ ,  $f := \langle g_{i+1} \rangle$ . Otherwise,  $f := \langle g_1, g_2, \dots, g_i, g_{i+1} \rangle$ .
- `kdtAlgorithm(i: Integer): KdtAlgorithm` returns the  $i$ th KDT algorithm of this KDT process flow. Precondition:  $i = 1, 2, \dots, |f|$ . Postcondition: `kdtAlgorithm(i) := gi`.
- `size(): Integer` returns the number of KDT algorithm in this KDT process flow. Precondition: None. Postcondition: `size() := |f|`.

## B.21 ADT for Tokens

**Abstract Data Type:** Token

**Data:** Token  $\check{w}: \text{String}$  is a sequence of characters that represents one natural language word, one punctuation mark, one numeral, or any other atomic and semantic-carrying character sequence.

**Operations:**

- `create( $\check{w}_{\text{init}}: \text{String}$ )` constructs a new token. Precondition: None. Postcondition:  $\check{w} := \check{w}_{\text{init}}$ .
- `length(): Integer` returns the length of this token. Precondition: None. Postcondition: `length() :=  $\check{w}$ .length()`.
- `content(): String` returns the textual content of this token. Precondition: None. Postcondition: `content() :=  $\check{w}$` .

## B.22 ADT for Tokenized Text Units

**Abstract Data Type:** TokenizedTextUnits

**Data:** Given the text unit  $\check{u}: \text{TextUnit}$ , tokenized text unit  $\check{u} := \langle \check{w}_1, \check{w}_2, \dots, \check{w}_{|\check{u}|} \rangle$  is a sequence of tokens identified in text unit  $\check{u}$  such that  $\check{w}_i: \text{Token}$ , where  $i = 1, 2, \dots, |\check{u}|$ . The set  $W_{\check{u}} := \{ \check{w} \mid \check{w}: \text{Token} := \check{u}[i], i = 1, 2, \dots, |\check{u}| \}$  contains all distinct tokens identified in  $\check{u}$ .

### Operations:

- `create()` constructs a new, empty tokenized text unit. Precondition: None. Postcondition:  $\ddot{u} := \varepsilon$ .
- `create( $\ddot{u}_{\text{init}}$ : TextUnit)` constructs a new tokenized text unit by splitting text unit  $\ddot{u}$  into a sequence of  $|\ddot{u}|$  tokens  $\langle \check{w}_1, \check{w}_2, \dots, \check{w}_{|\ddot{u}|} \rangle$ . Precondition: None. Postcondition:  $\ddot{u} := \langle \check{w}_1, \check{w}_2, \dots, \check{w}_{|\ddot{u}|} \rangle$ .
- `contains( $\check{w}$ : Token)`: Boolean checks whether the token  $\check{w}$  is contained in this tokenized text unit. Precondition:  $\check{w} \neq \text{null}$ . Postcondition: If token  $\check{w} \in W_{\ddot{u}}$ , `contains( $\check{w}$ )` := true. Otherwise, `contains( $\check{w}$ )` := false.
- `size()`: Integer returns the number of tokens in this tokenized text unit. Precondition: None. Postcondition: `size()` :=  $|\ddot{u}|$ .
- `token( $i$ : Integer)`: Token returns the  $i$ th token of this tokenized text unit. Precondition:  $i = 1, 2, \dots, |\ddot{u}|$ . Postcondition: `token( $i$ )` :=  $\check{w}_i$ .
- `tokens( $i$ : Integer,  $j$ : Integer)`: Token[]. Preconditions:  $i = 1, 2, \dots, \ddot{u}.length()$  and  $j = i, i + 1, \dots, \ddot{u}.length()$ . Postcondition: Let  $x$ : Token[] := `Token[ $j - i + 1$ ]` and  $x[1]$  :=  $\check{w}_i$ ,  $x[2]$  :=  $\check{w}_{i+1}, \dots, x[j - i + 1]$  :=  $\check{w}_j$  such that `tokens( $i, j$ )` :=  $x$ .

## B.23 ADT for Intermediate Named Entities

### Abstract Data Type: IntNamedEntity

**Data:** Given the tokenized text unit  $\ddot{u}$ : TokenizedTextUnit, intermediate named entity  $\bar{e} := (\text{p}: \text{NamedEntityType}, \check{e}: \text{String}, i: \text{Integer}, j: \text{Integer}, k: \text{Integer})$  is a 5-tuple comprising named entity type  $\text{p}$ , the canonical form  $\check{e}$  of instantiated named entity type  $\text{p}$ , and its unique identifier  $i$ , as well as the start token index  $j$  and the end token index  $k$ , where  $j = 1, 2, \dots, \ddot{u}.size()$  and  $k = j, j + 1, \dots, \ddot{u}.size()$ , such that  $\ddot{u}.tokens(j, k)$  corresponds to the tokens that instantiate named entity type  $\text{p}$  in  $\ddot{u}$ .

### Operations:

- `create( $\text{p}_{\text{init}}$ : NameEntityType,  $\check{e}_{\text{init}}$ : String,  $i_{\text{init}}$ : Integer,  $j_{\text{init}}$ : Integer,  $k_{\text{init}}$ : Integer)` constructs a new named entity. Precondition: None. Postconditions:  $\text{p} := \text{p}_{\text{init}}$ ,  $\check{e} := \check{e}_{\text{init}}$ ,  $i := i_{\text{init}}$ ,  $j := j_{\text{init}}$ , and  $k := k_{\text{init}}$ .
- `endTokenIndex()`: Integer returns the end token index of this named entity in its parent tokenized text unit. Precondition: None. Postcondition: `endTokenIndex()` :=  $k$ .
- `identifier()`: Integer returns the identifier of this named entity. Precondition: None. Postcondition: `identifier()` :=  $i$ .
- `isPlaceholder( $\check{w}$ : Token)`: Boolean determines whether token  $\check{w}$  is a named entity placeholder. Precondition: None. Postcondition: If token  $\check{w}$  is a named entity placeholder, `isPlaceholder( $\check{w}$ )` := true. Otherwise, `isPlaceholder( $\check{w}$ )` := false.
- `label()`: String returns the alphanumeric label of named entity type  $\text{p}$ . Precondition: None. Postcondition: `label()` :=  $\text{p}.label()$ .

- `placeholder()`: Token returns a token that substitutes all tokens that instantiate this intermediate named entity in its parent tokenized text unit. Precondition: None. Postcondition: `placeholder()` uniquely identifies  $\bar{e}$  within the scope of its intermediate text document.
- `startTokenIndex()`: Integer returns the start token index of this named entity in its parent tokenized text unit. Precondition: None. Postcondition: `startTokenIndex()` :=  $j$ .
- `type()`: NamedEntityType returns the type of this named entity. Precondition: None. Postcondition: `type()` :=  $p$ .
- `value()`: String returns the canonical form of this named entity. Precondition: None. Postcondition: `value()` :=  $\bar{e}$ .

## B.24 ADT for Sets of Intermediate Named Entities

**Abstract Data Type:** SetOfIntNamedEntities

**Data:** The set of intermediate named entities  $\bar{E} := \{\bar{e}_1, \bar{e}_2, \dots, \bar{e}_{|\bar{E}|}\}$  contains intermediate named entities such that  $\bar{e}_i: \text{IntNamedEntity}$ , where  $i = 1, 2, \dots, |\bar{E}|$ .

**Operations:**

- `create()` constructs a new, empty set of intermediate named entities. Precondition: None. Postcondition:  $\bar{E} = \emptyset$ .
- `add( $\bar{e}_{\text{new}}: \text{IntNamedEntity}$ )` adds an intermediate named entity to this set of intermediate named entities. Precondition: None. Postcondition:  $\bar{E} := \bar{E} \cup \bar{e}_{\text{new}}$ .
- `contains( $p: \text{NamedEntityType}$ )`: Boolean checks whether at least one intermediate named entity of type  $p$  is contained in this set of intermediate named entities. Precondition:  $p \neq \text{null}$ . Postcondition: Let the set  $\bar{E}_p = \{\bar{e} \mid \bar{e}: \text{IntNamedEntity} \in \bar{E} \text{ s.t. } \bar{e}.\text{type}() = p\}$  contain all intermediate named entities of type  $p$  in this set. If  $|\bar{E}_p| > 0$ , `contains( $p$ )` := true. Otherwise, `contains( $p$ )` := false.
- `elements()`: `IntNamedEntity[]` returns an array comprising all intermediate named entities contained in this set of intermediate named entities or null if this set is empty. Precondition: None. Postcondition: If  $\bar{E} = \emptyset$ , `elements()` := null. Otherwise, `elements()` :=  $x$ , where  $x: \text{IntNamedEntity}[] := \text{IntNamedEntity}[|\bar{E}|]$  and  $x[1] := \bar{e}_1, x[2] := \bar{e}_2, \dots, x[|\bar{E}|] := \bar{e}_{|\bar{E}|}$ .
- `size()`: Integer returns the number of intermediate named entities in this set of intermediate named entities. Precondition: None. Postcondition: `size()` :=  $|\bar{E}|$ .

## B.25 ADT for Text Unit Vectors

**Abstract Data Type:** TextUnitVector

**Data:** Text unit vector  $\mathbf{u} \in \mathbb{R}^n$ , where  $n \in \mathbb{N}$ , is an  $n$ -dimensional vector. The component  $v_i \in \mathbb{R}$ , where  $i = 1, 2, \dots, n$ , of text unit vector  $\mathbf{u} := [v_1, v_2, \dots, v_n]^T$  represents the weight of one text unit descriptor in the tokenized text unit mapped onto  $\mathbf{u}$ .

**Operations:**

- `create(uinit: Real[])` constructs a new text unit vector. Precondition: `uinit.size > 0`. Postcondition: `n := uinit.size` and `v1 := uinit[1], v2 := uinit[2], ..., vn := uinit[n]`.
- `vector(): Real[]` returns this new text unit vector. Precondition: None. Postcondition: `vector() := x`, where `x: Real[] := Real[n]` and `x[1] := v1, x[2] := v2, ..., x[n] := vn`.

## B.26 ADT for Intermediate Text Units

**Abstract Data Type:** `IntTextUnit`

**Data:** Intermediate text unit  $\bar{u} := (\check{u}: \text{TextUnit}, \ddot{u}: \text{TokenizedTextUnit}, \vec{u}: \text{TextUnitVector}, i: \text{Integer}, j: \text{Integer}, o: \text{Concept}, \bar{E}: \text{SetOfIntNamedEntities})$  is a 7-tuple comprising the original text unit  $\check{u}$ , its processed, tokenized text unit  $\ddot{u}$ , its text unit vector  $\vec{u}$ , its clustering iteration identifier  $i$ , its cluster identifier  $j$ , the domain-specific concept  $o$  that domain experts typically associate with text unit  $\check{u}$ , and the set of intermediate named entities  $\bar{E}$  identified in text unit  $\check{u}$ .

**Operations:**

- `create( $\check{u}_{init}: \text{TextUnit}, \ddot{u}_{init}: \text{TokenizedTextUnit}, \vec{u}_{init}: \text{TextUnitVector}, i_{init}: \text{Integer}, j_{init}: \text{Integer}, o_{init}: \text{Concept}, \bar{E}_{init}: \text{SetOfIntNamedEntities}$ )` constructs a new intermediate text unit. Precondition: None. Postconditions:  $\check{u} := \check{u}_{init}, \ddot{u} := \ddot{u}_{init}, \vec{u} := \vec{u}_{init}, i := i_{init}, j := j_{init}, o := o_{init},$  and  $\bar{E} := \bar{E}_{init}$ .
- `cluster(): Integer` returns the cluster identifier. Precondition: None. Postcondition: `cluster() := j`.
- `concept(): Concept` returns concept  $o$ . Precondition: None. Postcondition: `concept() := o`.
- `contains(vD: ControlledVocabularyTerm): Boolean` checks whether the token associated with  $v_D$  is contained in the processed text unit of this intermediate text unit. Precondition:  $v_D \neq \text{null}$ . Postcondition: `contains(vD) :=  $\ddot{u}$ .contains(vD.token())`.
- `contains(p: NamedEntityType): Boolean` checks whether at least one named entity of type  $p$  is contained in the set of intermediate named entities identified in this intermediate text unit. Precondition:  $p \neq \text{null}$ . Postcondition: `contains(p) :=  $\bar{E}$ .contains(p)`.
- `disambiguateWordSenses()` appends a sense tag, which indicates the concrete word sense, to all tokens that are natural language words with multiple senses in the tokenized text unit  $\ddot{u}$ . Precondition: The specific implementation of this operation fully supports the disambiguation of relevant word senses. Postcondition: All tokens that are relevant natural language words with multiple senses in the tokenized text unit  $\ddot{u}$  are appended by a sense tag, which indicates the concrete word sense.
- `identifyNamedEntities(P: SetOfNamedEntityTypes)` identifies instantiations of named entity types contained in the set  $P$  of domain-specific named entity types in the tokenized text unit  $\ddot{u}$ . For each discovered named entity, a corresponding intermediate named entity is created and stored in  $\bar{E}$ . Precondition: The specific implementation of this operation fully supports the identification of all named entity types in  $P$ . Postcondition:  $\bar{E}$  is first

emptied and finally contains all identified intermediate named entities. The tokenized text unit  $\bar{u}$  is not modified.

- `intNamedEntities()`: `SetOfIntNamedEntities` returns the set of intermediate named entities. Precondition: None. Postcondition: `intNamedEntities() :=  $\bar{E}$` .
- `iteration()`: `Integer` returns the clustering iteration identifier. Precondition: None. Postcondition: `iteration() :=  $i$` .
- `lemmatizeWords()` replaces all tokens that are natural language words in the tokenized text unit  $\bar{u}$  with their grammatical root forms. Precondition: The specific implementation of this operation fully supports the lemmatization of all relevant tokens. Postcondition: All tokens that are natural language words in the tokenized text unit  $\bar{u}$  are replaced with their grammatical root forms.
- `originalTextUnit()`: `TextUnit` returns the original text unit. Precondition: None. Postcondition: `originalTextUnit() :=  $\bar{u}$` .
- `processedTextUnit()`: `TokenizedTextUnit` returns the processed, tokenized text unit. Precondition: None. Postcondition: `processedTextUnit() :=  $\bar{u}$` .
- `replaceNamedEntities()` replaces tokens that instantiate an identified intermediate named entity in the tokenized text unit  $\bar{u}$  with the corresponding named entity placeholder. Precondition: The tokenized text unit does not contain named entity placeholders. Postcondition: Tokens that instantiate the identified intermediate named entity  $\bar{e}_i$ : `IntNamedEntity =  $\bar{E}$ .elements()[ $i$ ]`, where  $i = 1, 2, \dots, \bar{E}.elements().size()$ , in the tokenized text unit  $\bar{u}$  are replaced with named entity placeholder  `$\bar{e}_i$ .placeholder()`.
- `textUnitVector()`: `TextUnitVector` returns the text unit vector. Precondition: None. Postcondition: `textUnitVector() :=  $\bar{u}$` .

## B.27 ADT for Intermediate Text Unit Layers

**Abstract Data Type:** `IntTextUnitLayer`

**Data:** Given text unit layer  $\check{r}$ : `TextUnitLayer`, an intermediate text unit layer  $\bar{r}$  :=  $\langle \bar{u}_1, \bar{u}_2, \dots, \bar{u}_{|\bar{r}|} \rangle$  is a sequence of intermediate text units such that  $\bar{u}_i$ : `IntTextUnit` and  $\bar{u}_i.textUnit() = \check{r}.textUnit(i)$ , where  $i = 1, 2, \dots, \check{r}.size()$ .

**Operations:**

- `create()` constructs a new, empty intermediate text unit layer. Precondition: None. Postcondition:  `$\bar{r} := \varepsilon$` .
- `appendIntTextUnit( $\bar{u}_{new}$ : IntTextUnit)` appends an intermediate text unit to the end of this sequence of intermediate text units. Precondition: None. Postcondition: Let  $i := |\bar{r}|$  denote the length of this sequence prior to appending  $\bar{u}_{new}$  such that  $\bar{u}_{i+1} := \bar{u}_{new}$ . If  $i = 0$ ,  `$\bar{r} := \langle \bar{u}_{i+1} \rangle$` . Otherwise,  `$\bar{r} := \langle \bar{u}_1, \bar{u}_2, \dots, \bar{u}_i, \bar{u}_{i+1} \rangle$` .
- `intTextUnit( $i$ : Integer)`: `IntTextUnit` returns the  $i$ th intermediate text unit of this intermediate text unit layer. Precondition:  $i = 1, 2, \dots, |\bar{r}|$ . Postcondition: `intTextUnit( $i$ ) :=  $\bar{u}_i$` .
- `size()`: `Integer` returns the number of intermediate text units in this intermediate text unit layer. Precondition: None. Postcondition: `size() :=  $|\bar{r}|$` .

## B.28 ADT for Intermediate Text Documents

**Abstract Data Type:** IntTextDocument

**Data:** Given the text document  $\check{t}$ : TextDocument, an intermediate text document  $\bar{t} := (\check{t}: \text{TextDocument}, \bar{r}: \text{IntTextUnitLayer})$  is a 2-tuple comprising the text document  $\check{t}$  and the intermediate text unit layer  $\bar{r}$  that represents a decomposition of text document  $\check{t}$  into intermediate text units.

**Operations:**

- `create( $\check{t}_{\text{init}}: \text{TextDocument}, \bar{r}_{\text{init}}: \text{IntTextUnitLayer}$ )` constructs a new intermediate text document. Precondition: None. Postconditions:  $\check{t} := \check{t}_{\text{init}}$  and  $\bar{r} := \bar{r}_{\text{init}}$ .
- `intTextUnitLayer(): IntTextUnitLayer` returns the intermediate text unit layer. Precondition: None. Postcondition: `intTextUnitLayer() :=  $\bar{r}$` .
- `textDocument(): TextDocument` returns the semantically marked-up text document. Precondition: None. Postcondition: `textDocument() :=  $\check{t}$` .

## B.29 ADT for Intermediate Text Archives

**Abstract Data Type:** IntTextArchive

**Data:** Given text archive  $\check{a}$ : TextArchive, an intermediate text archive  $\bar{a} := \langle \bar{t}_1, \bar{t}_2, \dots, \bar{t}_{|\bar{a}|} \rangle$  is a sequence of not necessarily distinct intermediate text documents such that  $\bar{t}_i: \text{IntTextDocument}$  and  $\bar{t}_i.\text{textDocument}() = \check{a}.\text{textDocument}(i)$ , where  $i = 1, 2, \dots, \check{a}.\text{size}()$ .

**Operations:**

- `create()` constructs a new, empty intermediate text archive. Precondition: None. Postcondition:  $\bar{a} := \varepsilon$ .
- `appendIntTextDocument( $\bar{t}_{\text{new}}: \text{IntTextDocument}$ )` appends an intermediate text document to the end of this sequence of intermediate text documents. Precondition: None. Postcondition: Let  $i := |\bar{a}|$  denote the length of this sequence prior to appending  $\bar{t}_{\text{new}}$  such that  $\bar{t}_{i+1} := \bar{t}_{\text{new}}$ . If  $i = 0$ ,  $\bar{a} := \langle \bar{t}_{i+1} \rangle$ . Otherwise,  $\bar{a} := \langle \bar{t}_1, \bar{t}_2, \dots, \bar{t}_i, \bar{t}_{i+1} \rangle$ .
- `attributeSupport( $o: \text{Concept}, p: \text{NamedEntityType}$ ): Real` returns the attribute support (cf. Definition 40 on page 139) of  $p$  within intermediate text units of this intermediate text archive assigned to  $o$ . Precondition:  $o, p \neq \text{null}$ . Postcondition: Let the multi-set  $\bar{A}_o = \{ \{ \bar{u} \mid \bar{u}: \text{IntTextUnit} := \text{intTextUnit}(j, k_j) \text{ s.t. } \bar{u}.\text{concept}() = o \forall j = 1, 2, \dots, \text{size}() \forall k_j = 1, 2, \dots, \text{intTextUnitLayerSize}(j) \} \}$  contain all intermediate text units herein assigned to  $o$ . If  $|\bar{A}_o| > 0$ , `attributeSupport( $o, p$ ) :=  $|\{ \{ \bar{u} \mid \bar{u}: \text{IntTextUnit} \in \bar{A}_o \text{ s.t. } \bar{u}.\text{contains}(p) \} \}| / |\bar{A}_o|$` . Otherwise, `attributeSupport( $o, p$ ) := 0`.
- `intTextDocument( $i: \text{Integer}$ ): IntTextDocument` returns the  $i$ th intermediate text document of this intermediate text archive. Precondition:  $i = 1, 2, \dots, |\bar{a}|$ . Postcondition: `intTextDocument( $i$ ) :=  $\bar{t}_i$` .

- `intTextUnit(i: Integer, j: Integer)`: `IntTextUnit` returns the *j*th intermediate text unit of the *i*th intermediate text document of this intermediate text archive. Preconditions:  $i = 1, 2, \dots, |\bar{a}|$  and  $j = 1, 2, \dots, \bar{t}_i.\text{intTextUnitLayer}().\text{size}()$ . Postcondition: `intTextUnit(i, j) :=  $\bar{t}_i.\text{intTextUnitLayer}().\text{intTextUnit}(j)$ .`
- `intTextUnitLayerSize(i: Integer)`: `Integer` returns the number of intermediate text units in the intermediate text unit layer of the *i*th intermediate text document of this intermediate text archive. Precondition:  $i = 1, 2, \dots, |\bar{a}|$ . Postcondition: `intTextUnitLayerSize(i) :=  $\bar{t}_i.\text{intTextUnitLayer}().\text{size}()$ .`
- `size()`: `Integer` returns the number of intermediate text documents in this intermediate text archive. Precondition: `None`. Postcondition: `size() :=  $|\bar{a}|$ .`

## B.30 ADT for Controlled Vocabulary Terms

**Abstract Data Type:** `ControlledVocabularyTerm`

**Data:** The 4-tuple  $v := (\check{w} : \text{Token}, i : \text{Integer}, d : \text{Boolean}, j : \text{Integer})$  is a controlled vocabulary term comprising the token  $\check{w}$ , a unique term identifier *i*, the boolean variable *d* whose value indicates the term type, and the term identifier *j* of an optionally associated controlled vocabulary term. If  $d = \text{true}$ , controlled vocabulary term *v* is a text unit descriptor and  $j = \text{null}$ . Otherwise, controlled vocabulary term *v* is a text unit non-descriptor and  $j \neq \text{null}$ .

**Operations:**

- `create( $\check{w}_{\text{init}} : \text{Token}, i_{\text{init}} : \text{Integer}, d_{\text{init}} : \text{Boolean}, j_{\text{init}} : \text{Integer}$ )` constructs a new controlled vocabulary term. Precondition: `None`. Postconditions:  $\check{w} := \check{w}_{\text{init}}, i := i_{\text{init}}, d := d_{\text{init}}, j := j_{\text{init}}$ .
- `associatedTermId()`: `Integer` returns the identifier of the associated controlled vocabulary term. Precondition: `None`. Postcondition: `associatedTermId() := j.`
- `id()`: `Integer` returns the identifier. Precondition: `None`. Postcondition: `id() := i.`
- `isDescriptor()`: `Boolean` returns the type indicator. Precondition: `None`. Postcondition: `isDescriptor() := d.`
- `token()`: `Token` returns the token. Precondition: `None`. Postcondition: `token() :=  $\check{w}$ .`

## B.31 ADT for Controlled Vocabularies

**Abstract Data Type:** `ControlledVocabulary`

**Data:** The non-empty set  $V := \{v_1, v_2, \dots, v_{|V|}\}$  is a controlled vocabulary comprising the controlled vocabulary terms  $v_i : \text{ControlledVocabularyTerm}$ , where  $i = 1, 2, \dots, |V|$ , such that any token is represented by at most one controlled vocabulary term, and there exists one directly or indirectly associated text unit descriptor for each text unit non-descriptor. All text unit descriptors in controlled vocabulary *V*

are elements of the non-empty set  $V_D = \{v \mid v: \text{ControlledVocabularyTerm} \in V \text{ s.t. } v.\text{isDescriptor}() = \text{true}\} \subseteq V$  and satisfy the constraint  $v_j.\text{id}() = j$ , where  $v_j: \text{ControlledVocabularyTerm} \in V_D$  and  $j = 1, 2, \dots, |V_D|$ .

**Operations:**

- `create()` constructs a new, empty controlled vocabulary. Precondition: None. Postcondition:  $V := \emptyset$ .
- `add(vnew: Token)` adds a new controlled vocabulary term and ensures the constraints on unique term identifiers in the controlled vocabulary. Precondition: None. Postcondition:  $V := V \cup \{v_{\text{new}}\}$ .
- `descriptor(i: Integer): ControlledVocabularyTerm` returns the text unit descriptor  $v_i \in V_D$ . Precondition:  $i = 1, 2, \dots, |V_D|$ . Postcondition: `descriptor(i) := vi`.
- `numberOfDescriptors(): Integer` returns the number of text unit descriptors. Precondition: None. Postcondition: `numberOfDescriptors() := |VD| >= 0`.

## B.32 ADT for Text Unit Clusters

**Abstract Data Type:** TextUnitCluster

**Data:** Given intermediate text archive  $\bar{a}: \text{IntTextArchive}$ , the multi-set  $C_{t,i} = \{\{\bar{u} \mid \bar{u}: \text{IntTextUnit} := \bar{a}.\text{intTextUnit}(j, k_j) \ \forall j = 1, 2, \dots, \bar{a}.\text{size}() \ \forall k_j = 1, 2, \dots, \bar{a}.\text{intTextUnitLayerSize}(j) \text{ s.t. } \bar{u}.\text{iteration}() = t \text{ and } \bar{u}.\text{cluster}() = i\}\}$  is a text unit cluster comprising all intermediate text units in  $\bar{a}$ , whose text unit vectors are assigned to cluster  $i \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ . If the domain expert accepts (rejects) the automatically generated cluster quality assessment,  $q_{DE}: \text{Boolean}$  assumes the true (false) value. Otherwise,  $q_{DE}$  takes the null value. If this cluster is qualitatively acceptable and semantically labeled,  $o: \text{Concept}$  represents the concept associated with this cluster. Otherwise,  $o$  assumes the null value.

**Operations:**

- `create( $\bar{a}_{\text{init}}: \text{IntTextArchive}, t_{\text{init}}: \text{Integer}, i_{\text{init}}: \text{Integer}$ )` constructs a new text unit cluster. Precondition: None. Postcondition:  $C_{t,i} = \{\{\bar{u} \mid \bar{u}: \text{IntTextUnit} := \bar{a}_{\text{init}}.\text{intTextUnit}(j, k_j) \ \forall j = 1, 2, \dots, \bar{a}_{\text{init}}.\text{size}() \ \forall k_j = 1, 2, \dots, \bar{a}_{\text{init}}.\text{intTextUnitLayerSize}(j) \text{ s.t. } \bar{u}.\text{iteration}() = t_{\text{init}} \text{ and } \bar{u}.\text{cluster}() = i_{\text{init}}\}\}$ ,  $t := t_{\text{init}}$ , and  $i := i_{\text{init}}$ , as well as  $q_{DE} := \text{null}$ , and  $o := \text{null}$ .
- `approveClusterQualityAssessment()` sets the decision of a domain expert to approve the automatically generated cluster quality assessment. Precondition: None. Postcondition:  $q_{DE} := \text{true}$ .
- `concept(): Concept` returns the associated concept. Precondition: None. Postcondition: `concept() := o`.
- `clusterQualityAssessmentIsApproved(): Boolean` checks whether a domain expert has accepted the automatically generated cluster quality assessment. Precondition: None. Postcondition: If  $q_{DE} = \text{true}$ , `clusterQualityAssessmentIsApproved() := true`. Otherwise, `clusterQualityAssessmentIsApproved() := false`.

- `clusterQualityAssessmentIsRejected()`: Boolean checks whether a domain expert has rejected the automatically generated cluster quality assessment. Precondition: None. Postcondition: If  $q_{DE} = \text{false}$ , `clusterQualityAssessmentIsRejected()` := true. Otherwise, `clusterQualityAssessmentIsRejected()` := false.
- `descriptorCoverage(V: ControlledVocabulary,  $p_{RD}$ : Real)`: Real returns the ratio of the number of distinct, non-rare descriptors occurring in intermediate text units of this text unit cluster to the total number of text unit descriptors in controlled vocabulary  $V$ . Preconditions:  $V \neq \text{null}$ ,  $V.\text{numberOfDescriptors}() > 0$ , and  $p_{RD} \in [0, 1]$ . Postcondition: `descriptorCoverage(V,  $p_{RD}$ )` :=  $|\{v_D \mid v_D: \text{ControlledVocabularyTerm} \in V_D \text{ s.t. descriptorSupport}(v_D) > p_{RD}\}| / |V_D|$ , where  $V_D := \{v_D \mid v_D: \text{ControlledVocabularyTerm} := V.\text{descriptor}(i) \ \forall i = 1, 2, \dots, V.\text{numberOfDescriptors}()\}$ .
- `descriptorDominance(V: ControlledVocabulary,  $p_{RD}$ : Real,  $p_{DD}$ : Real)`: Real returns the ratio of the number of distinct, dominant descriptors occurring in intermediate text units of this text unit cluster to the number of distinct, non-rare descriptors occurring in intermediate text units of this cluster. Preconditions:  $V \neq \text{null}$ ,  $V.\text{numberOfDescriptors}() > 0$ , and  $p_{RD}, p_{DD} \in [0, 1]$ . Postcondition: If `descriptorCoverage(V,  $p_{RD}$ ) > 0`, `descriptorDominance(V,  $p_{RD}$ ,  $p_{DD}$ )` :=  $|\{v_D \mid v_D: \text{ControlledVocabularyTerm} \in V_D \text{ s.t. descriptorSupport}(v_D) > p_{DD}\}| / |\{v_D \mid v_D: \text{ControlledVocabularyTerm} \in V_D \text{ s.t. descriptorSupport}(v_D) > p_{RD}\}|$ , where  $V_D := \{v_D \mid v_D: \text{ControlledVocabularyTerm} := V.\text{descriptor}(i) \ \forall i = 1, 2, \dots, V.\text{numberOfDescriptors}()\}$ . Otherwise, `descriptorDominance(V,  $p_{RD}$ ,  $p_{DD}$ )` := 0.
- `descriptorSupport( $v_D$ : ControlledVocabularyTerm)`: Real returns the support of  $v_D$  in this text unit cluster that is defined as the ratio of the number of intermediate text units in  $C_{t,i}$  whose processed text unit contains the token  $v_D.\text{token}()$  to the total number of intermediate text units in  $C_{t,i}$ . Precondition:  $v_D \neq \text{null}$ . Postcondition: If `size() > 0`, `descriptorSupport( $v_D$ )` :=  $|\{\bar{u} \mid \bar{u}: \text{IntTextUnit} \in C_{t,i} \text{ s.t. } \bar{u}.\text{contains}(v_D)\}| / |C_{t,i}|$ . Otherwise, `descriptorSupport( $v_D$ )` := 0.
- `isDominantDescriptor( $v_D$ : ControlledVocabularyTerm,  $p_{DD}$ : Real)`: Boolean tests whether text unit descriptor  $v_D$  is dominant in this text unit cluster with respect to threshold  $p_{DD}$ . Preconditions:  $v_D \neq \text{null}$  and  $p_{DD} \in [0, 1]$ . Postcondition: If `descriptorSupport( $v_D$ )  $\geq p_{DD}$` , `isDominantDescriptor( $v_D$ ,  $p_{DD}$ )` := true. Otherwise, `isDominantDescriptor( $v_D$ ,  $p_{DD}$ )` := false.
- `isQualitativelyAcceptable(V: ControlledVocabulary,  $p_{RD}$ : Real,  $p_{DD}$ : Real,  $p_{\text{maxDC}}$ : Real,  $p_{\text{minDD}}$ : Real,  $p_{\text{minCS}}$ : Integer)`: Boolean checks whether this text unit cluster is qualitatively acceptable. Preconditions:  $V \neq \text{null}$ ,  $V.\text{numberOfDescriptors}() > 0$ ,  $p_{RD}, p_{DD}, p_{\text{maxDC}}, p_{\text{minDD}} \in [0, 1]$ , and  $p_{\text{minCS}} > 0$ . Postcondition: If  $C_{t,i}.\text{descriptorCoverage}(V, p_{RD}) \leq p_{\text{maxDC}}$ ,  $C_{t,i}.\text{descriptorDominance}(V, p_{RD}, p_{DD}) \geq p_{\text{minDD}}$ , and  $C_{t,i}.\text{size}() \geq p_{\text{minCS}}$ , `isQualitativelyAcceptable(V,  $p_{RD}$ ,  $p_{DD}$ ,  $p_{\text{maxDC}}$ ,  $p_{\text{minDD}}$ ,  $p_{\text{minCS}}$ )` := true. Otherwise, `isQualitativelyAcceptable(V,  $p_{RD}$ ,  $p_{DD}$ ,  $p_{\text{maxDC}}$ ,  $p_{\text{minDD}}$ ,  $p_{\text{minCS}}$ )` := false.
- `isRareDescriptor( $v_D$ : ControlledVocabularyTerm,  $p_{RD}$ : Real)`: Boolean tests whether text unit descriptor  $v_D$  is rare in this text unit cluster with respect to threshold  $p_{RD}$ . Preconditions:  $v_D \neq \text{null}$  and  $p_{RD} \in [0, 1]$ . Postcondition: If `descriptorSupport( $v_D$ )  $\leq p_{RD}$` , `isRareDescriptor( $v_D$ ,  $p_{RD}$ )` := true. Otherwise, `isRareDescriptor( $v_D$ ,  $p_{RD}$ )` := false.
- `qualityIndex(V: ControlledVocabulary,  $p_{RD}$ : Real,  $p_{DD}$ : Real,  $C_t$ : TextUnitClustering)`: Real returns the quality index of this text unit cluster. Preconditions:  $V \neq \text{null}$ ,  $V.\text{numberOfDescriptors}() > 0$ , and  $p_{RD}, p_{DD} \in [0, 1]$ . Postcondition: `qualityIndex(V,  $p_{RD}$ ,  $p_{DD}$ ,`

- $C_t := 1/3 \cdot (1 - \text{descriptorCoverage}(V, p_{RD})) + 1/3 \cdot \text{descriptorDominance}(V, p_{RD}, p_{DD}) + 1/3 \cdot \text{size}() / C_t.\text{numberOfIntTextUnits}() \in [0; 1]$  if  $\text{descriptorCoverage}(V, p_{RD}) > 0$ . Otherwise,  $\text{qualityIndex}(V, p_{RD}, p_{DD}, C_t) := 1/3 \cdot \text{size}() / C_t.\text{numberOfIntTextUnits}() \in [0; 1/3]$ .
- `rejectClusterQualityAssessment()` sets the decision of a domain expert to reject the automatically generated cluster quality assessment. Precondition: None. Postcondition:  $q_{DE} := \text{false}$ .
  - `setConcept(oinit: Concept)` sets the associated concept. Precondition: None. Postcondition:  $o := o_{\text{init}}$ . The concept of all intermediate text units assigned to this text unit cluster is set accordingly.
  - `size(): Integer` returns the number of intermediate text units in this text unit cluster. Precondition: None. Postcondition:  $\text{size}() := |C_{t,i}|$ .

## B.33 ADT for Text Unit Clusterings

**Abstract Data Type:** TextUnitClustering

**Data:** Given intermediate text archive  $\bar{a}$ : IntTextArchive and maximum cluster identifier  $i_{max} \in \mathbb{N}$  in clustering iteration  $t \in \mathbb{N}$ , the set  $C_t := \{C_{t,1}, C_{t,2}, \dots, C_{t,i_{max}}\}$  is a text unit clustering comprising all  $i_{max}$  text unit clusters  $C_{t,i}$ : TextUnitCluster, where  $i = 1, 2, \dots, i_{max}$ , discovered by a clustering algorithm in clustering iteration  $t$ .

**Operations:**

- `create( $\bar{a}_{\text{init}}$ : IntTextArchive,  $t_{\text{init}}$ : Integer,  $i_{max,\text{init}}$ : Integer)` constructs a new text unit clustering. Precondition: None. Postcondition:  $C_t := \{\text{TextUnitCluster.create}(\bar{a}_{\text{init}}, t_{\text{init}}, 1), \text{TextUnitCluster.create}(\bar{a}_{\text{init}}, t_{\text{init}}, 2), \dots, \text{TextUnitCluster.create}(\bar{a}_{\text{init}}, t_{\text{init}}, i_{max,\text{init}})\}$ ,  $t := t_{\text{init}}$ , and  $i_{max} := i_{max,\text{init}}$ .
- `automatedLabelingOfClusters(m: IterationMetadata)` assigns a default cluster label to each qualitatively acceptable cluster in this text unit clustering. Precondition: The operation `automatedRankingOfClusters` must have been completed. Postcondition: For each acceptable cluster in this text unit clustering, a default concept is instantiated that comprises the default cluster name. The operation `setConcept()` of ADT TextUnitCluster is employed to set the concept of each intermediate text unit assigned to the corresponding text unit cluster accordingly.
- `automatedRankingOfClusters(m: IterationMetadata)` partitions all text unit clusters in this text unit clustering into qualitatively acceptable and unacceptable ones. Within each category, text unit clusters are ranked by decreasing quality index. Precondition: None. Postcondition: The ranking of clusters is visualized appropriately for inspection by the domain expert.
- `conceptAssignmentInApplicationMode( $\bar{a}_A$ : IntTextArchive): IntTextArchive` sets the concept of all intermediate text units in  $\bar{a}_A$  that have been assigned to a qualitatively acceptable text unit cluster in classification iteration  $t$  of the knowledge application phase of the DIASDEM framework. Precondition: None. Postcondition: For all intermediate text units in  $\bar{a}_A$  assigned to a qualitatively acceptable text unit cluster  $C_{t,j}$ , where

- $j = 1, 2, \dots, i$  and  $C_{t,j}.concept() \neq \text{null}$ , in classification iteration  $t$ , the concept is set to  $C_{t,j}.concept()$ . The updated intermediate text archive  $\bar{a}_A$  is returned.
- `interactiveReviewOfClusterLabels(m: IterationMetadata)` executes an interactive procedure that enables the domain expert to approve or correct automatically generated cluster labels. Precondition: The operation `automatedLabelingOfClusters` must have been completed. Postcondition: The operation `setConcept()` of ADT `TextUnitCluster` is employed to update the concept that comprises the cluster name of text unit clusters as proposed by the domain expert.
  - `interactiveScreeningOfClusterRanking(m: IterationMetadata)` executes an interactive procedure that enables the domain expert to approve or reject automatically generated clusters quality assessments. Precondition: The operation `automatedRankingOfClusters` must have been completed. Postcondition: The operations `approveClusterQualityAssessment()` and `rejectClusterQualityAssessment()` of ADT `TextUnitCluster` are employed to update text unit clusters with decisions made by the domain expert.
  - `numberOfIntTextUnits(): Integer` returns the number of intermediate text units in this text unit clustering. Precondition: None. Postcondition:  $\text{size()} := C_{t,1}.\text{size()} + C_{t,2}.\text{size()} + \dots + C_{t,i_{max}}.\text{size}()$ .
  - `qualityIndex(V: ControlledVocabulary, pRD: Real, pDD: Real): Real` returns the quality index of this text unit clustering. Preconditions:  $V \neq \text{null}$ ,  $V.\text{numberOfDescriptors()} > 0$ , and  $p_{RD}, p_{DD} \in [0, 1]$ . Postcondition:  $\text{qualityIndex}(V, p_{RD}, p_{DD}, C_t) := \text{numberOfIntTextUnits()}^{-1} \cdot \sum_{j=1}^{i_{max}} C_{t,j}.\text{size()} \cdot C_{t,j}.\text{qualityIndex}(V, p_{RD}, p_{DD}, C_t) \in [0; 1]$ .

## B.34 ADT for Descriptor Weighting Schemata

**Abstract Data Type:** `DescriptorWeightingScheme`

**Data:** The abstract data type encapsulates a parameterized descriptor weighting scheme (cf. Subsection 4.2.5) and, if already executed in discovery mode, invariant scheme-specific information, like the collection frequency components of descriptor weights. This descriptor weighting scheme is uniquely identified by the number of its associated clustering iteration  $t$ : `Integer` in a certain DIASDEM knowledge discovery process, its name `Name: String`, its iteration-specific parameters `Parameters: String`, and the iteration-specific controlled vocabulary `V: ControlledVocabulary`.

**Operations:**

- `create(tinit: Integer, Nameinit: String, Parametersinit: String, Vinit: ControlledVocabulary)` constructs a new instance of the specified descriptor weighting scheme. Precondition:  $t_{init} > 0$  and  $\text{Name}_{init} \neq \text{null}$ . Postcondition:  $t := t_{init}$ ,  $\text{Name} := \text{Name}_{init}$ , and  $\text{Parameters} := \text{Parameters}_{init}$ , and  $V := V_{init}$ .
- `createVectorsInApplicationMode( $\bar{a}_A$ : IntTextArchive): IntTextArchive` creates text unit vectors for intermediate text units in  $\bar{a}_A$  that are not yet assigned to a semantic concept by utilizing existing invariant weighting components. Precondition: The operation

createVectorsInDiscoveryMode must have been completed. Postcondition: For all intermediate text units in createVectorsInApplicationMode( $\bar{a}_A$ ) that are not yet assigned to a semantic concept, a text unit vector is created.

- createVectorsInDiscoveryMode( $\bar{a}_T$ : IntTextArchive): IntTextArchive creates text unit vectors for intermediate text units in  $\bar{a}_T$  that are not yet assigned to a semantic concept. Invariant weighting components are computed and persistently stored for usage in the knowledge application phase. Precondition: None. Postcondition: For all intermediate text units in createVectorsInDiscoveryMode( $\bar{a}_T$ ) that are not yet assigned to a semantic concept, a text unit vector is created.

## B.35 ADT for Clustering Algorithms

**Abstract Data Type:** ClusteringAlgorithm

**Data:** The abstract data type encapsulates a parameterized clustering algorithm that satisfies the DIASDEM-specific requirements (cf. Subsection 4.3.2) and, if already executed in discovery mode, the labels of discovered text unit clusters, and the maximum cluster identifier  $i_{max} \in \mathbb{N}$ . This clustering algorithm is uniquely identified by the number of its associated clustering iteration  $t$ : Integer in a certain DIASDEM knowledge discovery process, its name Name: String, and its iteration-specific parameters Parameters: String.

**Operations:**

- create( $t_{init}$ : Integer, Name<sub>init</sub>: String, Parameters<sub>init</sub>: String) constructs a new instance of the specified clustering algorithm. Precondition:  $t_{init} > 0$  and Name<sub>init</sub>  $\neq$  null. Postcondition:  $t := t_{init}$ , Name := Name<sub>init</sub>, and Parameters := Parameters<sub>init</sub>.
- executeInApplicationMode( $\bar{a}_A$ : IntTextArchive): IntTextArchive executes the parameterized clustering algorithm in the knowledge application phase of the DIASDEM framework. The input data set to this clustering algorithm comprises only text unit vectors of intermediate text units in  $\bar{a}_A$  that are not yet assigned to a semantic concept. Each text unit vector is assigned to one text unit cluster. Precondition: The operation executeInDiscoveryMode must have been completed. Postcondition: For all intermediate text units in  $\bar{a}_A$  that are not yet assigned to a semantic concept, the clustering iteration identifier is set to  $t$  and the cluster identifier is set according to the new cluster assignment.
- executeInDiscoveryMode( $\bar{a}_T$ : IntTextArchive): IntTextArchive executes the parameterized clustering algorithm in the knowledge discovery phase of the DIASDEM framework. The input data set to this clustering algorithm comprises only text unit vectors of intermediate text units in  $\bar{a}_T$  that are not yet assigned to a semantic concept. The maximum cluster identifier  $i_{max}$  is set. Descriptions of all discovered text unit clusters are persistently stored for usage in the knowledge application phase. Precondition: None. Postcondition: For all intermediate text units in  $\bar{a}_T$  that are not yet assigned to a semantic concept, the clustering iteration identifier is set to  $t$  and the cluster identifier is set according to the new cluster assignment.

- `maximumClusterIdentifier()`: Integer returns the maximum cluster identifier. Precondition: The operation `executeInDiscoveryMode` must have been completed. Postcondition: `maximumClusterIdentifier() := imax`.

## B.36 ADT for Cluster Quality Criteria

**Abstract Data Type:** `ClusterQualityCriteria`

**Data:** The abstract data type encapsulates the dominant descriptor threshold  $p_{DD} \in [0; 1]$ , the rare descriptor threshold  $p_{RD} \in [0; 1]$ , the maximum descriptor coverage  $p_{maxDC} \in [0; 1]$ , the minimum descriptor dominance  $p_{minDD} \in [0; 1]$ , and the minimum cluster size  $p_{minCS} \in \mathbb{N}$ .

**Operations:**

- `create(pDD,init: Real, pRD,init: Real, pmaxDC,init: Real, pminDD,init: Real, pminCS,init: Integer)` constructs new cluster quality criteria. Precondition:  $p_{DD,init} \in [0; 1]$ ,  $p_{RD,init} \in [0; 1]$ ,  $p_{RD,init} \ll p_{DD,init}$ ,  $p_{maxDC,init} \in [0; 1]$ ,  $p_{minDD,init} \in [0; 1]$ , and  $p_{minCS,init} > 0$ . Postcondition:  $p_{DD} := p_{DD,init}$ ,  $p_{RD} := p_{RD,init}$ ,  $p_{maxDC} := p_{maxDC,init}$ ,  $p_{minDD} := p_{minDD,init}$ , and  $p_{minCS} := p_{minCS,init}$ .
- `dominantDescriptorThreshold()`: Real returns the dominant descriptor threshold. Precondition: None. Postcondition: `dominantDescriptorThreshold() := pDD`.
- `maximumDescriptorCoverage()`: Real returns the maximum descriptor coverage. Precondition: None. Postcondition: `maximumDescriptorCoverage() := pmaxDC`.
- `minimumClusterSize()`: Integer returns the minimum cluster size. Precondition: None. Postcondition: `minimumClusterSize() := pminCS`.
- `minimumDescriptorDominance()`: Real returns the minimum descriptor dominance. Precondition: None. Postcondition: `minimumDescriptorDominance() := pminDD`.
- `rareDescriptorThreshold()`: Real returns the rare descriptor threshold. Precondition: None. Postcondition: `rareDescriptorThreshold() := pRD`.

## B.37 ADT for Iteration Metadata

**Abstract Data Type:** `IterationMetadata`

**Data:** The 5-tuple  $m := (t: \text{Integer}, w: \text{DescriptorWeightingScheme}, g: \text{ClusteringAlgorithm}, q: \text{ClusterQualityCriteria}, C_t: \text{TextUnitClustering})$  represents metadata specified by KDT and domain experts, created in a clustering iteration of the knowledge discovery phase, and used in a classification iteration of the knowledge application phase. An iteration metadata item comprises the iteration number  $t$ , as well as the iteration-specific descriptor weighting scheme  $w$ , the iteration-specific clustering algorithm  $g$ , the iteration-specific DIASDEM cluster quality criteria  $q$ , and the iteration-specific text unit clustering  $C_t$  found in the knowledge discovery phase.

**Operations:**

- `create( $t_{\text{init}}$  : Integer,  $w_{\text{init}}$  : DescriptorWeightingScheme,  $g_{\text{init}}$  : ClusteringAlgorithm,  $q_{\text{init}}$  : ClusterQualityCriteria)` constructs new cluster quality criteria. Precondition:  $t_{\text{init}} > 0$ . Postcondition:  $t := t_{\text{init}}$ ,  $w := w_{\text{init}}$ ,  $g := g_{\text{init}}$ , and  $q := q_{\text{init}}$ , as well as  $C_t := \text{null}$ .
- `clusteringAlgorithm()`: ClusteringAlgorithm returns the iteration-specific clustering algorithm. Precondition: None. Postcondition: `clusteringAlgorithm()` :=  $g$ .
- `clusterQualityCriteria()`: ClusterQualityCriteria returns the iteration-specific DIASDEM cluster quality criteria. Precondition: None. Postcondition: `clusterQualityCriteria()` :=  $q$ .
- `descriptorWeightingScheme()`: DescriptorWeightingScheme returns the iteration-specific descriptor weighting scheme. Precondition: None. Postcondition: `descriptorWeightingScheme()` :=  $w$ .
- `iterationNumber()`: Integer returns the iteration number. Precondition: None. Postcondition: `iterationNumber()` :=  $t$ .
- `setTextUnitClustering( $C_{t,\text{new}}$  : TextUnitClustering)` sets the iteration-specific text unit clustering. Precondition: None. Postcondition:  $C_t := C_{t,\text{new}}$ .
- `textUnitClustering()`: TextUnitClustering returns the iteration-specific text unit clustering. Precondition: None. Postcondition: `textUnitClustering()` :=  $C_t$ .

## B.38 ADT for KDT Process Metadata

**Abstract Data Type:** KdtProcessMetadata

**Data:** Let  $n := \langle m_1, m_2, \dots, m_{|n|} \rangle$  denote a sequence of iteration metadata items instantiated during the iterative clustering step of one specific knowledge discovery process. The iteration metadata item  $m_t$ : IterationMetadata, where  $t = 1, 2, \dots, |n|$ , corresponds to the  $t$ th clustering iteration of the respective KDT process. Furthermore,  $P$ : SetOfNamedEntityTypeTypes denotes the set of named entity types, whose instances are extracted from text units in the focal KDT process. The 2-tuple  $k := (n, P)$  represents the KDT process metadata item created and persistently stored during knowledge discovery for subsequent usage during knowledge application.

**Operations:**

- `create( $P_{\text{init}}$  : SetOfNamedEntityTypeTypes)` constructs a new KDT process metadata item. Precondition: None. Postcondition:  $n := \varepsilon$  and  $P := P_{\text{init}}$ .
- `appendIterationMetadata( $m_{\text{new}}$  : IterationMetadata)` appends a new iteration metadata item to the end of the sequence of iteration metadata items. Precondition: None. Postcondition: Let  $t := |n|$  denote the length of sequence  $n$  prior to appending  $m_{\text{new}}$  such that  $m_{t+1} := m_{\text{new}}$ . If  $t = 0$ ,  $n := \langle m_{t+1} \rangle$ . Otherwise,  $n := \langle m_1, m_2, \dots, m_t, m_{t+1} \rangle$ .
- `iterationMetadata( $t$ : Integer)`: IterationMetadata returns the metadata item of the  $t$ th iteration of this KDT process. Precondition:  $t = 1, 2, \dots, |n|$ . Postcondition: `iterationMetadata( $t$ )` :=  $m_t$ .

- `numberOfIterations()`: Integer returns the number of iterations performed during the pattern discovery step of the focal KDT process. Precondition: None. Postcondition: `numberOfIterations() := |n|`.
- `setOfNamedEntityTypes()`: `SetOfNamedEntityTypes` returns set of named entity types, whoses instances are extracted from text units in the focal KDT process. Precondition: None. Postcondition: `setOfNamedEntityTypes() := P`.



# C List of Relevant German Vocabulary

The following list comprises German nouns and verbs that are considered useful to understand the meaning of Commercial Register entries in this case study. This list is mainly based on a translation of the German Commercial Code by Peltzer et al. (2000), which includes a concise introduction to the German Commercial Code as well.

**Abberufung** removal of sb. (e.g., removal of liquidator); e.g., see Peltzer et al. (2000, p. 122)

**Abspaltung** split-off [of company and legal entity, resp.]; broader term: Spaltung (translated as division); e.g., see Benkert and Bürkle (1996, p. 180)

**Aktiengesellschaft (AG)** [German] stock corporation; e.g., see Peltzer and Hickinbotham (1999, p. 35) and Peltzer et al. (2000, p. 43)

**Aktionär** shareholder of German stock corporation (Aktiengesellschaft, AG); e.g., cf. Peltzer and Hickinbotham (1999, p. 37)

**Amtsgericht** [German] District Court; Commercial Registers are usually maintained by the respective District Courts; e.g., cf. Peltzer et al. (2000, pp. 6–7)

**Änderung** change or modification of sth. (e.g., change of firm name or modification of partnership agreement); e.g., see Peltzer et al. (2000, p. 61)

**Aufhebung** dissolution of sth. (e.g., dissolution of a branch office); e.g., cf. Peltzer et al. (2000, pp. 42)

**Auflösung** dissolution of sth. (e.g., dissolution of commercial partnership); e.g., see Peltzer et al. (2000, p. 115)

**Aufsichtsrat** supervisory board of [German] stock corporation (AG); e.g., cf. Peltzer and Hickinbotham (1999, pp. 15–22)

**Ausscheiden** withdrawal [of a partner], [a partner] withdrawing [from a partnership]; e.g., see Peltzer et al. (2000, p. 116)

**Aufspaltung** split-up [of company and legal entity, resp.]; broader term: Spaltung (translated as division); e.g., see Benkert and Bürkle (1996, p. 180)

- Ausgliederung** [asset] drop-down [from company and legal entity, resp.]; broader term: Spaltung (translated as division); e.g., see Benkert and Bürkle (1996, p. 180)
- Beginn** here: commencement (e.g., of a partnership); e.g., cf. Peltzer et al. (2000, p. 102)
- Bedingtes Kapital** contingent capital [of German stock corporation]; e.g., cf. Peltzer and Hickinbotham (1999, p. 27)
- beginnen** here: to commence (e.g., a partnership); e.g., see Peltzer et al. (2000, p. 102)
- Beherrschungsvertrag** control agreement [between two enterprises]; e.g., see Peltzer and Hickinbotham (1999, p. X)
- Bekanntmachung** information that has to be officially published by companies according to the German Commercial Code; e.g., cf. Peltzer et al. (2000, p. 40)
- Bekanntmachungsblatt** publication newspaper; e.g., cf. Peltzer et al. (2000, pp. 40–41)
- Bezugsrecht** subscription right [of shareholders]; e.g., cf. Peltzer and Hickinbotham (1999, p. 25)
- Bundesanzeiger** official German newspaper that regularly publishes Commercial Register entries and corporate news; e.g., cf. Peltzer et al. (2000, p. 40)
- Bestellung** appointment of sb. (e.g., appointment of liquidator); e.g., see Peltzer et al. (2000, p. 122)
- Einteilung, Zerlegung** sub-division [of the capital stock]; e.g., cf. Peltzer and Hickinbotham (1999, p. 45)
- Eintragung** registration [in the Commercial Register]; e.g., see Peltzer et al. (2000, p. 40)
- Eintritt** joining [of a partner], [a partner] joining [a partnership]; e.g., see Peltzer et al. (2000, p. 113)
- Einzelkaufmann, Einzelkauffrau** sole proprietorship; e.g., cf. Peltzer et al. (2000, p. 59)
- Einzelvertretungsbefugnis, Einzelvertretungsmacht** sole power of representation, sole power to [legally] represent [a company]; antonym: Gesamtvertretungsbefugnis, Gesamtvertretungsmacht (translated as joint power of representation); e.g., cf. Peltzer et al. (2000, p. 109)
- Erhöhung** increase in sth. (e.g., increase in share capital or increase in capital contribution); e.g., cf. Peltzer et al. (2000, p. 132)
- Erlöschen** termination of sth. (e.g., termination of Prokura or power to represent a company or firm name); e.g., see Peltzer et al. (2000, p. 61 and p. 66)

- 
- Errichtung** establishment of sth. (e.g., establishment of a branch office); e.g., cf. Peltzer et al. (2000, pp. 40–41)
- erteilen** here: to confer sth. (e.g., to confer Prokura or power to represent a company); e.g., see Peltzer et al. (2000, p. 65)
- Erteilung** conferral of sth. (e.g., conferral of Prokura or power to represent a company); e.g., see Peltzer et al. (2000, p. 66)
- Firma** here: [commercial] firm name; legal name of a company as registered in the respective Commercial Register; legal name under which a merchant transacts business and executes agreements; a merchant may sue and may be sued under his firm name; e.g., see Peltzer et al. (2000, p. 54)
- Forderungen** receivables; e.g., see Peltzer et al. (2000, p. 160)
- Formwechsel** change of [legal] form [of company and legal entity, resp.]; e.g., see Benkert and Bürkle (1996, p. 180)
- Fortführung** continuation (e.g., continuation of business); e.g., see Peltzer et al. (2000, p. 56)
- Gegenstand des Unternehmens** purpose of the enterprise; e.g., see Peltzer et al. (2000, p. 35)
- Genehmigtes Kapital** authorized capital [of German stock corporation]; e.g., cf. Peltzer and Hickinbotham (1999, p. 27)
- Gesamtvertretungsbefugnis, Gesamtvertretungsmacht** joint power of representation, joint power to [legally] represent [a company]; antonym: Einzelvertretungsbefugnis, Einzelvertretungsmacht (translated as sole power of representation); e.g., cf. Peltzer et al. (2000, p. 109)
- Geschäftsanteil** business share [in German registered co-operative association]; e.g., see Peltzer et al. (2000, p. 258)
- Geschäftsführer, Geschäftsführerin** managing director of German limited liability company (GmbH); e.g., see Peltzer et al. (2000, p. 47)
- Geschäftsräume** business premises; e.g., see Peltzer et al. (2000, p. 70)
- Gesellschafter, Gesellschafterin** partner in German commercial partnership (e.g., OHG and KG) or in German limited liability company (GmbH); e.g., cf. Peltzer et al. (2000, p. 56)
- Gesellschaft** here: [commercial] partnership and company, respectively; e.g., see Peltzer et al. (2000, p. 101)

- Gesellschaft mit beschränkter Haftung (GmbH)** [German] limited liability company; e.g., see Peltzer et al. (1996, p. 67)
- Gesellschafterversammlung** meeting of [commercial] partners and share holders, respectively; e.g., see Peltzer et al. (1996, p. 100)
- Gesellschaftsvertrag** [commercial] partnership agreement; e.g., cf. Peltzer et al. (2000, p. 102)
- Gewinnabführungsvertrag** profit transfer agreement [between two or more enterprises]; e.g., see Peltzer and Hickinbotham (1999, p. X)
- Grundkapital** capital stock [of German stock corporation], synonym: Kapital (translated as capital); e.g., see Peltzer and Hickinbotham (1999, p. 35)
- Gründung** here: formation of sth. (e.g., formation of an enterprise); e.g., see Peltzer and Hickinbotham (1999, p. 45)
- Gründungsaufwand** formation expenses [to be borne by a stock corporation]; e.g., see Peltzer and Hickinbotham (1999, p. 56)
- Haftsumme** liability sum [of co-operative member in German registered co-operative association]; e.g., see Peltzer et al. (2000, p. 259)
- Handelsregister** Commercial Register; e.g., see Peltzer et al. (2000, p. 35)
- Handelsregisternummer** Commercial Register number; e.g., cf. Peltzer et al. (1996, p. 91)
- Hauptniederlassung** head office [of company]; e.g., see Peltzer et al. (2000, p. 41)
- Hauptversammlung** general meeting [of German stock corporation]; e.g., see Peltzer and Hickinbotham (1999, p. 130)
- Herabsetzung** reduction of sth. (e.g., reduction of capital contribution); e.g., cf. Peltzer et al. (2000, p. 133)
- Inhaber, Inhaberin** owner of [German] sole proprietorship; e.g., cf. Peltzer et al. (2000, p. 58)
- Inhaberaktie** bearer share; e.g., cf. Peltzer and Hickinbotham (1999, p. 46)
- Insolvenzverfahren** insolvency proceedings; e.g., see Peltzer et al. (2000, p. 116)
- Kapitalerhöhung** capital increase; e.g., cf. Peltzer and Hickinbotham (1999, p. VIII)
- Kapitalherabsetzung** capital reduction; e.g., cf. Peltzer and Hickinbotham (1999, p. VIII)

---

**Kommanditeinlage** capital contribution of limited partner in [German] limited partnership (KG); e.g., see Peltzer et al. (2000, p. 118 and p. 132)

**Kommanditist, Kommanditistin** limited partner in [German] limited partnership (KG); e.g., see Peltzer et al. (2000, p. 128)

**Kommanditgesellschaft (KG)** German limited partnership; e.g., see Peltzer et al. (2000, p. 128)

**Liquidator, Liquidatorin** liquidator [of company]; e.g., see Peltzer et al. (2000, p. 115)

**Löschung** deletion (e.g., of firm names in the Commercial Register); e.g., cf. Peltzer et al. (2000, p. 34 and p. 116)

**Namensaktie** registered share; e.g., cf. Peltzer and Hickinbotham (1999, p. 46)

**Offene Handelsgesellschaft (OHG)** [German] general commercial partnership; e.g., see Peltzer et al. (2000, p. 101)

**persönlich haftender Gesellschafter, persönlich haftende Gesellschafterin** general partner in [German] commercial partnership (e.g., OHG and KG) with unlimited personal liability; e.g., see Peltzer et al. (2000, p. 17 and p. 67)

**Prokura** power to legally represent a company regulated by the German Commercial Code; Prokura includes all judicial and non-judicial transactions that are related to the operations of a commercial business; Prokura might be conferred with either sole or joint power of representation; e.g., see Peltzer et al. (2000, pp. 65–67)

**Prokurist, Prokuristin** Prokura holder; e.g., see Peltzer et al. (2000, p. 67)

**Rechtsform** legal form [of company]; e.g., see Peltzer et al. (2000, p. 248)

**Rechtsträger** legal entity; e.g., see Benkert and Bürkle (1996, p. 180)

**Sacheinlage** contribution in kind; e.g., cf. Peltzer and Hickinbotham (1999, p. 47)

**Satzung** articles of association [of German stock corporation or German limited liability company]; e.g., see Peltzer and Hickinbotham (1999, p. 35)

**Selbstkontraktion** exemption from the restrictions of §181 BGB (German Civil Code); e.g., cf. Peltzer et al. (1996, p. 91)

**Sicherheitsleistung** provision of security, to provide security; e.g., see Peltzer et al. (2000, p. 311)

**Sitz** domicile [of enterprise]; e.g., see Peltzer et al. (2000, p. 41)

- Stammkapital** share capital [of German limited liability company]; e.g., see Peltzer et al. (1996, pp. 68–69)
- Spaltung** division [of company and legal entity, resp.]; e.g., see Benkert and Bürkle (1996, p. 180)
- Tätigkeit** here: purpose of company; e.g., see Peltzer et al. (1996, p. 5)
- übernehmender Rechtsträger** acquiring legal entity; e.g., see Benkert and Bürkle (1996, p. 181)
- übertragender Rechtsträger** transferor legal entity; e.g., see Benkert and Bürkle (1996, p. 181)
- Umwandlung** reorganization [of company and legal entity, resp.]; e.g., see Benkert and Bürkle (1996, p. 180)
- Unternehmen** enterprise; e.g., see Peltzer et al. (2000, p. 41)
- Verbindlichkeiten** liabilities; e.g., see Peltzer et al. (2000, p. 162)
- Vermögensübertragung** transfer of assets [between companies and legal entities, resp.]; e.g., see Benkert and Bürkle (1996, p. 180)
- Verschmelzung** merger [of companies and legal entities, resp.]; e.g., see Benkert and Bürkle (1996, p. 180)
- Vertretungsbefugnis, Vertretungsmacht** power of representation, power to [legally] represent [a company]; e.g., see Peltzer et al. (2000, p. 111)
- Vorstand** managing board [of German stock corporation], synonym: management board; e.g., see Peltzer and Hickinbotham (1999, p. 88) and Peltzer et al. (2000, p. 43)
- Vorstandsmitglied** member of the managing board [of German stock corporations], synonym: member of the management board; e.g., see Peltzer and Hickinbotham (1999, p. 88) and Peltzer et al. (2000, p. 62)
- Zweigniederlassung** branch, branch office [of company]; e.g., see Peltzer et al. (2000, p. 41)
- Zusammensetzung** composition (e.g., composition of the managing board); e.g., see Peltzer et al. (2000, p. 49)

# List of Abbreviations

ACM	Association for Computing Machinery
ADT	abstract data type
AI	artificial intelligence
approx.	approximately
BT	broader term (thesaurus)
cf.	confer, compare
CI	competitive intelligence
DAML	DARPA Agent Markup Language
DAML+OIL	DARPA Agent Markup Language + Ontology Inference Layer
DARPA	Defense Advanced Research Projects Agency
DFG	Deutsche Forschungsgemeinschaft (German Research Foundation)
DIAsDEM	name of research project
DTD	document type definition
e.g.	exempli gratia, for example
EUR	Euro
geb.	geboren am (born on)
GmbH	Gesellschaft mit begrenzter Haftung (limited liability company)
GNU	GNU's Not Unix
HTML	Hypertext Markup Language
i.e.	id est, that is
ID	identifier
IE	information extraction
IEEE	Institute of Electrical and Electronics Engineers
int.	intermediate
IR	information retrieval
ISIN	International Securities Identification Number
ISO	International Organization for Standardization
IT	information technology
KDD	knowledge discovery in databases
KDT	knowledge discovery in textual databases
M&A	mergers and acquisitions
MUC	Message Understanding Conference
NE	named entity
NewsML	News Markup Language
NLP	natural language processing

## *List of Abbreviations*

---

NP	noun phrase
NT	narrower term (thesaurus)
OEM	Object Exchange Model
OLAP	online analytical processing
ORA-SS	Object Relationship Attribute Data Model for Semistructured Data
OWL	Web Ontology Language
p.	page
pp.	pages
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
regex	regular expression
resp.	respectively
s.t.	subject to
sb.	somebody
SEC	Securities and Exchange Commission
smu.	semantically marked-up
SGML	Standard Generalized Markup Language
SNN	shared nearest neighbor
SOM	Self-Organizing Map
SQL	Structured Query Language
sth.	something
TDT	topic detection and tracking
TEI	Text Encoding Initiative
U.S.	United States
UF	used for non-preferred term (thesaurus)
URI	Uniform Resource Identifier
USD	United States Dollar
USE	use for preferred term (thesaurus)
vs.	versus
W3C	World Wide Web Consortium
WKN	Wertpapierkennnummer (German securities identification number)
w.r.t.	with respect to
WWW	World Wide Web
XIRQL	XML IR Query Language
XML	Extensible Markup Language

# Notation and List of Symbols

**Abstract Data Types** Let  $x: T$  denote a variable  $x$  of type  $T$  that can be either a primitive data type described in Appendix B.1 (i.e., Boolean, Integer, Real, Char, and arrays) or an abstract data type (e.g., String) specified in the remainder of Appendix B. If  $T$  is an abstract data type and  $o(\cdot)$  is an operation thereof,  $x.o(\cdot)$  denotes the execution of operation  $o(\cdot)$  on the instantiation  $x$  of abstract data type  $T$ .

**Arrays** Let  $T$  denote a primitive or abstract data type defined in Appendix B.1. The primitive data type that represents an array comprising components of type  $T$  is denoted by  $T[]$ . Let  $x: T[]$  be a variable that represents an array of type  $T$ . A new array of size  $i \in \mathbb{N}$  is instantiated by the expression  $x := T[i]$ . Individual array components are denoted by  $x[j]$ , where  $j \in \mathbb{N}$  and  $j \leq i$ . The constant  $x.size$  denotes the invariant size of the instantiated array  $x$ .

**Multi-Sets** Let the set  $X$  denote an arbitrary domain. A non-empty, finite multi-set  $Y := \{\{y_1, y_2, \dots, y_i\}\}$  is a collection comprising  $i \in \mathbb{N}$  unordered, not necessarily distinct elements  $y_1, y_2, \dots, y_i$  such that  $y_j \in X$ , where  $j = 1, 2, \dots, i$ . Analogous to sets, the number of elements in multi-set  $Y$  (i.e., its cardinality) is denoted by  $|Y| \in \mathbb{N}$ , and an empty multi-set is denoted by  $\emptyset$ , such that  $|\emptyset| = 0$ . Unlike sets, however, multi-sets may comprise duplicate elements.

**Relations** Given  $n \in \mathbb{N}$  arbitrary, not necessarily distinct, and atomic domains  $D_i$  ( $i = 1, 2, \dots, n$ ), relation  $R \subseteq D_1 \times \dots \times D_n$  is defined as the Cartesian product of  $n$  domains or as a subset thereof (cf. Codd, 1970, 1990). Each domain  $D_i$  is associated with a descriptive attribute name  $a_i \in \{a_1, \dots, a_n\}$ . The schema of relation  $R$  is denoted by  $R(a_1: D_1, \dots, a_n: D_n)$ . An  $n$ -ary element  $(d_1, \dots, d_n) \in R$  with attribute values  $d_i \in D_i$  is referred to as a tuple of relation  $R$ .

**Sequences** Let the set  $X$  denote an arbitrary domain. A non-empty sequence  $x := \langle x_1, x_2, \dots, x_i \rangle$  is a tuple comprising  $i \in \mathbb{N}$  ordered elements  $x_1, x_2, \dots, x_i$  such that  $x_j \in X$ , where  $j = 1, 2, \dots, i$ . The number of elements  $|x| \in \mathbb{N}$  in sequence  $x$  (i.e., its length) is not necessarily constant. The  $j$ th element in sequence  $x$  is denoted by  $x[j]$ , where  $j = 1, 2, \dots, |x|$ . An empty sequence is denoted by  $\varepsilon$  ( $|\varepsilon| = 0$ ).

**Tuples** A  $k$ -tuple  $\mathbf{x} := (x_1, x_2, \dots, x_k)$  comprises  $k \in \mathbb{N}$  ordered elements  $x_1, x_2, \dots, x_k$ . If the sets  $X_i$ , where  $i = 1, 2, \dots, k$ , denote  $k$  arbitrary domains, then  $\mathbf{x} := (x_1: X_1, x_2: X_2, \dots, x_k: X_k)$  defines a  $k$ -tuple such that  $x_i \in X_i$ . The number of elements  $|\mathbf{x}| = k$  in  $k$ -tuple  $\mathbf{x}$  (i.e., its arity) is constant. Elements  $x_i$  of  $k$ -tuple  $\mathbf{x}$  may take the null value (i.e.,  $x_i = \text{null}$ ) if  $x_i$  is unknown, inappropriate, or non-existent. The  $i$ th element in  $k$ -tuple  $\mathbf{x}$  is denoted by  $\mathbf{x}[i]$ .

**Vectors** Let  $\mathbf{x} = [\xi_1, \xi_2, \dots, \xi_m]^T$  and  $\mathbf{y} = [\eta_1, \eta_2, \dots, \eta_m]^T$  denote two  $m$ -dimensional column vectors, where  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$  and  $m \in \mathbb{N}$ . The scalar product of vectors  $\mathbf{x}$  and  $\mathbf{y}$  is denoted by  $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^m \xi_i \cdot \eta_i$ . The Euclidean norm of vector  $\mathbf{x}$  is denoted by  $\|\mathbf{x}\| = (\sum_{i=1}^m \xi_i^2)^{1/2}$ .

### List of Symbols

$:=$	assignment operator
$\emptyset$	empty set
$\hat{\mathbf{A}}$	set of semantically marked-up text units
$\check{\mathbf{a}}$	text archive
$\bar{\mathbf{a}}$	intermediate text archive
$\hat{\mathbf{a}}$	semantically marked-up text archive
$\hat{a}_{\text{Attribute}}$	estimate of XML attribute accuracy
$\hat{a}_{\text{TagName}}$	estimate of XML tag name accuracy
$C_{t,i}$	text unit cluster $i \in \mathbb{N}$ in clustering iteration $t \in \mathbb{N}$
$C_t$	text unit clustering in clustering iteration $t \in \mathbb{N}$
$D$	set of content descriptors
$d$	content descriptor
$\text{df}(\cdot)$	document frequency
$\text{dist}(\cdot)$	distance
$E$	set of named entities
$\bar{E}$	set of intermediate named entities
$e$	named entity
$\check{e}$	canonical form of named entity $e$
$\bar{e}$	intermediate named entity
$f$	parameterizable KDT process flow
$\tilde{f}$	parameterized KDT process flow
$g$	parameterizable KDT algorithm
$\tilde{g}$	parameterized KDT algorithm
$\text{idf}(\cdot)$	inverse document frequency
$k$	KDT process metadata
$\check{l}$	label
$\ln(\cdot)$	logarithmus naturalis
$m$	iteration metadata

---

$\mathbb{N}$	set of natural numbers
$n_{CC}$	number of completely correct named entities
$n_{FN}$	number of false negatives
$n_{FP}$	number of false positives
$n_I$	number of incorrect named entities
$n_M$	number of missing named entities
$n_{NE}$	number of relevant named entities
$n_{PC}$	number of partially correct named entities
$n_{TN}$	number of true negatives
$n_{TP}$	number of true positives
null	null value
$\text{ntf}(\cdot)$	normalized term frequency
$\mathbb{O}$	set of concepts
$o$	sequence of concepts
$o$	concept
$\mathbb{P}$	set of named entity types
$p$	named entity type
$p_{AS}$	attribute support threshold
$p_{DD}$	dominant descriptor threshold
$p_{\max DC}$	maximum descriptor coverage
$p_{\min CS}$	minimum cluster size
$p_{\min DD}$	minimum descriptor dominance
$p_{RD}$	rare descriptor threshold
$\hat{p}_{\text{Attribute}}$	estimate of XML attribute precision
$\hat{p}_{\text{TagName}}$	estimate of XML tag name precision
$\text{prox}(\cdot)$	proximity
$q$	cluster quality criteria
$\mathbb{R}$	set of real numbers
$\hat{r}_{\text{Attribute}}$	estimate of XML attribute recall
$\hat{r}_{\text{TagName}}$	estimate of XML tag name recall
$\check{r}$	text unit layer
$\bar{r}$	intermediate text unit layer
$\hat{r}$	semantically marked-up text unit layer
$s$	number of semantically marked-up text units in sample
$\hat{s}$	conceptual document structure
$\text{sim}(\cdot)$	similarity
$\mathbf{T}$	text vectors by content descriptors matrix
$\mathbf{t}$	text vector
$\check{t}$	text document
$\bar{t}$	intermediate text document
$\hat{t}$	semantically marked-up text document
$\text{tf}(\cdot)$	term frequency

$\text{tudf}(\cdot)$	text unit descriptor frequency
$\text{tuf}(\cdot)$	text unit frequency
$\tilde{u}$	text unit
$\ddot{u}$	tokenized text unit
$\mathbf{U}$	text unit vectors by text unit descriptors matrix
$\mathbf{u}$	text unit vector (vector-space model)
$\vec{u}$	text unit vector (abstract data type)
$\bar{u}$	intermediate text unit
$\hat{u}$	semantically marked-up text unit
$V$	controlled vocabulary
$v$	controlled vocabulary term
$k$	descriptor weighting schema
$\check{W}$	set of tokens
$\check{W}_{\ddot{u}}$	set of tokens in tokenized text unit $\ddot{u}$
$\tilde{w}$	token
$\hat{x}_{\text{Docs}}$	array of semantically tagged XML documents
$\hat{x}_{\text{DTD}}$	concept-based XML document type definition
$\mathbb{Z}$	set of integer numbers

# Bibliography

- Serge Abiteboul, Peter Buneman, and Dan Siciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufman Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, 2000.
- Mohammad Abolhassani, Norbert Fuhr, and Norbert Gövert. Information extraction and automatic markup for XML documents. In Henk M. Blanken, Torsten Grabs, Hans-Jörg Schek, Ralf Schenkel, and Gerhard Weikum, editors, *Intelligent Search on XML Data: Applications, Languages, Models, Implementations, and Benchmarks*, volume 2818 of *Lecture Notes in Computer Science*, pages 159–174. Springer-Verlag, Berlin, Heidelberg, 2003.
- Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 70–81, Dallas, TX, USA, May 2000. ACM Press.
- Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 85–94, San Antonio, TX, USA, June 2000. ACM Press.
- Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, Santiago de Chile, Chile, September 1994. Morgan Kaufmann.
- Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, DC, USA, May 1993. ACM Press.
- Rakesh Agrawal, Roberto Bayardo, and Ramakrishnan Srikant. Athena: Mining-based interactive management of text databases. In Carlo Zaniolo, Peter C. Lockemann, Marc H. Scholl, and Torsten Grust, editors, *Advances in Database Technology - EDBT 2000: 7th International Conference on Extending Database Technology, Konstanz, Germany, March 2000: Proceedings*, volume 1777 of *Lecture Notes in Computer Science*, pages 365–379. Springer-Verlag, Berlin, Heidelberg, 2000.
- Vincent Aguilera, Sophie Cluet, Tova Milo, Pierangelo Veltri, and Dan Vodislav. Views in a large-scale XML repository. *The VLDB Journal*, 11(3):238–255, November 2002.

- Helena Ahonen. Automatic generation of SGML content models. *Electronic Publishing: Origination, Dissemination, and Design*, 8(2/3):195–206, June/September 1995.
- Helena Ahonen. *Generating Grammars for Structured Documents Using Grammatical Inference Methods*. PhD thesis, Department of Computer Science, University of Helsinki, Helsinki, Finland, November 1996.
- Alfredo Alba, Varun Bhagwan, Mike Ching, Alex Cozzi, Raj Desai, Daniel Gruhl, Kevin Haas, Linda Kato, Jeff Kusnitz, Bryan Langston, Ferdy Nagy, Linda A. Nguyen, Jan Pieper, Savitha Srinivasan, Anthony Stuart, and Renjie Tang. A funny thing happened on the way to a billion. *IEEE Data Engineering Bulletin*, 29(4):27–36, December 2006.
- James Allan, editor. *Topic Detection and Tracking: Event-Based Information Organization*. Kluwer International Series on Information Retrieval. Kluwer Academic Publishers, Boston, Dordrecht, 2002.
- James Allan, Victor Lavrenko, and Russell Swan. Explorations within topic tracking and detection. In James Allan, editor, *Topic Detection and Tracking: Event-Based Information Organization*, Kluwer International Series on Information Retrieval, pages 197–224. Kluwer Academic Publishers, Boston, Dordrecht, 2002.
- Sihem Amer-Yahia, Pat Case, Thomas Rölleke, Jayavel Shanmugasundaram, and Gerhard Weikum. Report on the DB/IR panel at SIGMOD 2005. *ACM SIGMOD Record*, 34(4):71–74, December 2005.
- Morgan Ames and Mor Naaman. Why we tag: Motivations for annotation in mobile and online media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 971–980, San Jose, CA, USA, April/May 2007. ACM Press.
- Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Myers, and Mabry Tyson. SRI International FASTUS system MUC-6 test results and analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 237–248, Columbia, MD, USA, November 1995. Morgan Kaufmann.
- Arnulfo P. Azcarraga, Teddy N. Yap, Jonathan Tan, and Tat Seng Chua. Evaluating keyword selection methods for WEBSOM text archives. *IEEE Transactions on Knowledge and Data Engineering*, 16(3):380–383, March 2004.
- Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, New York, 1999.
- Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2670–2676, Hyderabad, India, January 2007.

- David W. Barron. Why use SGML? *Electronic Publishing – Origination, Dissemination, and Design*, 2(1):3–24, April 1989.
- Petra Saskia Bayerl, Harald Lungen, Daniela Goecke, Andreas Witt, and Daniel Naber. Methods for the semantic analysis of document markup. In *Proceedings of the 2003 ACM Symposium on Document Engineering*, pages 161–170, Grenoble, France, November 2003a. ACM Press.
- Petra Saskia Bayerl, Harald Lungen, Ulrike Gut, and Karsten Ingmar Paul. Methodology for reliable schema development and evaluation of manual annotations. In *Proceedings of the K-CAP 2003 Workshop on Knowledge Markup and Semantic Annotation (SEMANNOT 2003)*, Sanibel, FL, USA, October 2003b.
- Sean Bechhofer and Carole Goble. COHSE: Conceptual Open Hypermedia Service. In Siegfried Handschuh and Steffen Staab, editors, *Annotation for the Semantic Web*, volume 96 of *Frontiers in Artificial Intelligence and Applications*, pages 193–211. IOS Press, Amsterdam, Berlin, 2003.
- Richard Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton (New Jersey), 1961.
- Manfred Benkert and Annegret Bürkle. *Law of Reorganization / Reorganization Tax Law: Bilingual Edition (Ger./Engl.) with Comprehensive Introduction*. RWS Verlag Kommunikationsforum GmbH, Köln, 1996.
- Sonia Bergamaschi, Silvana Castano, and Maurizio Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, March 1999.
- Bryan Bergeron. *Essentials of XBRL: Financial Reporting in the 21st Century*. John Wiley & Sons, Hoboken (New Jersey), 2003.
- Pavel Berkhin. A survey of clustering data mining techniques. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 25–71. Springer-Verlag, Berlin, Heidelberg, 2006.
- Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284(5):29–37, May 2001.
- Geert Jan Bex, Frank Neven, and Stijn Vansummeren. Inferring XML Schema definitions from XML data. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 998–1009, Vienna, Austria, September 2007. VLDB Endowment.

- Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is “nearest neighbor” meaningful? In Catriel Beeri and Peter Buneman, editors, *Database Theory - ICDT'99: 7th International Conference, Jerusalem, Israel, January 1999: Proceedings*, volume 1540 of *Lecture Notes on Computer Science*, pages 217–235. Springer-Verlag, Berlin, Heidelberg, 1999.
- James C. Bezdek and Nikhil R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 28(3):301–315, June 1998.
- Alejandro Bia, Rafael C. Carrasco, and Mikel L. Forcada. Identifying a reduced DTD from marked up documents. In *Proceedings of the IX Spanish Symposium on Pattern Recognition and Image Analysis (SNRFAI-2001)*, pages 385–390, Castellón, Spain, May 2001.
- Cliff Binstock, Dave Peterson, Mitchell Smith, Mike Wooding, Chris Dix, and Chris Galtenberg. *The XML Schema Complete Reference*. Addison-Wesley, Boston, San Francisco, 2003.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.
- Lynne Bowker and Jennifer Pearson, editors. *Working with Specialized Language: A practical guide to using corpora*. Routledge, London, New York, 2002.
- Stefanie Brüninghaus and Kevin D. Ashley. Improving the representation of legal case texts with information extraction methods. In *Proceedings of the 8th International Conference on Artificial Intelligence and Law*, pages 42–51, St. Louis, Missouri, United States, 2001. ACM Press.
- Paul Buitelaar and Thierry Declerck. Linguistic annotation for the Semantic Web. In Siegfried Handschuh and Steffen Staab, editors, *Annotation for the Semantic Web*, volume 96 of *Frontiers in Artificial Intelligence and Applications*, pages 93–111. IOS Press, Amsterdam, Berlin, 2003.
- Paul Buitelaar, Daniel Olejnik, and Michael Sintek. A Protégé plug-in for ontology extraction from text based on linguistic analysis. In Christoph Bussler, John Davies, Dieter Fensel, and Rudi Studer, editors, *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece, May 10-12, 2004, Proceedings*, volume 3053 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, 2004.
- Peter Buneman. Semistructured data. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 117–121, Tucson, AZ, USA, May 1997. ACM Press.

- Peter Buneman, Susan Davidson, Mary Fernandez, and Dan Suciu. Adding structure to unstructured data. In Foto N. Afrati and Phokion G. Kolaitis, editors, *Database Theory — ICDT '97: 6th International Conference, Delphi, Greece, January 8–10, 1997: Proceedings*, volume 1186 of *Lecture Notes in Computer Science*, pages 336–350. Springer-Verlag, Berlin, Heidelberg, 1997.
- Razvan Bunescu and Raymond J. Mooney. Collective information extraction with Relational Markov Networks. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 438, Barcelona, Spain, July 2004. Association for Computational Linguistics.
- Lou Burnard. What is SGML and how does it help? In Nancy Ide and Jean Véronis, editors, *Text Encoding Initiative: Background and Context*, pages 41–50. Kluwer Academic Publishers, Dordrecht, Boston, 1995. Reprinted from *Computers and the Humanities*, Volume 29, Nos. 1, 2 & 3 (1995), edited by Glyn Holmes (With the addition of an SGML/TEI Bibliography).
- Terry Butler, Sue Fisher, Greg Coulombe, Patricia Clements, Isobel Grundy, Susan Brown, Jean Wood, and Rebecca Cameron. Can a team tag consistently? Experiences on the Orlando project. *Markup Languages: Theory and Practice*, 2(2):111–125, Spring 2000.
- Mary Elaine Califf and Raymond J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Machine Learning Research*, 4: 177–210, December 2003.
- James P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310. Springer-Verlag, July 1994.
- Sharon A. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 120–126, College Park, MD, USA, June 1999. Association for Computational Linguistics.
- David Carmel, Yoelle S. Maarek, Matan Mandelbrod, Yosi Mass, and Aya Soffer. Searching XML documents via XML Fragments. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 151–158, Toronto, Canada, 2003. ACM Press.
- Donald O. Case. *Looking for Information: A Survey of Research on Information Seeking, Needs, and Behavior*. Library and Information Science. Academic Press, Amsterdam, London, 2002.

- Malú Castellanos. HotMiner: Discovering hot topics from dirty text. In Michael W. Berry, editor, *Survey of Text Mining: Clustering, Classification, and Retrieval*, pages 123–157. Springer-Verlag, New York, Berlin, 2004.
- Venkatesan T. Chakaravarthy, Himanshu Gupta, Prasan Roy, and Mukesh Mohania. Efficiently linking text documents with relevant structured information. In *Proceedings of 32nd International Conference on Very Large Data Bases*, pages 667–678, Seoul, Korea, September 2006. VLDB Endowment.
- Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufman Series in Data Management Systems. Morgan Kaufman Publishers, Amsterdam, Boston, 2003.
- Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal*, 7(3):163–178, August 1998.
- Daniel Chandler. *Semiotics: The Basics*. Routledge, London, New York, 2002.
- Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, and Khaled F. Shaalan. A survey of Web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, October 2006.
- Akmal B. Chaudhri, Awais Rashid, and Roberto Zicari, editors. *XML Data Management: Native XML and XML-Enabled Database Systems*. Addison-Wesley, Boston, San Francisco, 2003.
- Hsinchun Chen, Tobun D. Ng, Joanne Martinez, and Bruce R. Schatz. A concept space approach to addressing the vocabulary problem in scientific information retrieval: An experiment on the worm community system. *Journal of the American Society for Information Science*, 48(1):17–31, January 1997.
- Hsinchun Chen, Michael Chau, and Daniel Zeng. CI Spider: A tool for competitive intelligence on the Web. *Decision Support Systems*, 34(1):1–17, December 2002.
- Hsinchun Chen, Ann M. Lally, Bin Zhu, and Michael Chau. HelpfulMed: Intelligent searching for medical information over the Internet. *Journal of the American Society for Information Science and Technology*, 54(7):683–694, May 2003.
- Ong Siou Chin, Narayanan Kulathuramaiyer, and Alvin W. Yeo. Automatic discovery of concepts from text. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 1046–1049, Hong Kong, December 2006. IEEE Computer Society.

- Jennifer Chu-Carroll, John Prager, Krzysztof Czuba, David Ferrucci, and Pablo Duboue. Semantic search via XML fragments: A high-precision approach to IR. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 445–452, Seattle, WA, USA, 2006. ACM Press.
- Christina Yip Chung, Raymond Lieu, Jinhui Liu, Alpha Luk, Jianchang Mao, and Prabhakar Raghavan. Thematic mapping – from unstructured documents to taxonomies. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 608–610, McLean, VA, USA, November 2002. ACM Press.
- Philipp Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer Science+Business Media, New York, 2006.
- Fabio Ciravegna and Yorick Wilks. Designing adaptive information extraction for the Semantic Web in Amilcare. In Siegfried Handschuh and Steffen Staab, editors, *Annotation for the Semantic Web*, volume 96 of *Frontiers in Artificial Intelligence and Applications*, pages 112–127. IOS Press, Amsterdam, Berlin, 2003.
- Fabio Ciravegna, Alexiei Dingli, Yorick Wilks, and Daniela Petrelli. Using adaptive information extraction for effective human-centred document annotation. In Jürgen Franke, Gholamreza Nakhaeizadeh, and Ingrid Renz, editors, *Text Mining: Theoretical Aspects and Applications*, Advances in Soft Computing, pages 153–163. Physica-Verlag, Heidelberg, 2003.
- Edgar F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.
- Edgar F. Codd. *The Relational Model for Database Management: Version 2*. Addison-Wesley Publishing Company, Reading (Massachusetts), Menlo Park (California), 1990.
- William F. Cody, Jeffrey T. Kreulen, Vikas Krishna, and W. Scott Spangler. The integration of business intelligence and knowledge management. *IBM Systems Journal*, 41(4):697–713, 2002.
- Robert M. Colomb. *Information Spaces: The Architecture of Cyberspace*. Springer-Verlag, London, Berlin, 2002.
- James H. Coombs, Allen H. Renear, and Steven J. DeRose. Markup systems and the future of scholarly text processing. *Communications of the ACM*, 30(11):933–947, November 1987.
- Jim Cowie and Wendy Lehnert. Information extraction. *Communications of the ACM*, 39(1):80–91, January 1996.

- Hang Cui, Ji-Rong Wen, and Tat-Seng Chua. Hierarchical indexing and flexible element retrieval for structured document. In Fabrizio Sebastiani, editor, *Advances in Information Retrieval: 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14–16, 2003: Proceedings*, volume 2633 of *Lecture Notes in Computer Science*, pages 73–87. Springer-Verlag, Berlin, Heidelberg, 2003.
- Hamish Cunningham. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36(2):223–254, May 2002.
- Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Cristian Ursu, and Marin Dimitrov. *Developing Language Processing Components with GATE (a User Guide): For GATE version 2.0*. The University of Sheffield, Sheffield, 2002.
- Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, Copenhagen, Denmark, June 1992. ACM Press.
- Nell Dale and Henry M. Walker. *Abstract Data Types: Specifications, Implementations, and Applications*. D. C. Heath and Company, Lexington (Massachusetts), Toronto, 1996.
- Thomas H. Davenport and Laurence Prusak. *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press, Boston, 2000. Paperback Edition.
- Scott Deerwester, Susan T. Durnais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.
- Steven J. DeRose, David G. Durand, Elli Mylonas, and Allen H. Renear. What is text, really? *Journal of Computing in Higher Education*, 1(2):3–26, Winter 1990.
- Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1-2):143–175, January-February 2001.
- Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Toms, John A. Tomlin, and Jason Y. Zien. SemTag and Seeker: Bootstrapping the Semantic Web via automated semantic annotation. In *Proceedings of the Twelfth International Conference on World Wide Web*, pages 178–186, Budapest, Hungary, May 2003. ACM Press.
- Hong-Hai Do and Erhard Rahm. Matching large schemas: Approaches and evaluation. *Information Systems*, 32(6):857–885, September 2007.

- Jochen Dörre, Peter Gerstl, and Roland Seiffert. Text Mining. In Hajo Hippner, Ulrich Küsters, Matthias Meyer, and Klaus Wilde, editors, *Handbuch Data Mining im Marketing: Knowledge Discovery in Marketing Databases*, pages 465–488. Friedr. Vieweg & Sohn Verlagsgesellschaft, Braunschweig, Wiesbaden, 2001. In German.
- Doug Downey, Matthew Broadhead, and Oren Etzioni. Locating complex named entities in Web text. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2733–2739, Hyderabad, India, January 2007.
- Peter F. Drucker. The coming of the new organization. *Harvard Business Review*, 66(1): 45–53, January-February 1988.
- Richard C. Dubes and Anil K. Jain. Clustering techniques: The user’s dilemma. *Pattern Recognition*, 8(4):247–260, 1976.
- Angela Edmunds and Anne Morris. The problem of information overload in business organisations: A review of the literature. *International Journal of Information Management*, 20(1):17–28, February 2000.
- M. Erdmann, A. Maedche, H.-P. Schnurr, and S. Staab. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. *Linköping Electronic Articles in Computer and Information Science*, 6(2), 2001. Available from <http://www.ep.liu.se/ea/cis/2001/002/>, accessed 2008-08-01.
- Henrik Eriksson. An annotation tool for semantic documents (system description). In *The Semantic Web: Research and Applications: 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007. Proceedings*, volume 4519 of *Lecture Notes in Computer Science*, pages 759–768. Springer-Verlag, Berlin, Heidelberg, 2007.
- Levent Ertöz, Michael Steinbach, and Vipin Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the Third SIAM International Conference on Data Mining*, San Francisco, CA, USA, May 2003. Society for Industrial and Applied Mathematics.
- Levent Ertöz, Michael Steinbach, and Vipin Kumar. Finding topics in collections of documents: A shared nearest neighbor approach. In Weili Wu, Hui Xiong, and Shashi Shakhara, editors, *Clustering and Information Retrieval*, volume 11 of *Network Theory and Applications*, pages 83–103. Kluwer Academic Publishers, Boston, Dordrecht, 2004.
- Martin Ester and Jörg Sander. *Knowledge Discovery in Databases: Techniken und Anwendungen*. Springer-Verlag, Berlin, Heidelberg, 2000. In German.
- Vladimir Estivill-Castro. Why so many clustering algorithms: A position paper. *ACM SIGKDD Explorations Newsletter*, 4(1):65–75, June 2002.

- Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Webscale information extraction in KnowItAll (preliminary results). In *Proceedings of the 13th International Conference on World Wide Web*, pages 100–110, New York, NY, USA, May 2004. ACM Press.
- Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence*, 165(1):91–134, June 2005.
- Oren Etzioni, Michele Banko, and Michael J. Cafarella. Machine reading. In *Machine Reading: Papers from 2007 AAAI Spring Symposium: Technical Report SS-07-06*, pages 1–5, Stanford, CA, USA, March 2007. The AAAI Press.
- Brian S. Everitt, Sabine Landau, and Morven Leese, editors. *Cluster Analysis*. Arnold (Hodder Headline Group), London, fourth edition, 2001.
- Stefan Evert. Significance tests for the evaluation of ranking methods. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 945–951, Geneva, Switzerland, August 2004.
- Ali F. Farhoomand and Don H. Drury. Managerial information overload. *Communications of the ACM*, 45(10):127–131, October 2002.
- Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54, Fall 1996a.
- Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery: An overview. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press / The MIT Press, Menlo Park (California) / Cambridge (Massachusetts), London, 1996b.
- Ronen Feldman and Ido Dagan. Knowledge discovery in textual databases (KDT). In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 112–117, Montreal, Canada, August 1995. AAAI Press.
- Ronen Feldman and James Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, Cambridge, New York, 2007.
- Ronen Feldman, Moshe Fresko, Yakkov Kinar, Yehuda Lindell, Orly Liphstat, Martin Rajman, Yonatan Schler, and Oren Zamir. Text mining at the term level. In Jan Żytkow and Mohamed Quafafou, editors, *Principles of Data Mining and Knowledge*

- 
- Discovery, Second European Symposium, PKDD '98, Nantes, France, September 23-26, 1998, Proceedings*, volume 1510 of *Lecture Notes in Computer Science*, pages 65–73. Springer-Verlag, Berlin, Heidelberg, 1998.
- Ronen Feldman, Yonatan Aumann, and Moshe Fresko. Text mining via information extraction. In Jan Żytkow and Jan Rauch, editors, *Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD '99, Prague, Czech Republic, September 15-18, 1999, Proceedings*, volume 1704 of *Lecture Notes in Computer Science*, pages 165–173. Springer-Verlag, Berlin, Heidelberg, 1999.
- Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge (Massachusetts), London, 1998.
- Dieter Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, Berlin, Heidelberg, second edition, 2004.
- Dieter Fensel, James Hendler, Henry Lieberman, and Wolfgang Wahlster. Introduction. In Dieter Fensel, James Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, pages 1–25. The MIT Press, Cambridge (Massachusetts), London, 2003.
- David Ferrucci and Adam Lally. Building an example application with the Unstructured Information Management Architecture. *IBM Systems Journal*, 43(3):455–475, 2004.
- Edward W. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21(3):768–769, September 1965. Abstract of paper presented at the Spring Meeting of E N A R held at Florida State University at Tallahassee, Florida on April 29 to May 1, 1965.
- George Forman. Tackling concept drift by temporal inductive transfer. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 252–259, Seattle, WA, USA, August 2006. ACM Press.
- George Forman, Evan Kirshenbaum, and Jaap Suermondt. Pragmatic text mining: Minimizing human effort to quantify many issues in call logs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 852–861, Philadelphia, PA, USA, August 2006. ACM Press.
- S. Mehan Forno, L. Farinetti. Can data mining techniques ease the semantic tagging burden? In *Proceedings of SWDB'03: The First International Workshop on Semantic Web and Databases: Co-Located with VLDB 2003*, pages 277–292, Berlin, Germany, September 2003.
- D. J. Foskett. Thesaurus. In Karen Sparck Jones and Peter Willet, editors, *Readings in Information Retrieval*, pages 111–134. Morgan Kaufmann Publishers, San Francisco, 1997. Reprinted by permission.

- Christopher Fox. Lexical analysis and stoplists. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 102–130. Prentice Hall PTR, Upper Saddle River (New Jersey), 1992.
- William B. Frakes. Stemming algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 131–160. Prentice Hall PTR, Upper Saddle River (New Jersey), 1992.
- William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval: Data Structures & Algorithms*. Prentice Hall PTR, Upper Saddle River (New Jersey), 1992.
- William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. Knowledge discovery in databases: An overview. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–27. AAAI Press / The MIT Press, Menlo Park (California) / Cambridge (Massachusetts), London, 1991.
- Dayne Freitag. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2–3):169–202, May–June 2000.
- Norbert Fuhr. XML information retrieval and information extraction. In Jürgen Franke, Gholamreza Nakhaeizadeh, and Ingrid Renz, editors, *Text Mining: Theoretical Aspects and Applications*, Advances in Soft Computing, pages 21–32. Physica-Verlag, Heidelberg, 2003.
- Norbert Fuhr and Kai Großjohann. XIRQL: An XML query language based on information retrieval concepts. *ACM Transactions on Information Systems*, 22(2):313–356, April 2004.
- Leonard M. Fuld. *The New Competitor Intelligence: The Complete Resource for Finding, Analyzing, and Using Information About Your Competitors*. New Directions in Business. John Wiley & Sons, New York, Chichester, 1995.
- Robert D. Galliers. Choosing appropriate information systems research approaches: A revised taxonomy. In Hans-Erik Nissen, Heinz K. Klein, and Rudy Hirschheim, editors, *Information Systems Research: Contemporary Approaches & Emergent Traditions*, pages 327–345. North-Holland, Amsterdam, New York, 1991.
- John F. Gantz, David Reinsel, Christopher Chute, Wolfgang Schlichting, John McArthur, Stephen Minton, Irida Xheneti, Anna Toncheva, and Alex Manfrediz. The expanding digital universe: A forecast of worldwide information growth through 2010. IDC white paper sponsored by EMC. Retrieved from [http://www.emc.com/about/destination/digital\\_universe/pdf/Expanding\\_Digital\\_Universe\\_IDC\\_WhitePaper\\_022507.pdf](http://www.emc.com/about/destination/digital_universe/pdf/Expanding_Digital_Universe_IDC_WhitePaper_022507.pdf) on 2008-08-01, 2007.
- Peter Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. The MIT Press, Cambridge (Massachusetts), London, 2000.

- Minos Garofalakis, Aristides Gionis, Rajeev Rastogi, S. Seshadri, and Kyuseok Shim. XTRACT: A system for extracting document type descriptors from XML documents. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 165–176, Dallas, TX, USA, 2000. ACM Press.
- Minos Garofalakis, Aristides Gionis, Rajeev Rastogi, S. Seshadri, and Kyuseok Shim. XTRACT: Learning document type descriptors from XML document collections. *Data Mining and Knowledge Discovery*, 7(1):23–56, January 2003.
- Patrick A. Gaughan. *Mergers, Acquisitions, and Corporate Restructurings*. John Wiley & Sons, New York, third edition, 2002.
- Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3):9, September 2006.
- John H. Gennari, Mark A. Musen, Ray W. Ferguson, William E. Grosso, Monica Crubézy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu. The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1):89–123, January 2003.
- Filippo Geraci, Marco Pellegrini, Marco Maggini, and Fabrizio Sebastiani. Cluster generation and cluster labelling for Web snippets: A fast and accurate hierarchical solution. In Fabio Crestani, Paolo Ferragina, and Mark Sanderson, editors, *String Processing and Information Retrieval: 13th International Conference, SPIRE 2006, Glasgow, UK, October 11-13, 2006: Proceedings*, volume 4209 of *Lecture Notes in Computer Science*, pages 25–36. Springer-Verlag, Berlin, Heidelberg, 2006.
- John Gerdes Jr. EDGAR-Analyzer: Automating the analysis of corporate data contained in the SEC’s EDGAR database. *Decision Support Systems*, 35(1):7–29, April 2003.
- Vladimir Geroimenko. *Dictionary of XML Technologies and the Semantic Web*. Springer Professional Computing. Springer-Verlag, London, Berlin, 2004.
- Joydeep Ghosh and Alexander Strehl. Similarity-based text clustering: A comparative study. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, *Grouping Multi-dimensional Data: Recent Advances in Clustering*, pages 73–97. Springer-Verlag, Berlin, Heidelberg, 2006.
- Natalie Glance, Matthew Hurst, Kamal Nigam, Matthew Siegler, Robert Stockton, and Takashi Tomokiyo. Deriving marketing intelligence from online discussion. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 419–428. ACM Press, August 2005.
- Eric Glover, David M. Pennock, Steve Lawrence, and Robert Krovetz. Inferring hierarchical descriptions. In *Proceedings of the Eleventh International Conference on Information*

- and Knowledge Management*, pages 507–514, McLean, VA, USA, November 2002. ACM Press.
- Eric J. Glover, Stephen R. Lawrence, and David M. Pennock. Inferring hierarchical descriptions of a set of documents. United States Patent No. 20030167163 filed through NEC Research Institute, Inc., September 2003.
- Cliff Goddard. *Semantic Analysis: A Practical Introduction*. Oxford Textbooks in Linguistics. Oxford University Press, Oxford, New York, 1998.
- C. F. Goldfarb. A generalized approach to document markup. In *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation*, pages 68–73, 1981.
- Roy Goldman and Jennifer Widom. DataGuides: Enabling query formulation and optimization in semistructured databases. In *Proceedings of 23rd International Conference on Very Large Data Bases*, pages 436–445, Athens, Greece, August 1997. Morgan Kaufmann.
- Roy Goldman and Jennifer Widom. Approximate DataGuides. In *Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, pages 436–445, Jerusalem, Israel, January 1999.
- Earl Gose, Richard Johnsonbaugh, and Steve Jost. *Pattern Recognition and Image Analysis*. Prentice Hall PTR, Upper Saddle River (New Jersey), 1996.
- Johannes Grabmeier and Andreas Rudolph. Techniques of cluster algorithms in data mining. *Data Mining and Knowledge Discovery*, 6(4):303–360, October 2002.
- Gregory Grefenstette. *Explorations in Automatic Thesaurus Discovery*, volume 278 of *Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Dordrecht, 1994.
- Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *PNAS: Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl. 1): 5228–5235, April 2004.
- Ralph Grishman. Information extraction: Techniques and challenges. In Maria Teresa Pazienza, editor, *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology: International Summer School, SCIE-97, Frascati, Italy, July 14-18, 1997*, volume 1299 of *Lecture Notes in Computer Science*, pages 10–27. Springer-Verlag, Berlin, Heidelberg, 1997.
- David A. Grossman and Ophir Frieder. *Information Retrieval: Algorithms and Heuristics*. The Kluwer International Series on Information Retrieval. Springer, Dordrecht, second edition, 2004.

- Thomas R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, June 1993.
- Daniel Gruhl, Laurent Chavet, David Gibson, Jörg Meyer, Pradhan Pattanayak, Andrew Tomkins, and Jason Zien. How to build a WebFountain: An architecture for very large-scale text analytics. *IBM Systems Journal*, 43(1):64–77, 2004.
- R. Guha and Rob McCool. TAP: A Semantic Web platform. *Computer Networks*, 42(5):557–577, August 2003.
- John V. Guttag and James J. Horning. The algebraic specification of abstract data types. *Acta Informatica*, 10(1):27–52, 1978.
- Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.
- Richard D. Hackathorn. *Web Farming for the Data Warehouse: Exploiting Business Intelligence and Knowledge Management*. Morgan Kaufman Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, 1999.
- Alon Halevy, Oren Etzioni, AnHai Doan, Zachary Ives, Jayant Madhavan, Luke McDowell, and Igor Tatarinov. Crossing the structure chasm. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR-2003)*, Asilomar, CA, USA, January 2003.
- Alon Halevy, Anand Rajaraman, and Joann Ordille. Data integration: The teenage years. In *Proceedings of 32nd International Conference on Very Large Data Bases*, pages 9–16, Seoul, Korea, September 2006. VLDB Endowment.
- Brian Hammond, Amit Sheth, and Krzysztof Kochut. Semantic enhancement engine: A modular document enhancement platform for semantic applications over heterogeneous content. In Vipul Kashyap and Leon Shklar, editors, *Real World Semantic Web Applications*, volume 92 of *Frontiers in Artificial Intelligence and Applications*, pages 29–49. IOS Press, Amsterdam, The Netherlands, 2002.
- Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufman Series in Data Management Systems. Morgan Kaufman Publishers, San Francisco, second edition, 2006.
- Lixin Han, Guihai Chen, and Li Xie. AASA: A method of automatically acquiring semantic annotations. *Journal of Information Science*, 33(4):435–450, August 2007.
- David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. The MIT Press, Cambridge (Massachusetts), London, 2001.

- Siegfried Handschuh and Steffen Staab, editors. *Annotation for the Semantic Web*, volume 96 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, Berlin, 2003a.
- Siegfried Handschuh and Steffen Staab. CREAM: CREATing Metadata for the Semantic Web. *Computer Networks*, 42(5):579–598, August 2003b.
- Siegfried Handschuh and Steffen Staab. Annotation for the shallow and the deep Web. In Siegfried Handschuh and Steffen Staab, editors, *Annotation for the Semantic Web*, volume 96 of *Frontiers in Artificial Intelligence and Applications*, pages 25–45. IOS Press, Amsterdam, Berlin, 2003c.
- Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-CREAM – Semi-automatic CREAtion of Metadata. In Asunci Gómez-Pérez and V. Richard Benjamins, editors, *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web: 13th International Conference, EKAW 2002, Sigüenza, Spain, October 1-4, 2002: Proceedings*, volume 2473 of *Lecture Notes in Computer Science*, pages 358–372. Springer-Verlag, Berlin, Heidelberg, 2002.
- John A. Hartigan. *Clustering Algorithms*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, London, 1975.
- Ji He, Ah-Hwee Tan, Chew-Lim Tan, and Sam-Yuan Sung. On quantitative evaluation of clustering systems. In Weili Wu, Hui Xiong, and Shashi Shakhara, editors, *Clustering and Information Retrieval*, volume 11 of *Network Theory and Applications*, pages 105–133. Kluwer Academic Publishers, Boston, Dordrecht, 2004.
- Marti Hearst. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64, March 1997.
- Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fifteenth International Conference on Computational Linguistics*, volume 1, pages 539–545, Nantes, France, August 1992. Association for Computational Linguistics.
- Marti A. Hearst. Automated discovery of WordNet relations. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 131–151. The MIT Press, Cambridge (Massachusetts), London, 1998a.
- Marti A. Hearst. Trends & controversies: Information integration. *IEEE Intelligent Systems*, 13(5):12–24, September/October 1998b.
- Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–68, Pittsburgh, PA, USA, June/July 1993. ACM Press.

- Frans C. Heeman. Granularity in structured documents. *Electronic Publishing – Origin, Dissemination and Design*, 5(3):143–155, September 1992.
- Jan Hegewald, Felix Naumann, and Melanie Weis. XStruct: Efficient schema extraction from multiple and large XML documents. In *Proceedings of the 22nd International Conference on Data Engineering Workshops*, page 81, Atlanta, GA, USA, April 2006. IEEE Press.
- Sven Helmer. Measuring the structural similarity of semistructured documents using entropy. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 1022–1032, Vienna, Austria, September 2007. VLDB Endowment.
- Jan P. Herring. Building a business intelligence system. *The Journal of Business Strategy*, 9(3):4–9, May/June 1988.
- Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, March 2004.
- Jerry R. Hobbs, Douglas E. Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In Emmanuel Roche and Yves Schabes, editors, *Finite State Devices for Natural Language Processing*, pages 383–406. The MIT Press, 1996.
- Andreas Hotho, Alexander Maedche, Steffen Staab, and Valentin Zacharias. On knowledgeable unsupervised text mining. In Jürgen Franke, Gholamreza Nakhaeizadeh, and Ingrid Renz, editors, *Text Mining: Theoretical Aspects and Applications*, Advances in Soft Computing, pages 131–152. Physica-Verlag, Heidelberg, 2003a.
- Andreas Hotho, Steffen Staab, and Gerd Stumme. Explaining text clustering results using semantic structures. In N. Lavrac, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings*, volume 2838 of *Lecture Notes in Computer Science*, pages 217–228. Springer-Verlag, Berlin, Heidelberg, 2003b.
- Andrea L. Houston, Hsinchun Chen, Bruce R. Schatz, Susan M. Hubbard, Robin R. Sewell, and Tobun D. Ng. Exploring the use of concept spaces to improve medical information retrieval. *Decision Support Systems*, 30(2):171–186, December 2000.
- Yifen Huang and Tom M. Mitchell. Text clustering with extended user feedback. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 413–420, Seattle, WA, USA, August 2006. ACM Press.

- Scott B. Huffman and Catherine Baudin. Toward structured retrieval in semi-structured information spaces. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 751–756, Nagoya, Japan, August 1997. AAAI Press.
- Mark Hurst. *Bit Literacy: Productivity in the Age of Information and E-mail Overload*. Good Experience Press, New York, 2007.
- W. J. Hutchins. The concept of ‘aboutness’ in subject indexing. In Karen Sparck Jones and Peter Willet, editors, *Readings in Information Retrieval*, pages 93–97. Morgan Kaufmann Publishers, San Francisco, 1997. Reprinted by permission.
- Nancy M. Ide and C. M. Sperberg-McQueen. The TEI: History, goals, and future. In Nancy Ide and Jean Véronis, editors, *Text Encoding Initiative: Background and Context*, pages 5–15. Kluwer Academic Publishers, Dordrecht, Boston, 1995. Reprinted from *Computers and the Humanities*, Volume 29, Nos. 1, 2 & 3 (1995), edited by Glyn Holmes (With the addition of an SGML/TEI Bibliography).
- ISO 2788. *Documentation – Guidelines for the establishment of monolingual thesauri*. International Organization for Standardization (ISO), second edition, 1986. International Standard ISO 2788.
- ISO 5963. *Documentation – Methods for examining documents, determining their subjects, and selecting indexing terms*. International Organization for Standardization (ISO), 1985. International Standard ISO 5963.
- ISO 8879. *Information processing – Text and office systems – Standard Generalized Markup Language (SGML)*. International Organization for Standardization (ISO), 1986. International Standard ISO 8879.
- Peter Jackson and Isabelle Moulinier. *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization*, volume 5 of *Natural Language Processing*. John Benjamins Publishing Company, Amsterdam, Philadelphia, 2002.
- A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.
- Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice Hall Advanced Reference Series. Prentice Hall, Englewood Cliffs, 1988.
- Anil K. Jain, Alexander Topchy, Martin H. C. Law, and Joachim M. Buhmann. Landscape of clustering algorithms. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR’04)*, volume 1, pages 260–263, Cambridge, UK, August 2004. IEEE Press.

- R. A. Jarvis and Edward A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, C-22(11):1025–1034, November 1973.
- Anant D. Jhingran, Nelson Nelson Mattos, and Hamid Pirahesh. Information integration: A research agenda. *IBM Systems Journal*, 41(4):555–562, 2002.
- Jing Jiang and ChengXiang Zhai. Extraction of coherent relevant passages using hidden Markov models. *ACM Transactions on Information Systems (TOIS)*, 24(3):295–319, July 2006.
- William P. Jones and George W. Furnas. Pictures of relevance: A geometric analysis of similarity measures. *Journal of the American Society for Information Science*, 38(6):420–442, November 1987.
- José Kahan, Marja-Riitta Koivunen, Eric Prud’Hommeaux, and Ralph R. Swick. Annotea: An open RDF infrastructure for shared Web annotations. *Computer Networks*, 39(5):589–608, August 2002.
- Larry Kahaner. *Competitive Intelligence: How to Gather, Analyze, and Use Information to Move your Business to the Top*. Touchstone, New York, 1997.
- Jaap Kamps, Maarten Marx, Maarten de Rijke, and Börkur Sigurbjörnsson. Articulating information needs in XML query languages. *ACM Transactions on Information Systems*, 24(4):407–436, October 2006.
- George Karypis and Eui-Hong (Sam) Han. Fast supervised dimensionality reduction algorithm with applications to document categorization & retrieval. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, pages 12–19, McLean, VA, USA, November 2000a. ACM Press.
- George Karypis and Eui-Hong (Sam) Han. Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval & categorization. Technical Report TR-00-0016, University of Minnesota, Department of Computer Science, 2000b.
- Vipul Kashyap and Amit Sheth. *Information Brokering across Heterogeneous Digital Data: A Metadata-based Approach*, volume 20 of *Kluwer International Series on Advances in Database Systems*. Kluwer Academic Publishers, Boston, Dordrecht, 2000.
- Samuel Kaski, Timo Honkela, Krista Lagus, and Teuvo Kohonen. WEBSOM – self-organizing maps of document collections. *Neurocomputing*, 21(1–3):101–117, November 1998.
- Gjergji Kasneci, Fabian M. Suchanek, Maya Ramanath, and Gerhard Weikum. How NAGA uncoils: Searching with entities and relations. In *Proceedings of the 16th International Conference on World Wide Web*, pages 1167–1168, Banff, Alberta, Canada, 2007. ACM Press.

- Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, Chichester, 1990.
- Yong Seog Kim, W. Nick Street, and Filippo Menczer. Feature selection in unsupervised learning via evolutionary search. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, Boston, MA, USA, August 2000. ACM Press.
- Atanas Kiryakov, Borislav Popov, Damyan Ognyanoff, Dimitar Manov, Angel Kirilov, and Mirosлав Goranov. Semantic annotation, indexing, and retrieval. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *The Semantic Web - ISWC 2003: Second International Semantic Web Conference, Sanibel Island, FL, USA, October 20-23, 2003: Proceedings*, volume 2870 of *Lecture Notes in Computer Science*, pages 484–499. Springer-Verlag, Berlin, Heidelberg, 2003.
- Paul Kogut and Jeff Heflin. Semantic Web technologies for aerospace. In *Proceedings of the 2003 IEEE Aerospace Conference*, volume 6, pages 2887–2894, Big Sky, MT, USA, March 2003. IEEE Press.
- Paul Kogut and William Holmes. AeroDAML: Applying information extraction to generate DAML annotations from Web pages. In *Proceedings of the K-CAP 2001 Workshop on Knowledge Markup and Semantic Annotation*, pages 111–113, Victoria, BC, Canada, October 2001.
- Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer-Verlag, Berlin, Heidelberg, third edition, 2001.
- Teuvo Kohonen. Things you haven't heard about the self-organizing map. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 3, pages 1147–1156, San Francisco, CA, USA, March/April 1993. IEEE Service Center.
- Robert R. Korfhage. *Information Storage and Retrieval*. John Wiley & Sons, New York, Chichester, 1997.
- Hans-Peter Kriegel, Karsten M. Borgwardt, Peer Kröger, Alexey Pryakhin, Matthias Schubert, and Arthur Zimek. Future trends in data mining. *Data Mining and Knowledge Discovery*, 15(1):87–97, August 2007.
- Klaus Krippendorff. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, Thousand Oaks (California), second edition, 2004.
- Anagha Kulkarni and Ted Pedersen. SenseClusters: Unsupervised clustering and labeling of similar contexts. In *Proceedings of the Demonstration and Interactive Poster Session of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, MI, USA, June 2005. Association for Computational Linguistics.

- Manuela Kunze. *Linguistische Analysen für die semantische Auszeichnung natürlicher sprachlicher Dokumente*. Verlag Dr. Hut, München, 2006. In German.
- Krista Lagus and Samuel Kaski. Keyword selection method for characterizing text document maps. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN '99)*, volume 1, pages 371–376, Edinburgh, UK, September 1999. The Institution of Electrical Engineers.
- Krista Lagus, Samuel Kaski, and Teuvo Kohonen. Mining massive document collections by the WEBSOM method. *Information Sciences*, 163(1–3):135–156, June 2004.
- Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 16–22, San Diego, CA, USA, August 1999. ACM Press.
- Stephen Laurence and Eric Margolis. Concepts and cognitive science. In Eric Margolis and Stephen Laurence, editors, *Concepts: Core Readings*, pages 3–81. The MIT Press, Cambridge (Massachusetts), London, 1999.
- Claudia Leacock and Martin Chodorow. Combining local context and WordNet similarity for word sense identification. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 265–283. The MIT Press, Cambridge (Massachusetts), London, 1998.
- Wei-Po Lee and Cheng-Che Lu. Customising WAP-based information services on mobile networks. *Personal and Ubiquitous Computing*, 7(6):321–330, December 2003.
- David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5: 361–397, April 2004.
- Hang Li and Kenji Yamanishi. Topic analysis using a finite mixture model. *Information Processing & Management*, 39(4):521–541, July 2003.
- Jianming Li, Lei Zhang, and Yong Yu. Learning to generate semantic annotation for domain specific sentences. In *Proceedings of the K-CAP 2001 Workshop on Knowledge Markup and Semantic Annotation*, pages 89–96, Victoria, BC, Canada, October 2001.
- Tok Wand Ling, Mong Li Lee, and Gillian Dobbie. *Semistructured Database Design*, volume 1 of *Web Information Systems Engineering and Internet Technologie*. Springer Science+Business Media, New York, 2005.
- Tao Liu, Shengping Liu, Zheng Chen, and Wei-Ying Ma. An evaluation on feature selection for text clustering. In *Machine Learning: Proceedings of the Twentieth International Conference (ICML 2003)*, pages 488–495, Washington, DC, USA, August 2003. AAAI Press.

- Jacques Loeckx, Hans-Dieter Ehrich, and Markus Wolf. *Specification of Abstract Data Types*. Wiley-Teubner, Chichester, New York, 1996.
- Stanley Loh, Leandro Krug Wives, and José Palazzo M. de Oliveira. Concept-based knowledge discovery in texts extracted from the Web. *ACM SIGKDD Explorations Newsletter*, 2(1):29–39, 2000.
- Robert M. Losee. Minimizing information overload: The ranking of electronic messages. *Journal of Information Science*, 15(3):179–189, August 1989.
- Peter Lyman and Hal R. Varian. How much information. Retrieved from <http://www.sims.berkeley.edu/how-much-info-2003> on 2008-08-01, 2003.
- J. MacQueen. Some methods for classification and analysis of multivariate observations. In Lucien M. Le Cam and Jerzy Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume I: Statistics, pages 281–297. University of California Press, Berkeley, Los Angeles, 1967.
- Sofus A. Macskassy, Arunava Banerjee, Brian D. Davison, and Haym Hirsh. Human performance on clustering web pages: A preliminary study. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 264–268, New York, NY, USA, August 1998. AAAI Press.
- Alexander Maedche. *Ontology learning for the Semantic Web*, volume 665 of *Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Dordrecht, 2002.
- Alexander Maedche and Steffen Staab. Mining ontologies from text. In Rose Dieng and Olivier Corby, editors, *Knowledge Engineering and Knowledge Management: Methods, Models, and Tools: 12th International Conference, EKAW 2000, Juan-les-Pins, France, October 2-6, 2000: Proceedings*, volume 1937 of *Lecture Notes in Computer Science*, pages 189–202. Springer-Verlag, Berlin, Heidelberg, 2000.
- Alexander Maedche and Steffen Staab. KAON - The Karlsruhe ontology and Semantic Web meta project. *KI - Künstliche Intelligenz*, 17(3):27–29, July 2003.
- Alexander Maedche, Steffen Staab, Nenad Stojanovic, Rudi Studer, and York Sure. SE-mantic portAL: The SEAL Approach. In Dieter Fensel, James Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, pages 317–359. The MIT Press, Cambridge (Massachusetts), London, 2003.
- Jens-Erik Mai. Semiotics and indexing: An analysis of the subject indexing process. *Journal of Documentation*, 57(5):591–622, September 2001.

- Eve Maler and Jeanne El Andaloussi. *Developing SGML DTDs: From Text to Model to Markup*. Prentice Hall PTR, Upper Saddle River (New Jersey), 1996.
- Heikki Mannila. Methods and problems in data mining. In Foto N. Afrati and Phokion G. Kolaitis, editors, *Database Theory — ICDT '97: 6th International Conference, Delphi, Greece, January 8–10, 1997: Proceedings*, volume 1186 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, Berlin, Heidelberg, 1997.
- Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge (Massachusetts), London, 1999.
- Jianchang Mao, Sumit Taank, Christina Chung, and Alpha Luk. Method and system for naming a cluster of words and phrases. United States Patent No. 20030177000 filed through Verity, Inc., September 2003a.
- Song Mao, Azriel Rosenfeld, and Tapas Kanungo. Document structure analysis algorithms: A literature survey. In *Proceedings of the 15th Annual Symposium on Electronic Imaging: Science and Technology*, volume 5010, pages 197–207, Santa Clara, CA, USA, January 2003b. The International Society for Optical Engineering and The Society for Imaging Science and Technology.
- Salvatore T. March and Gerald F. Smith. Design and natural science research on information technology. *Decision Support Systems*, 15(4):251–266, December 1995.
- Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. HT06, tagging paper, taxonomy, Flickr, academic article, to read. In *Proceedings of the Seventeenth Conference on Hypertext and Hypermedia*, pages 31–40, Odense, Denmark, August 2006. ACM Press.
- Laura A. Mather and Jarrod Note. Discovering encyclopedic structure and topics in text. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD-2000) Workshop on Text Mining*, pages 45–50, Boston, MA, USA, August 2000.
- Michael L. Mauldin. *Conceptual Information Retrieval: A Case Study in Adaptive Partial Parsing*, volume 152 of *Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.
- Diana Maynard, Valentin Tablan, Christan Ursu, Hamish Cunningham, and Yorick Wilks. Named entity recognition from diverse text types. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP-2001)*, Tzigov Chark, Bulgaria, September 2001.
- Dave McComb. *Semantics in Business Systems: The Savvy Manager's Guide*. Morgan Kaufman Publishers, Amsterdam, Boston, 2004.

- Luke McDowell, Oren Etzioni, Steven D. Gribble, Alon Halevy, Henry Levy, William Pentney, Deepak Verma, and Stani Vlasheva. Mangrove: Enticing ordinary people onto the Semantic Web via instant gratification. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *The Semantic Web - ISWC 2003: Second International Semantic Web Conference, Sanibel Island, FL, USA, October 20-23, 2003: Proceedings*, volume 2870 of *Lecture Notes in Computer Science*, pages 754 – 770. Springer-Verlag, Berlin, Heidelberg, 2003.
- Deborah L. McGuinness. Ontologies come of age. In Dieter Fensel, James Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, pages 171–194. The MIT Press, Cambridge (Massachusetts), London, 2003.
- Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 490–499. ACM Press, August 2007.
- Jim Melton and Stephen Buxton. *Querying XML: XQuery, XPath, and SQL/XML in Context*. Morgan Kaufman Series in Data Management Systems. Morgan Kaufman Publishers, San Francisco, 2006.
- Andrei Mikheev. Feature lattices for maximum entropy modelling. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 848–854, Montréal, Canada, August 1998. Morgan Kaufmann.
- Glenn W. Milligan and Martha C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, June 1985.
- Jun-Ki Min, Jae-Yong Ahn, and Chin-Wan Chung. Efficient extraction of schemas for XML documents. *Information Processing Letters*, 85(1):7–12, January 2003.
- Michele Missikoff, Roberto Navigli, and Paola Velardi. Integrated approach to Web ontology learning and engineering. *IEEE Computer*, 35(11):60–63, November 2002a.
- Michele Missikoff, Roberto Navigli, and Paola Velardi. The usable ontology: An environment for building and assessing a domain ontology. In Ian Horrocks and James A. Hendler, editors, *The Semantic Web - ISWC 2002: First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002: Proceedings*, volume 2342 of *Lecture Notes in Computer Science*, pages 39–53. Springer-Verlag, Berlin, Heidelberg, 2002b.
- Marie-Francine Moens. *Automatic Indexing and Abstracting of Document Texts*. Kluwer International Series on Information Retrieval. Kluwer Academic Publishers, Boston, Dordrecht, 2000.
- Marie-Francine Moens. *Information Extraction: Algorithms and Prospects in a Retrieval Context*. The Information Retrieval Series. Springer, Dordrecht, 2006.

- Chuang-Hue Moh, Ee-Peng Lim, and Wee-Keong Ng. Re-engineering structures from Web documents. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 67–76, San Antonio, Texas, United States, June 2000a. ACM Press.
- Chuang-Hue Moh, Ee-Peng Lim, and Wee-Keong Ng. DTD-Miner: A tool for mining DTD from XML documents. In *Proceedings of the Second International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS 2000)*, pages 144–151, Milpitas, CA, USA, June 2000b. IEEE Press.
- G. William Moore and Jules J. Berman. Medical data mining and knowledge discovery. In *Anatomic Pathology Data Mining*, volume 60 of *Studies in Fuzziness and Soft Computing*, pages 72–117, Heidelberg, New York, 2001. Physica-Verlag.
- Masaki Mori, Takao Miura, and Isamu Shioya. Topic detection and tracking for news Web pages. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 338–342, Hong Kong, December 2006. IEEE Computer Society.
- Enrico Motta, Enrico Buckingham Shum, and John Domingue. Ontology-diven document enrichment: Principles, tools and applications. *International Journal of Human-Computer Studies*, 52(6):1071–1109, June 2000.
- John Naisbitt. *Megatrends: Ten New Directions Transforming Our Lives*. Warner Books, New York, 1982.
- Tetsuya Nasukawa and Tohru Nagano. Text analysis and knowledge mining system. *IBM Systems Journal*, 40(4):967–984, 2001.
- Gonzalo Navarro and Ricardo Baeza-Yates. Proximal nodes: A model to query document databases by content and structure. *ACM Transactions on Information Systems (TOIS)*, 15(4):400–435, October 1997.
- Gonzalo Navarro, Ricardo Baeza-Yates, and João Marcelo Azevedo Arcoverde. Match-simile: A flexible approximate matching tool for searching proper names. *Journal of the American Society for Information Science and Technology*, 54(1):3–15, January 2003.
- Svetlozar Nestorov, Serge Abiteboul, and Rajeev Motwani. Extracting schema from semistructured data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 295–306, Seattle, WA, USA, June 1998. ACM Press.
- Günter Neumann and Jakub Piskorski. A shallow text processing core engine. *Computational Intelligence*, 18(3):451–476, August 2002.
- Günter Neumann, Rolf Backofen, Judith Baur, Markus Becker, and Christian Braun. An information extraction core system for real world German text processing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 209–216, Washington, DC, USA, March/April 1997. Morgan Kaufmann Publishers.

- David Newman, Kat Hagedorn, Chaitanya Chemudugunta, and Padhraic Smyth. Subject metadata enrichment using statistical topic models. In *Proceedings of the 2007 Conference on Digital Libraries*, pages 366–375, Vancouver, BC, Canada, June 2007. ACM Press.
- Jay F. Nunamaker, Jr., Minder Chen, and Titus D. M. Purdin. Systems development in information systems research. *Journal of Management Information Systems*, 7(3): 89–106, Winter 1991.
- Charles K. Ogden and Ivor A. Richards. *The Meaning of Meaning: A Study of the Influence of Language upon Thought and the Science of Symbolism*. C. K. Ogden and Linguistics. Routledge/Thoemmes Press, London, 1994. Edited and introduced by W. Terrence Gordon.
- James Osborn and Leon Sterling. JUSTICE: A judicial search tool using intelligent concept extraction. In *Proceedings of the Seventh International Conference on Artificial Intelligence and Law*, pages 173–181, Oslo, Norway, June 1999. ACM Press.
- Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 251–260, Taipei, Taiwan, March 1995. IEEE Press.
- Maria Teresa Pazienza, editor. *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology: International Summer School, SCIE-97, Frascati, Italy, July 14–18, 1997*, volume 1299 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, 1997.
- Martin Peltzer, Jonathan J. Doyle, and Elizabeth A. Voight. *German Commercial Code: German-English Text with an Introduction in English*. Verlag Dr. Otto Schmidt, Köln, fourth revised edition, 2000.
- Martin Peltzer and Anthony G. Hickinbotham. *German Stock Corporation Act and Co-Determination Act: German-English Text with an Introduction in English*. Verlag Dr. Otto Schmidt, Köln, 1999.
- Martin Peltzer, Jermyn P. Brooks, and Terry Hopcroft. *German Law Pertaining to Companies with Limited Liability: German-English Text with an Introduction in English*. Verlag Dr. Otto Schmidt, Köln, 3rd revised edition, 1996.
- Wendell Piez. Beyond the “descriptive vs. procedural” distinction. *Markup Languages: Theory and Practice*, 3(2):141–172, April 2001.
- Antonella Poggi and Serge Abiteboul. XML data integration with identification. In Gavin Bierman and Christoph Koch, editors, *Database Programming Languages: 10th International Symposium, DBPL 2005, Trondheim, Norway, August 28-29, 2005, Revised*

- 
- Selected Papers*, volume 3774 of *Lecture Notes in Computer Science*, pages 106–121. Springer-Verlag, Berlin, Heidelberg, 2005.
- Aurora Pons-Porrata, Rafael Berlanga-Llavori, and José Ruiz-Shulcloper. Topic discovery based on text mining techniques. *Information Processing & Management*, 43(3):752–768, May 2007.
- Jay M. Ponte and W. Bruce Croft. Text segmentation by topic. In *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, pages 113–125, Pisa, Italy, September 1997. Springer-Verlag.
- Alexandrin Popescul and Lyle H. Ungar. Automatic labeling of document clusters. Department of Computer and Information Science, University of Pennsylvania, unpublished paper available from <http://www.cis.upenn.edu/~popescul/Publications/popescul00labeling.pdf>, accessed 2008-08-01, 2000.
- Borislav Popov, Atanas Kiryakov, Angel Kirilov, Dimitar Manov, Damyan Ognyanoff, and Miroslav Goranov. KIM - semantic annotation platform. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *The Semantic Web - ISWC 2003: Second International Semantic Web Conference, Sanibel Island, FL, USA, October 20-23, 2003: Proceedings*, volume 2870 of *Lecture Notes in Computer Science*, pages 834–849. Springer-Verlag, Berlin, Heidelberg, 2003.
- Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, and Angel Kirilov. KIM – A semantic platform for information extraction and retrieval. *Natural Language Engineering*, 10(3–4):375–392, September 2004.
- M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980a.
- Michael E. Porter. *Competitive Strategy: Techniques for Analyzing Industries and Competitors*. The Free Press, New York, London, 1980b.
- Eddie Rasmussen. Clustering algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 419–442. Prentice Hall PTR, Upper Saddle River (New Jersey), 1992.
- Lisa F. Rau. Conceptual information extraction and information retrieval from natural language input. In *Proceedings of RIAO-88: Conference on User-Oriented, Content-Based, Text and Image Handling*, pages 424–437, Cambridge, Ma, USA, March 1988.
- Andreas Rauber. LabelSOM: On the labeling of self-organizing maps. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'99), Washington, DC, July 10–16 1999*, pages 3527–3532. IEEE Press, 1999.
- Andreas Rauber and Dieter Merkl. Text mining in the SOMLib digital library system: The representation of topics and genres. *Applied Intelligence*, 18(3):271–293, May/June 2003.

- Lawrence Reeve and Hyoil Han. Survey of semantic annotation platforms. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, pages 1634–1638, Santa Fe, NM, USA, March 2005. ACM Press.
- Allen Renear. The descriptive/procedural distinction is flawed. *Markup Languages: Theory and Practice*, 2(4):411–420, Fall 2000.
- Allen Renear, David Dubin, and C. M. Sperberg-McQueen. Towards a semantics for XML markup. In *Proceedings of the 2002 ACM Symposium on Document Engineering*, pages 119–126. ACM Press, November 2002.
- Ingrid Renz and Jürgen Franke. Text mining. In Jürgen Franke, Gholamreza Nakhaeizadeh, and Ingrid Renz, editors, *Text Mining: Theoretical Aspects and Applications*, Advances in Soft Computing, pages 1–19. Physica-Verlag, Heidelberg, 2003.
- Reuters Business Information, editor. *Dying for Information? An Investigation into the Effects of Information Overload in the UK and Worldwide*. Reuters, London, 1996.
- Jeffrey C. Reynar. Statistical models for topic segmentation. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 357–364, College Park, MD, USA, June 1999. Association for Computational Linguistics.
- Tony G. Rose, Mark Stevenson, and Miles Whitehead. The Reuters Corpus Volume 1 - from yesterday's news to tomorrow's language resources. In *Proceedings of Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 827–833, Las Palmas, Canary Islands, Spain, May 2002. European Language Resources Association.
- Dietmar Rösner and Manuela Kunze. The XDOC Document Suite – a workbench for document mining. In Jürgen Franke, Gholamreza Nakhaeizadeh, and Ingrid Renz, editors, *Text Mining: Theoretical Aspects and Applications*, Advances in Soft Computing, pages 113–130. Physica-Verlag, Heidelberg, 2003.
- Dietmar Rösner, Manuela Kunze, and Sylke Kröttsch. Transforming and enriching documents for the Semantic Web. *KI - Künstliche Intelligenz*, 18(1):24–29, February 2004.
- Philip Russom. BI search and text analytics: New additions to the BI technology stack. Second Quarter 2007 TDWI Best Practices Report. Retrieved from <http://www.tdwi.org/Research/ReportSeries/index.aspx> on 2008-08-01, 2007.
- Giovanni M. Sacco. Dynamic taxonomies: A model for large information bases. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):468–479, May/June 2000.
- John I. Saeed. *Semantics*. Introducing Linguistics. Blackwell Publishing, Malden (Massachusetts), Oxford (United Kingdom), second edition, 2003.

- Naomi Sager. Sublanguage: Linguistic phenomenon, computational tool. In Ralph Grishman and Richard Kittredge, editors, *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, pages 1–17. Lawrence Erlbaum Associates, Publishers, Hillsdale, London, 1986.
- Gerard Salton. *Automatic Information Organization and Retrieval*. McGraw-Hill Computer Science Series. McGraw-Hill, New York, St. Louis, 1968.
- Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Company, Reading (Massachusetts), Menlo Park (California), 1989.
- Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, August 1988.
- Gerard Salton and Michael E. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36, January 1968.
- Gerard Salton, J. Allan, and Chris Buckley. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58. ACM Press, June/July 1993.
- Gerard Salton, James Allan, Chris Buckley, and Amit Singhal. Automatic analysis, theme generation, and summarization of machine-readable texts. In Maristella Agosti and Alan F. Smeaton, editors, *Information Retrieval and Hypertext*, pages 51–73. Kluwer Academic Publishers, Boston, London, 1996.
- Mark Sanderson and Dawn Lawrie. Building, testing, and applying concept hierarchies. In W. Bruce Croft, editor, *Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval*, volume 7 of *Kluwer International Series on Information Retrieval*, pages 235–266. Kluwer Academic Publishers, Boston, Dordrecht, 2000.
- Sergio M. Savaresi, Daniel L. Boley, Sergio Bittanti, and Giovanna Gazzaniga. Cluster selection in divisive clustering algorithms. In *Proceedings of the Second SIAM International Conference on Data Mining*, pages 299–314, Arlington, VA, USA, April 2002. Society for Industrial and Applied Mathematics.
- Allen G. Schick, Lawrence A. Gordon, and Susan Haka. Information overload: A temporal approach. *Accounting, Organizations and Society*, 15(3):199–220, Spring 1990.
- Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK, September 1994. Centre for Computational Linguistics, Institute of Science and Technology, University of Manchester.

- Helmut Schmid. Improvements in part-of-speech tagging with an application to German. In Susan Armstrong, Kenneth Church, Pierre Isabelle, Sandra Manzi, Evelyn Tzoukermann, and David Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*, volume 11 of *Text, Speech and Language Technology*, pages 13–25, Dordrecht, Boston, 1999. Kluwer Academic Publishers.
- Anja Schulz, Myra Spiliopoulou, and Karsten Winkler. Kursrelevanzprognose von Ad-hoc-Meldungen: Text Mining wider die Informationsüberlastung im Mobile Banking. In Wolfgang Uhr, Werner Esswein, and Eric Schoop, editors, *Wirtschaftsinformatik 2003: Medien, Märkte, Mobilität*, volume II, pages 181–200. Physika-Verlag, 2003. In German.
- Erich Schweighofer, Andreas Rauber, and Michael Dittenbach. Automatic text representation, classification and labeling in European law. In *Proceedings of the 8th International Conference on Artificial Intelligence and Law*, pages 78–87, St. Louis, MO, USA, 2001. ACM Press.
- Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.
- Fabrizio Sebastiani. A tutorial on automated text categorisation. In Analia Amandi and Ricardo Zunino, editors, *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence*, pages 7–35, Buenos Aires, Argentina, 1999.
- Satoshi Sekine. On-demand information extraction. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, pages 731–738, Sydney, Australia, July 2006. Association for Computational Linguistics.
- Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. Extended named entity hierarchy. In *Proceedings of Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1818–1824, Las Palmas, Canary Islands, Spain, May 2002. European Language Resources Association.
- Arijit Sengupta. Database concepts for marked-up textual documents. In Amita Goyal Chin, editor, *Text Databases and Document Management: Theory and Practice*, pages 118–159. Idea Group Publishing, Hershey, PA, USA, July 2000.
- Arijit Sengupta and Sandeep Puro. Transitioning existing content: Inferring organization specific documents. *Australian Journal of Information Systems*, 8(1):91–99, September 2000.
- Keith E. Shafer. Generation and reduction of an SGML defined grammer. United States Patent 5,583,762, December 1996.

- Amit Sheth. Data semantics: What, where and how? In Robert Meersman and Leo Mark, editors, *Database Applications Semantics, Proceedings of the Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6), Stone Mountain, Atlanta, Georgia, USA, May 30 - June 2, 1995*, volume 74 of *IFIP Conference Proceedings*, pages 601–610. Chapman & Hall, London, 1996.
- Amit Sheth, Clemens Bertram, David Avant, Brian Hammond, Krzysztof Kochut, and Yashodhan Warke. Managing semantic content for the Web. *IEEE Internet Computing*, 6(4):80–87, July/August 2002.
- Avi Silberschatz and Alexander Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):970–974, December 1996.
- Andrew E. Smith. Machine mapping of document collections: The Leximancer system. In *Proceedings of the Fifth Australasian Document Computing Symposium (ADCS)*, Sunshine Coast, Australia, December 2000. Distributed Systems Technology Centre.
- Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 801–808, Sydney, Australia, July 2006. Association for Computational Linguistics.
- Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, February 1999.
- Amit Somani, David Choy, and James C. Kleewein. Bringing together content and data management systems: Challenges and opportunities. *IBM Systems Journal*, 41(4):686–696, 2002.
- John F. Sowa, editor. *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufmann Series in Representation and Reasoning. Morgan Kaufmann Publishers, San Mateo, 1991.
- C. M. Sperberg-McQueen and Lou Burnard, editors. *Guidelines for Electronic Text Encoding and Interchange*. Published for the TEI Consortium by the Humanities Computing Unit, University of Oxford, Oxford, p4 edition, 2002.
- C. M. Sperberg-McQueen and Lou Burnard. The design of the TEI encoding scheme. In Nancy Ide and Jean Véronis, editors, *Text Encoding Initiative: Background and Context*, pages 17–39. Kluwer Academic Publishers, Dordrecht, Boston, 1995. Reprinted from *Computers and the Humanities*, Volume 29, Nos. 1, 2 & 3 (1995), edited by Glyn Holmes (With the addition of an SGML/TEI Bibliography).

- Myra Spiliopoulou, Irene Ntoutsi, Yannis Theodoridis, and Rene Schult. MONIC: modeling and monitoring cluster transitions. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 706–711. ACM Press, August 2006.
- Steffen Staab. Emergent semantics. *IEEE Intelligent Systems*, 17(1):78–86, January/February 2002.
- Benno Stein and Sven Meyer zu Eissen. Automatic document categorization: Interpreting the performance of clustering algorithms. In Andreas Günter, Rudolf Kruse, and Bernd Neumann, editors, *KI 2003: Advances in Artificial Intelligence: 26th Annual German Conference on AI, KI 2003 Hamburg, Germany, September 15-18, 2003: Proceedings*, volume 2821 of *Lecture Notes on Computer Science*, pages 254–266. Springer-Verlag, Berlin, Heidelberg, 2003.
- Benno Stein and Sven Meyer zu Eissen. Topic identification: Framework and application. In *Proceedings of the 4th International Conference on Knowledge Management (I-KNOW 04)*, pages 353–360, Graz, Austria, June/July 2004. Journal of Universal Computer Science.
- Benno Stein, Sven Meyer zu Eissen, and Frank Wißbrock. On cluster validity and the information need of users. In *Proceedings of the Third IASTED International Conference on Artificial Intelligence and Applications*, pages 216–221, Benalmádena, Spain, September 2003. ACTA Press.
- Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *Workshop on Text Mining at the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–110, Boston, MA, USA, August 2000.
- Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on Web-page clustering. In *Proceedings of Seventeenth National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64, Austin, TX, USA, July 2000. AAAI Press.
- Heiner Stuckenschmidt and Frank van Harmelen, editors. *Information Sharing on the Semantic Web*. Advanced Information and Knowledge Processing. Springer-Verlag, Berlin, Heidelberg, 2005.
- Fabian M. Suchanek, Georgiana Ifrim, and Gerhard Weikum. Combining linguistic and statistical analysis to extract relations from Web documents. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 712–717. ACM Press, August 2006.

- Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A core of semantic knowledge unifying WordNet and Wikipedia. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706, Banff, Alberta, Canada, 2007. ACM Press.
- Dan Sullivan. *Document Warehousing and Text Mining: Techniques for Improving Business Operations, Marketing, and Sales*. John Wiley & Sons, New York, Chichester, 2001.
- Marcelo Tallis. Semantic word processing for content authors. In *Proceedings of the K-CAP 2003 Workshop on Knowledge Markup and Semantic Annotation (SEMANNOT 2003)*, Sanibel, FL, USA, October 2003.
- Marcelo Tallis, Neil M. Goldman, and Robert M. Balzer. The Briefing Associate: A role for COTS applications in the Semantic Web. In Isabel F. Cruz, Stefan Decker, Jérôme Euzenat, and Deborah L. McGuinness, editors, *Proceedings of SWWS'01, The First Semantic Web Working Symposium*, pages 463–475, Stanford, CA, USA, July/August 2001.
- Marcelo Tallis, Neil M. Goldman, and Robert M. Balzer. The Briefing Associate: Easing authors into the Semantic Web. *IEEE Intelligent Systems*, 17(1):26–32, January/February 2002.
- Naiyana Tansalarak and Kajal T. Claypool. QMatch - Using paths to match XML schemas. *Data & Knowledge Engineering*, 60(2):260–282, February 2007.
- Paul Thompson and Christopher C. Dozier. Name recognition and retrieval performance. In Tomek Strzalkowski, editor, *Natural Language Information Retrieval*, volume 7 of *Text, Speech and Language Technology*, pages 261–272. Kluwer Academic Publishers, Boston, Dordrecht, 1999.
- Frank Wm. Tompa. What is (tagged) text? In *Dictionaries in the Electronic Age: Proceedings of the Fifth Conference of University of Waterloo Centre for the New OED*, pages 81–93, Oxford, UK, September 1989.
- Pucktada Treeratpituk and Jamie Callan. Automatically labeling hierarchical clusters. In *Proceedings of the 2006 International Conference on Digital Government Research*, pages 167–176, San Diego, CA, USA, May 2006. ACM Press.
- Andrew Trotman and Börkur Sigurbjörnsson. Narrowed extended XPath I (NEXI). In Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltán Szlávik, editors, *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004*, volume 3493 of *Lecture Notes in Computer Science*, pages 16–40. Springer-Verlag, Berlin, Heidelberg, 2005.

- Frank S. C. Tseng and Annie Y. H. Chou. The concept of document warehousing for multi-dimensional modeling of textual-based business intelligence. *Decision Support Systems*, 42(2):727–744, November 2006.
- Jordi Turmo, Alicia Ageno, and Neus Català. Adaptive information extraction. *ACM Computing Surveys*, 38(2):4, July 2006.
- Victoria S. Uren, Philipp Cimiano, José Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(1):14–28, January 2006.
- Michael Uschold. Where are the semantics in the semantic web? *AI Magazine*, 24(3): 25–36, September 2003.
- Mike Uschold and Michael Gruninger. Ontologies: Principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, June 1996.
- Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt, and Fabio Ciravegna. MnM: Ontology driven semi-automatic and automatic support for semantic markup. In Asunci Gómez-Pérez and V. Richard Benjamins, editors, *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web: 13th International Conference, EKAW 2002, Sigüenza, Spain, October 1-4, 2002: Proceedings*, volume 2473 of *Lecture Notes in Computer Science*, pages 379–391, Berlin, Heidelberg, 2002. Springer-Verlag.
- Michalis Vazirgiannis, Maria Halkidi, and Dimitrios Gunopulos. *Uncertainty Handling and Quality Assessment in Data Mining*. Advanced Information and Knowledge Processing. Springer-Verlag, London, Berlin, 2003.
- Richard G. Vedder, Michael T. Vanecek, C. Stephen Guynes, and James J. Cappel. CEO and CIO perspectives on competitive intelligence. *Communications of the ACM*, 42(8): 108–116, August 1999.
- Juha Vesanto and Esa Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):586–600, May 2000.
- Špela Vintar, Paul Buitelaar, Bärbel Ripplinger, Bogdan Sacaleanu, Bogdan Raileanu, and Detlef Prescher. An efficient and flexible format for linguistic and semantic annotation. In *Proceedings of Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1658–1662, Las Palmas, Canary Islands, Spain, May 2002. European Language Resources Association.
- Martin Volk and Simon Clematide. Learn – filter – apply – forget: Mixed approaches to named entity recognition. In Ana M. Moreno and Reind P. van de Riet, editors,

- Applications of Natural Language to Information Systems, NLDB'2001: International Workshop 28.–29. June 2001 in Madrid, Spain: Proceedings*, volume P-3 of *GI-Edition – Lecture Notes in Informatics*, pages 153–163, Bonn, June 2001. Köllen Druck+Verlag.
- Martin Volk, Bärbel Ripplinger, Špela Vintar, Paul Buitelaar, Diana Raileanu, and Bogdan Sacaleanu. Semantic annotation for concept-based cross-language medical information retrieval. *International Journal of Medical Informatics*, 67(1–3):97–112, December 2002.
- Ke Wang and Huiqing Liu. Discovering structural association of semistructured data. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):353–371, May/June 2000.
- Gerhard Weikum. DB&IR: Both sides now. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 25–30, Beijing, China, June 2007. ACM Press.
- Sholom M. Weiss, Nitin Indurkha, Tong Zhang, and Fred J. Damerau. *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer Science+Business Media, New York, 2005.
- Mischa Weiss-Lijn, Janet T. McDonnell, and Leslie James. An empirical evaluation of the interactive visualization of metadata to support document use. In Katy Börner and Chaomei Chen, editors, *Visual Interfaces to Digital Libraries*, volume 2539 of *Lecture Notes in Computer Science*, pages 50–64. Springer-Verlag, Berlin, Heidelberg, 2002.
- Mischa Weiss-Lijn, Janet T. McDonnell, and Leslie James. Interactive visualization of paragraph-level metadata to support corporate document use. In Vladimir Geroimenko and Chaomei Chen, editors, *Visualizing the Semantic Web: XML-based Internet and Information Visualization*, pages 116–134. Springer-Verlag, London, Berlin, 2003.
- Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Clustering user queries of a search engine. In *Proceedings of the Tenth International World Wide Web Conference on World Wide Web*, pages 162–168, Hong Kong, May 2001. ACM Press.
- Chris West. Competitive intelligence in Europe. *Business Information Review*, 16(3): 143–150, September 1999.
- Adam B. Wilcox and George Hripcsak. The role of domain knowledge in automating medical text report classification. *The Journal of the American Medical Informatics Association*, 10(4):330–338, July-August 2003.
- Karsten Winkler. Wettbewerbsinformationssysteme: Begriff, An- und Herausforderungen. Technical Report No. 59, HHL - Leipzig Graduate School of Management, Leipzig, Germany, 2003. In German.

- Karsten Winkler. Getting started with DIAsDEM Workbench 2.2: A case-based tutorial. Technical Report at the Research Group ITI/KMD, Faculty of Computer Science, Otto von Guericke University of Magdeburg, Magdeburg, Germany, 2007.
- Karsten Winkler and Myra Spiliopoulou. Structuring domain-specific text archives by deriving a probabilistic XML DTD. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Principles of Data Mining and Knowledge Discovery: 6th European Conference, PKDD 2002, Helsinki, Finland, August 19-23, 2002: Proceedings*, volume 2431 of *Lecture Notes in Computer Science*, pages 461–474. Springer-Verlag, Berlin, Heidelberg, 2002.
- Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufman Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, second edition, 2005. Weka is available at <http://www.cs.waikato.ac.nz/~ml/weka>, accessed 2008-08-01.
- William A. Woods. What’s in a link: Foundations for semantic networks. In Daniel G. Bobrow and Allan Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 35–82. Academic Press, New York, San Francisco, 1975.
- William A. Woods. Understanding subsumption and taxonomy: A framework for progress. In John F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann Series in Representation and Reasoning, pages 45–94. Morgan Kaufmann Publishers, San Mateo, 1991.
- William A. Woods. Conceptual indexing: A better way to organize knowledge. Technical Report TR-97-61, Sun Microsystems Laboratories, Mountain View, CA, USA, April 1997.
- World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Second Edition): W3C Recommendation. Retrieved from <http://www.w3.org/TR/2000/REC-xml-20001006> on 2008-08-01, October 2000.
- World Wide Web Consortium. XML Schema part 0: Primer: W3C Recommendation. Retrieved from <http://www.w3.org/TR/2001/REC-xmlschema-0-20010502> on 2008-08-01, May 2001.
- Weili Wu, Hui Xiong, and Shashi Shakhara, editors. *Clustering and Information Retrieval*, volume 11 of *Network Theory and Applications*. Kluwer Academic Publishers, Boston, Dordrecht, 2004.
- Lian Yan, Richard H. Wolniewicz, and Robert Dodier. Predicting customer behavior in telecommunications. *IEEE Intelligent Systems*, 19(2):50–58, March/April 2004.

- Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in statistical learning of text categorization. In *Proceedings on the 14th International Conference on Machine Learning*, pages 412–420, Nashville, TN, USA, July 1997. Morgan Kaufmann Publishers.
- Illhoi Yoo and Xiaohua Hu. A comprehensive comparison study of document clustering for a biomedical digital library MEDLINE. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 220–229, Chapel Hill, NC, USA, June 2006. ACM Press.
- Matthew Young-Lai and Frank W. M. Tompa. Stochastic grammatical inference of text database structure. *Machine Learning*, 40(2):111–137, August 2000.
- Hong Yu and Eugene Agichtein. Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, 19:340i–349i, Supplement 1 2003.
- Oren Zamir, Oren Etzioni, Omid Madani, and Richard M. Karp. Fast and intuitive clustering of Web documents. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 287–290, Newport Beach, CA, USA, August 1997. AAAI Press.
- Chun Zhang, Jeffrey Naughton, David DeWitt, Qiong Luo, and Guy Lohman. On supporting containment queries in relational database management systems. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data on Management of Data*, pages 425–436, Santa Barbara, CA, USA, May 2001. ACM Press.
- Shubin Zhao and Jonathan Betz. Corroborate and learn facts from the Web. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 995–1003. ACM Press, August 2007.
- Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 515–524. ACM Press, November 2002.
- Ying Zhao and George Karypis. Criterion functions for clustering on high-dimensional data. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 211–237. Springer-Verlag, Berlin, Heidelberg, 2006.
- Shi Zhong and Joydeep Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:1001–1037, November 2003.
- Lina Zhou. Ontology learning: State of the art and open issues. *Information Technology and Management*, 8(3):241–252, September 2007.



# Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Karsten Winkler  
Berlin, 10. August 2008