HHL Working Paper No. 58

Getting Started with DIAsDEM Workbench 2.0: A Case-Based Tutorial

Karsten Winkler

HHL – Leipzig Graduate School of Management
Department of E-Business
Jahnallee 59, D-04109 Leipzig, Germany
E-Mail: kwinkler@ebusiness.hhl.de

Contents

1 Introduction

Most organizations are not only "drowning" in data, they are also "struggling" to cope with huge amounts of text documents. Currently, up to 80% of a company's information is stored in unstructured textual documents [Sul01, p. 56]. Hence, capturing interesting and actionable knowledge from textual databases is a major challenge. Creating semantic markup is one form of providing explicit knowledge about text archives to facilitate searching and browsing or to enable information integration with related data sources. Unfortunately, most users are not willing to manually create meta-data due to the efforts and costs involved. Thus, text mining techniques are required that (semi-) automatically create semantic markup.

The semantic annotation of text archives using the Extensible Markup Language XML results in application-specific, semantic meta-data in the form of XML tags and an archive-specific XML document type definition (DTD). Semantic meta-data can be utilized to facilitate for example knowledge management and information integration. Appropriate XML query languages could be employed to submit both content- and structure-based queries against XML archives. However, two main problems must be solved to semi-automatically create text annotations: Firstly, an appropriately structured, semantic DTD should be derived for each textual archive. Secondly, all text documents contained in an archive should be semantically tagged according to the previously derived document type definition.

In the next section, the DIAsDEM[1] framework for semantic tagging of large, domain-specific text archives is concisely introduced. The reader might refer to [GWS01, GSW01, WS01c, WS01a, GWS01] for a complete description of the DIAsDEM framework as well as a thorough discussion of related work. Figure 1.1 illustrates the user interface of DIAsDEM Workbench 2.0. This Java-based research prototype supports the entire framework. However, the current release does not support automated batch processing.

1.1 The DIAsDEM Framework

In DIAsDEM, the notion of semantic tagging refers to annotating texts with domain-specific XML tags that might contain additional attributes describing extracted named

[1]The acronym DIAsDEM is the name of a research project funded by Deutsche Forschungsgemeinschaft (German Research Society, http://www.dfg.de), DFG grants: SP 572/4-1 and SP 572/4-3.

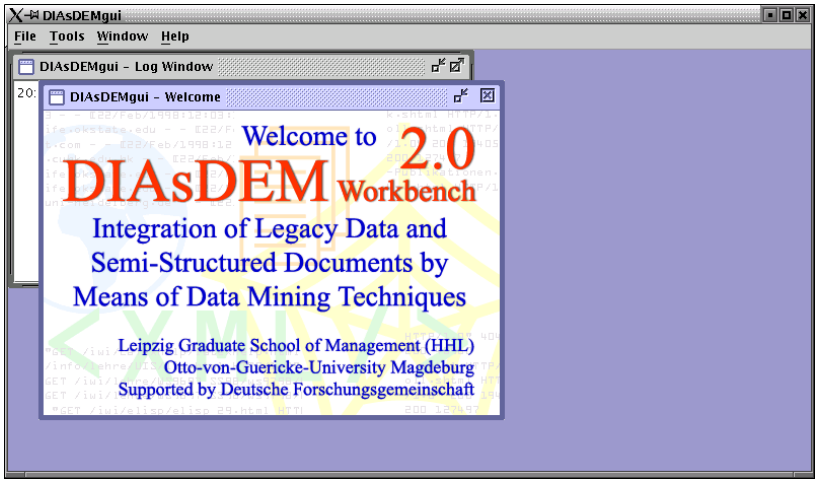

Figure 1.1: Java-based GUI of DIAsDEM Workbench 2.0

entities (e.g., names of persons). Rather than classifying entire documents or tagging single terms, we aim at semantically annotating structural text units such as sentences or paragraphs in order to make their semantics explicit. The following example illustrates two tagged sentences contained in a German Commercial Register entry, whereas each sentence is a text unit:

> <BusinessPurpose> Der Betrieb von Spielhallen in Teltow und das Aufstellen von Geldspiel- und Unterhaltungsautomaten. </BusinessPurpose>
> <AppointmentManagingDirector Person="Balski; Pawel"> Pawel Balski ist zum Geschäftsführer bestellt. </AppointmentManagingDirector>

Semantic tagging in DIAsDEM is a two-phase process. We have designed a knowledge discovery in textual databases (KDT) process that constitutes the first phase in order to build clusters of semantically similar text units, to tag documents in XML according to the results and to derive an XML DTD describing the archive. The KDT process that is depicted in Figure 1.2 results in a final set of clusters whose labels serve as XML tags and DTD elements. Huge amounts of new documents can be converted into XML documents in the second, batch-oriented and productive phase of the DIAsDEM framework.

Besides the initial text documents to be tagged, the following domain knowledge constitutes input to the KDT process: A thesaurus [ISO86] containing a domain-specific taxonomy of terms and concepts, a preliminary UML schema of the domain and descriptions of specific named entities, e.g. persons and companies. The UML schema reflects the semantics of named entities and the relationships among them, as they are initially conceived by application experts. This schema serves as a reference for the DTD to be

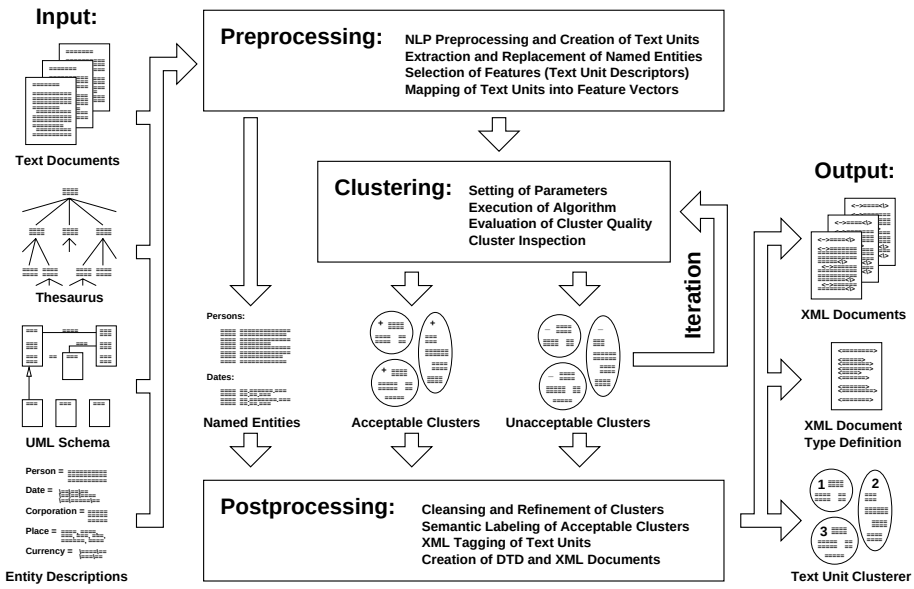

Figure 1.2: Iterative and interactive KDT process of the DIAsDEM framework

derived from discovered semantic tags. However, there is no guarantee that the final DTD will be contained in or will contain this schema.

Similarly to a conventional KDD process, the process starts with a pre-processing phase that includes basic NLP pre-processing tasks such as tokenization, normalization and word stemming as well as named entity extraction. Instead of removing stop words, we establish a drastically reduced feature space by selecting a limited set of terms and concepts (so-called text unit descriptors) from the thesaurus and the UML schema. Text unit descriptors are currently chosen by the knowledge engineer, because they must reflect important concepts of the application domain. All text units are mapped onto Boolean vectors of this feature space. Thereafter, Boolean text unit vectors are further processed by applying an information retrieval weighting schema (i.e., TFxIDF).

In the pattern discovery phase, all text unit vectors contained in the initial archive are clustered based on content similarity. The objective is to discover dense and homogeneous text unit clusters. Clustering is performed in multiple iterations. Each iteration outputs a set of clusters, which is partitioned into qualitatively “acceptable” and “unacceptable” ones according to our quality criteria. A cluster of text unit vectors is ”acceptable”, if and only if (i) its cardinality is large and the corresponding text units are (ii) homogeneous and (iii) can be semantically described by a small number of text unit descriptors. Members of “acceptable” cluster are subsequently removed from the data set for later labeling, whereas the remaining text unit vectors are input to the clus-

tering algorithm in the next iteration. In each iteration, the cluster similarity threshold value is stepwisely decreased such that "acceptable" clusters become progressively less specific in content. The KDT process is based on a plug-in and a plug-out concept that allows the execution of various clustering algorithms within DIAsDEM Workbench.

In the post-mining phase, qualitatively "acceptable" clusters are semi-automatically assigned a semantic label. DIAsDEM Workbench suggests default cluster labels for "acceptable" clusters that are derived from prevailing feature space dimensions (i.e., text unit descriptors) in each "acceptable" cluster. Cluster labels actually correspond to XML tags that are subsequently used to annotate cluster members. Thereafter, original documents are annotated by valid XML tags that include attributes reflecting previously extracted named entities and their values. Finally, an unstructured XML DTD is derived that coarsely describes the semantic structure of the XML collection by enumerating discovered XML tags. The following DTD excerpt was created in a recent case study [WS01c]:

```
<!ELEMENT CommercialRegisterEntry ( #PCDATA | BusinessPurpose | ShareCapital |
SupervisoryBoard | AppointmentManagingDirector | (...) | Owner |
FoundationPartnership )* >  <!ELEMENT BusinessPurpose (#PCDATA)> (...)
<!ELEMENT FoundationPartnership (#PCDATA)>
```

1.2 Code Credits and Trademarks

Code Contributors: Markus Banach (diasdem.kernel.preprocessor.*), Henner Graubitz (diasdem.misc.io.*, diasdem.neex.*) and Karsten Winkler (diasdem.client.*, diasdem.kernel.*, diasdem.messages.*, diasdem.misc.*, diasdem.neex.*, diasdem.objects.*, diasdem.server.*, kwinkler.*)

The research project DIAsDEM is funded by Deutsche Forschungsgemeinschaft (German Research Foundation), DFG grants SP 572/4-1 and SP 572/4-3. Information about Deutsche Forschungsgemeinschaft is available at http://www.dfg.de.

3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jdom.org.
4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management (pm@jdom.org).

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the JDOM Project and was originally created by Brett McLaughlin <brett@jdom.org> and Jason Hunter <jhunter@jdom.org>. For more information on the JDOM Project, please see <http://www.jdom.org/>.

The DIAsDEM Workbench utilizes WEKA 3.3.3, 28 June 2002, Java Programs for Machine Learning. Copyright (C) 1998, 1999, 2000, 2001, 2002 Eibe Frank, Leonard Trigg, Mark Hall, Richard Kirkby. WEKA is distributed under the GNU public license available at http://www.opensource.org/licenses/gpl-license.php.

The DIAsDEM Workbench utilizes GNU Regular Expressions for Java 1.0.8 (http://www.cacas.org/java/gnu/regexp/). GNU Regular Expressions for Java 1.0.8 is distributed under the GNU Lesser General Public License available at http://www.opensource.org/licenses/lgpl-license.php.

Sun, Sun Microsystems, the Sun logo, Sun Microsystems, JavaSoft, JavaBeans, JDK, Java, the Java Coffee Cup logo, and Visual Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other tradenames, trademarks, and registered trademarks are the property of their respective owners.

1.3 License of DIAsDEM Workbench 2.0

Copyright (c) 2000-2003, Henner Graubitz, Myra Spiliopoulou, Karsten Winkler. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

1.4 Typographical Conventions

Italic is used for emphasis within text and to indicate selectable items in DIAsDEM Workbench windows and menus. *Italic* is also used to represent field names (i.e., parameters) in DIAsDEM Workbench dialogs and windows. `Courier` is used to refer to directories, file names and file extentions. Additionally, `Courier` is used for computer output, XML tags and contents of files (e.g., XML and text files). In the remainder of this case study, the following abbreviations indicate directories on your file system as listed below. Note, these four abbreviations do not correspond to environment variables.

- `${DIAsDEM_HOME}` denotes the local directory of DIAsDEM Workbench, e.g. `/home/kwinkler/diasdem/DIAsDEM.workbench2`.
- `${PARAMETER_HOME}` denotes the local subdirectory of `${DIAsDEM_HOME}` that contains default parameter files, i.e., `${DIAsDEM_HOME}/data/parameters`.
- `${SAMPLES_HOME}` denotes the local subdirectory of `${DIAsDEM_HOME}` that contains sample text files, i.e., `${DIAsDEM_HOME}/data/samples`.
- `${PROJECT_HOME}` denotes the local directory that contains all files related to a single project, e.g., `/home/kwinkler/diasdem/DIAsDEM.cases/tutorial`.

2 Installation

2.1 Prerequisites

The target machine must be equipped with at least 256 MB memory. Either the Java 2 Runtime Environment 1.4.0 or the Java 2 Software Development Kit, Standard Edition, 1.4.0 must have been installed on the target machine. Visit the Web site http://java.sun.com to download the required Java release.

2.2 Unix/Linux

1. Visit the Web site http://www.hypknowsys.org/diasdem and download the compressed archive file `DIAsDEM.workbench2.tar.gz`.

2. Create a directory for DIAsDEM Workbench (e.g., `/home/kwinkler/diasdem`) and copy the file `DIAsDEM.workbench2.tar.gz` into this directory. Additionally, ensure that you have write permission in this new directory.

3. Make the DIAsDEM-specific directory (e.g., `/home/kwinkler/diasdem`) your current working directory and unzip the compressed file by submitting the following two commands at the prompt:

   ```
   /home/kwinkler/diasdem> gunzip DIAsDEM.workbench2.tar.gz
   /home/kwinkler/diasdem> tar -xf DIAsDEM.workbench2.tar
   ```

4. Using any common text editor, modify the environment variables `JAVA_HOME` and `DIAsDEM_HOME` in the shell script `DIAsDEM.workbench2/bin/diasdemgui` (e.g., below the directory `/home/kwinkler/diasdem`) according to your system. For example, if Java has been installed in the directory `/usr/lib/j2sdk1.4.0_01` and if the file `DIAsDEM.workbench2.tar.gz` has been uncompressed in the directory `/home/kwinkler/diasdem`, these environment variables have to be set as follows:

   ```
   DIAsDEM_HOME=/home/kwinkler/diasdem/DIAsDEM.workbench2
   JAVA_HOME=/usr/lib/j2sdk1.4.0_01
   ```

5. Make sure that the shell script `DIAsDEM.workbench2/bin/diasdemgui` (e.g., below the directory `/home/kwinkler/diasdem`) is an executable file:

   ```
   /home/kwinkler/diasdem> chmod a+x DIAsDEM.workbench2/bin/diasdemgui
   ```

6. Thereafter, DIAsDEM Workbench can be launched by executing the shell script `DIAsDEM.workbench2/bin/diasdemgui` (e.g., below `/home/kwinkler/diasdem`).

   ```
   /home/kwinkler/diasdem> DIAsDEM.workbench2/bin/diasdemgui
   ```

2.3 Windows

1. Visit the Web site http://www.hypknowsys.org/diasdem and download the compressed archive file `DIAsDEM.workbench2.zip`. Note, this zipped archive includes exactly the same contents as `DIAsDEM.workbench2.tar.gz`.

2. Create a directory for DIAsDEM Workbench (e.g., `C:\Programs\diasdem`) and copy the file `DIAsDEM.workbench2.zip` into this directory.

3. Using for example WinZip that is available at http://www.winzip.com, extract the compressed file into the DIAsDEM-specific directory (e.g., `C:\Programs\diasdem`).

4. Using any common text editor, modify the environment variables `JAVA_HOME` and `DIAsDEM_HOME` in the batch file `DIAsDEM.workbench2\bin\diasdemgui.bat` (e.g., below the directory `C:\Programs\diasdem`) according to your system. For example, if Java has been installed in `C:\Programs\Java\j2re1.4.0_01` and if the file `DIAsDEM.workbench2.zip` has been extracted in `C:\Programs\diasdem`, these environment variables have to be set as follows:

   ```
   DIAsDEM_HOME=C:\Programs\diasdem\DIAsDEM.workbench2
   JAVA_HOME=C:\Programs\Java\j2re1.4.0_01
   ```

5. Thereafter, DIAsDEM Workbench can be launched by opening Windows Explorer and double-clicking the batch file `DIAsDEM.workbench2\bin\diasdemgui.bat` (e.g., below the directory `C:\Programs\diasdem`).

6. Using Windows95 or Windows98 with standard system configurations, double-clicking `diasdemgui.bat` is likely to produce an "environment out of memory" error. In this case, the MS-DOS environment must be allocated more memory: Open Windows Explorer, right-click the icon of `diasdemgui.bat`, select *Properties*, click on the *Memory* tab and adjust *Initial Environment* from *Auto* to *2048*. After clicking on *OK* to commit the change, a PIF-file (i.e., `diasdemgui.pif`) is created that should afterwards be double-clicked to start DIAsDEM Workbench.

3 Case Study

3.1 Application Domain and Data Set

In Germany, each district court maintains a Commercial Register that contains important information about the companies in the court's district. According to law, many company activities like the establishment of branch offices, changes of share capital or mergers and acquisitions must be reported to the respective Register. Knowledge of these entries is indispensable for business activities, as they have both a right-confirmation and a right-generating effect according to the German Commercial Code.

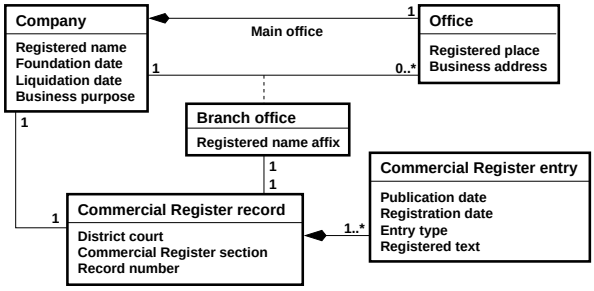

Figure 3.1: Simplified application domain (UML class diagram)

Commercial Register entries are made available to the public, since up-to-date knowledge about a company's affairs is essential to its (prospective) stakeholders. Three main categories of Commercial Register entries can be distinguished: foundation entries of new companies, update entries (e.g., changes in the managerial head of a company) and entries announcing that a company closes. The conceptual model the application domain is partly depicted as UML class diagrams in Figure 3.1 and Figure 3.2, respectively.

The directory `${SAMPLES_HOME}/de/commercialRegister1` contains 1146 German Commercial Register entries published by the district court Postsdam in 1999 via its Web site (http://www.amtsgericht-potsdam.org). Each entry announces the foundation of a new company in the Potsdam district. Table 3.1 illustrates the Commercial Register entry contained in the file `${SAMPLES_HOME}/de/commercialRegister1/file10780.txt`. All text files are ISO-8856-1 encoded and have Unix-like line feeds. However, each entry

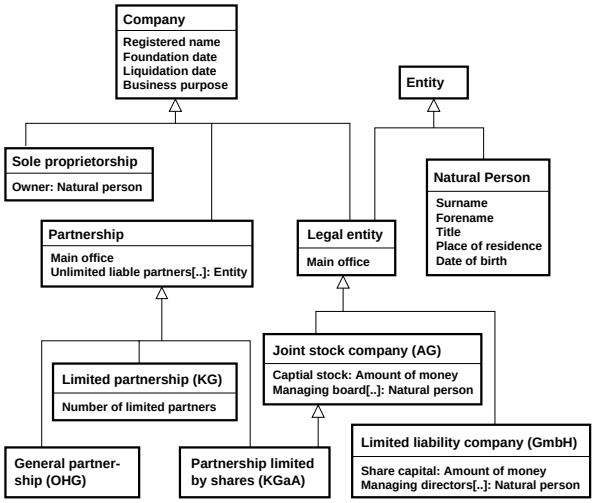

Figure 3.2: Simplified taxonomy of German companies (UML class diagram)

is stored in a single line in order to avoid line feed related problems. Note, a concise list of relevant German vocabulary based on [PDV00] is available on page 93.

Der Handel mit Waren aller Art sowie Import und Export. Der Dienstleistungsbereich bezieht sich auf Vermittlung, Beratung und Schulungen. Stammkapital: 50.000 DM. Gesellschaft mit beschränkter Haftung. Der Gesellschaftsvertrag ist am 18. April 1994 abgeschlossen und am 04. Dezember 1997 / 27. Mai 1998 abgeändert in §1 (Firma), §2 (Gegenstand) und §4 (Geschäftsführer). Durch Beschluss der Gesellschafterversammlung vom 17. November 1998 ist der Sitz der Gesellschaft von Maintal nach Damsdorf verlegt und der Gesellschaftsvertrag geändert in §1 (Firma und Sitz). Ist nur ein Geschäftsführer bestellt, so vertritt er die Gesellschaft allein. Sind mehrere Geschäftsführer bestellt, so wird die Gesellschaft durch zwei Geschäftsführer oder durch einen Geschäftsführer in Gemeinschaft mit einem Prokuristen vertreten. Einzelvertretungsbefugnis kann erteilt werden. Marion Marcella Adolph geb. Priester, 22.03.1957, Offenbach, ist zur Geschäftsführerin bestellt. Sie ist befugt, Rechtsgeschäfte mit sich selbst oder mit sich als Vertreter Dritter abzuschließen. Nicht eingetragen: Die Bekanntmachungen der Gesellschaft erfolgen im Bundesanzeiger.

Table 3.1: A German Commercial Register entry

3.2 Text Pre-Processing

In the following screen-shots of this case study, the directory `/home/kwinkler/diasdem/DIAsDEM.workbench2` corresponds to `${DIAsDEM_HOME}`. Moreover, `${PROJECT_HOME}` corresponds to the directory `/home/kwinkler/diasdem/DIAsDEM.cases/tutorial`. All

file names and directory names are Unix/Linux-based in this case study.

Create a local directory `${PROJECT_HOME}` on your machine that can be used to store all case-related files. Additionally, create an extra subdirectory `${PROJECT_HOME}/xml` for 1146 intermediate XML files used by the DIAsDEM Workbench for internal processing purposes as well as 1146 final XML files. These final XML files will contain semantically annotated Commercial Register entries after the completion of this case study.

```
DIAsDEM.cases/tutorial> pwd
/home/kwinkler/diasdem/DIAsDEM.cases/tutorial
DIAsDEM.cases/tutorial> ls
xml
```

3.2.1 Importing Text Files

Existing plain text files must be converted into an intermediary, DIAsDEM-specific format by selecting *File → Import Text Files*. Please provide the following parameters:

Parameter	Value
Text File Directory	`${SAMPLES_HOME}/de/commercialRegister1`
File Name Extension	`.txt`
Collection Directory	`${PROJECT_HOME}/xml`

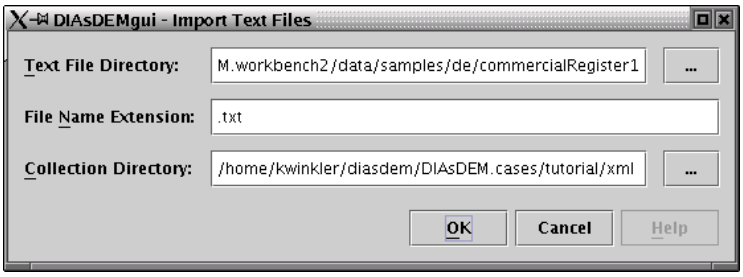

Figure 3.3: *Import Text Files* window of DIAsDEM Workbench 2.0

Please make sure to replace the abbreviations `${SAMPLES_HOME}` and `${PROJECT_HOME}` with the corresponding directories according to your individual installation of DIAsDEM Workbench. Files and directories can always be chosen by clicking on the button "..." beside the respective text field and afterwards selecting a file from the file dialog.

Click the *OK* button to start the import process: All text files that are contained in *Text File Directory* and whose file names end with *File Name Extension* will be imported into the *File Collection Directory*. Check the contents of the subdirectory `${PROJECT_HOME}/xml`:

```
DIAsDEM.cases/tutorial> ls
xml

DIAsDEM.cases/tutorial> ls -l xml | more
-rw-r--r--   1 kwinkler users      1256 Dez 8 18:09 DiasdemDocument.dtd
-rw-r--r--   1 kwinkler users      1276 Dez 8 18:09 file10001.txt.xml
-rw-r--r--   1 kwinkler users      1118 Dez 8 18:09 file10002.txt.xml ...
-rw-r--r--   1 kwinkler users      1747 Dez 8 18:09 file11160.txt.xml
```

The file `DiasdemDocument.dtd` contains the XML document type definition of intermediate DIAsDEM files. For example, `${PROJECT_HOME}/xml/file10780.txt.xml` is an XML document that contains the textual German Commercial Register entry listed in Table 3.1, i.e., `${SAMPLES_HOME}/de/commercialRegister1/file10780.txt`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE diasdemDocument SYSTEM "DiasdemDocument.dtd">
<diasdemDocument>
  <metaData>
    <name>SourceFile</name>
    <content>/home/.../samples/de/commercialRegister1/file10780.txt</content>
  </metaData>
  <text>Der Handel mit Waren aller Art sowie Import und Export. ...</text>
</diasdemDocument>
```

Import Text Files: Summary

Module: *File → Import Text Files*

Use Case: The user wants to employ DIAsDEM Workbench to semantically annotate text documents that are stored in plain text files within a single local directory. Each text file contains exactly one text document that has to be semantically annotated.

Prerequisites: None

Result: Files in *Text File Directory* whose file names end with *File Name Extension* are transformed into the intermediate DIAsDEM format (i.e., XML documents conforming to XML DTD `DiasdemDocument.dtd` described in section 4.1 on page 75) and are copied to *Collection Directory*. Additionally, the DIAsDEMgui preference *Default Collection Directory* is set.

Remarks: Alternatively, the user might employ third-party pre-processing scripts (e.g., Perl) to convert text documents into XML documents conforming to the XML DTD `DiasdemDocument.dtd`. In this case, additional metadata can be included in intermediate XML files as well.

Import Text Files: Parameters

Text File Directory: Existing local directory that contains text files to be imported

File Name Extension: Names of text files to be imported must end with this file name extension; default value: `.txt`

Collection Directory: Existing local directory that will contain the imported XML documents; proposed value: `${PROJECT_HOME}/xml`

3.2.2 Creating a New Collection File

Intermediate DIAsDEM files that belong to the same application domain are considered to be a collection of related documents. Only one collection can be processed by DIAsDEM Workbench at any time. A collection file contains meta-data about the archive as well as references to local intermediate XML files constituting the collection. For certain intermediate DIAsDEM files in a local directory, a collection file can be created by selecting *File* → *New Collection File*. Please input the following parameters:

Parameter	Value
Collection Directory	`${PROJECT_HOME}/xml`
File Name Extension	`.xml`
Collection File	`${PROJECT_HOME}/collection.dcf`
Advanced Options	`Yes, Perform XML Validity Check`

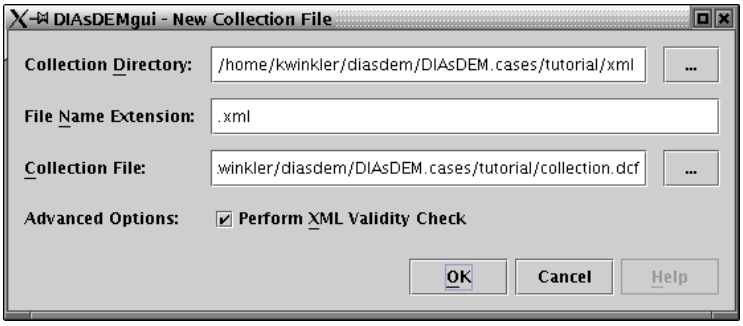

Figure 3.4: *New Collection File* window of DIAsDEM Workbench 2.0

Click on *OK* to create a new collection file: All intermediate files that are contained in *Collection Directory* and whose file names end with *File Name Extension* will be added to *Collection File*. Furthermore, a check is performed for all files in order to check their well-formedness and validity with respect to their XML DTD. The directory `${PROJECT_HOME}/xml` should now contain the following files:

```
DIAsDEM.cases/tutorial> ls
collection.dcf collection.dcf.files xml

DIAsDEM.cases/tutorial> more collection.dcf
#This is an automatically created file: Please do not edit this file manually!
#Sun Dec 08 23:31:30 CET 2002
COLLECTION_DIRECTORY=/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/xml/
COLLECTION_FILE_NAME=/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/collection.dcf.files
NUMBER_OF_DOCUMENTS=1146

DIAsDEM.cases/tutorial> tail -n 3 collection.dcf.files
/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/xml/file10531.txt.xml
/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/xml/file10705.txt.xml
/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/xml/file10889.txt.xml
```

Subsequently, the file `collection.dcf` is referred to as *Collection File* that uniquely identifies the corresponding archive and contains relevant meta-data. Each *Collection File* is accompanied by an auxiliary file whose file extention is `.dcf.files`, e.g. `collection.dcf.files`. This file only contains absolute file names of all intermediate XML files comprising the collection. Neither of these two files should be modified or deleted manually.

New Collection File: Summary

Module: *File → New Collection File*

Use Case: The user wants to employ the DIAsDEM Workbench to semantically annotate texts that are stored as XML documents in a single local directory.

Prerequisites: Each XML file must conform to the XML DTD `DiasdemDocument.dtd` that is described in section 4.1 on page 75.

Result: A new *Collection File* is created and all files in *Collection Directory* whose file names end with *File Name Extension* are part of the collection. Additionally, the DIAsDEMgui preferences *Default Collection File*, *Default Collection Directory* and *Default Project Directory* are set.

Remarks: All subsequent processing modules of the DIAsDEM Workbench require a specific *Collection File* as an input parameter.

New Collection File: Parameters

Collection Directory: Existing local directory that contains XML documents of the archive to be tagged; default value: DIAsDEMgui preference *Default Collection Directory*

File Name Extension: Names of XML files in the new collection must end with the specified file name extension; default value: `.xml`

Collection File: Valid local file name of new collection file; file extension: `.dcf`; proposed value: `${PROJECT_HOME}/collection.dcf`

Advanced Options: If *Perform XML Validity Check* is enabled, only well-formed and valid XML documents will be added to *Collection File*.

3.2.3 Creating Text Units

After importing text files and creating a collection file, the text pre-processing phase starts with identifying and separating text units. In this case study, each sentence of Commercial Register entries corresponds to a text unit. Hence, only sentences will be semantically annotated by DIAsDEM Workbench. Select *File* → *Create Text Units* and type in the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Text Unit Algorithm	`Heuristic Sentence Identifier`
Abbreviations File	`${PARAMETER_HOME}/createTextUnits/de/AbbreviationsDE.txt`
Full Stop Regex File	`${PARAMETER_HOME}/createTextUnits/de/FullStopRegexDE.txt`
Replaced Full Stops	`Keep Asterisks for Tokenization`

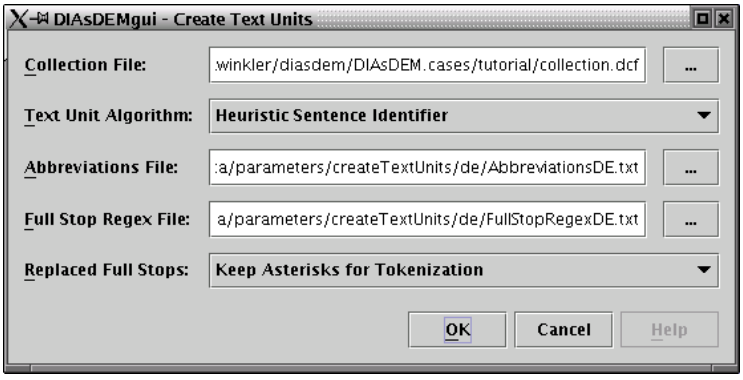

Figure 3.5: *Create Text Units* window of DIAsDEM Workbench 2.0

Click the *OK* button to start the process of identifying and separating sentences within text documents. In the entire text, Heuristic Sentence Identifier first replaces full stops in abbreviations (e.g., "z.B.") listed in *Abbreviations File* with asterisks. Thereafter, all regular expressions contained in *Full Stop Regex File* are matched against the text. These

regular expressions match full stops that are no sentence boundaries (e.g., "01.01.2002") and replace them with asterisks as well. Both textual parameter files can be edited in order to include additional domain knowledge.

Have a look at the contents of the file `${PROJECT_HOME}/xml/file10780.txt.xml`. This intermediate XML document has been extended by the new section `<textUnits>` whose elements `<textUnit>` mark up single sentences.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE diasdemDocument SYSTEM "DiasdemDocument.dtd">
<diasdemDocument> ...
  <text>Der Handel mit Waren aller Art sowie Import und Export. ...</text>
  <textUnits>
    <textUnit>Der Handel mit Waren aller Art sowie Import und Export.</textUnit> ...
    <textUnit>Stammkapital: 50*000 DM.</textUnit>
    <textUnit>Gesellschaft mit beschränkter Haftung.</textUnit> ...
    <textUnit>Einzelvertretungsbefugnis kann erteilt werden.</textUnit>
    <textUnit>Marion Marcella Adolph geb* Priester, 22*03*1957, Offenbach, ist zur
    Geschäftsführerin bestellt.</textUnit> ...
    <textUnit>Nicht eingetragen: Die Bekanntmachungen der Gesellschaft erfolgen im
    Bundesanzeiger.</textUnit>
  </textUnits>
</diasdemDocument>
```

Each asterisk either replaces a full stop in a known abbreviation (e.g., "geb." is contained in *Abbreviations File*) or that is matched by a regular expression in *Full Stop Regex File*. For example, the date literal "22.03.1957" is matched by the regular expression "([0-9]{1,2})\.([\ ]*[0-9]{1,2})\.([\ ]*[0-9]{2,4})". Due to this match, the original date literal "22.03.1957" has been replaced by the corresponding replacement string "$1*$2*$3" which results in "22*03*1957".

Create Text Units: Summary

Module: *File → Create Text Units*

Use Case: The user must pre-process all imported texts as part of the DIAsDEM KDD process for semantic tagging of domain-specific texts archives. Creating text units is pre-processing phase 1 of 5.

Prerequisites: Each XML file must conform to the XML DTD `DiasdemDocument.dtd` that is described in section 4.1 on page 75.

Result: Each intermediate XML file of the collection is extended by a new section `<textUnits>` whose elements `<textUnit>` mark up either single sentences (*Heuristic Sentence Identifier*) or the entire text (*Text as a Single Text*

Unit). Any previously existing section `<textUnits>` will be completely replaced by a new one.

Additionally, the DIAsDEMgui preferences *Default Collection File*, *Default Project Directory*, *Default Abbreviations File* and *Default Full Stop Regex File* are set and updated, respectively.

Remarks: Creating text units is a prerequisite for the remaining pre-processing phases 2 (i.e., tokenization) through 5 (i.e., lemmatization). Take the following side effect into consideration: All asterisks contained in the original text document might subsequently be replaced by full stops.

Create Text Units: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

Text Unit Algorithm: If the recommended option *Heuristic Sentence Identifier* is enabled, this module heuristically identifies sentences for subsequent semantic annotation. If the option *Text as a Single Text Unit* is enabled, the entire text will be marked up as a single text unit. In the latter case, the entire text will be annotated by one XML tag.

Abbreviations File: Valid local file name of existing file that contains known abbreviations in the format described in section 4.2.1 on page 76; file extension: `.txt`; default value: DIAsDEMgui preference *Default Abbreviations File*

Full Stop Regex File: Valid local file name of existing file that contains regular expressions in the format described in section 4.2.1 on page 76; file extension: `.txt`; default value: DIAsDEMgui preference *Default Full Stop Regex File*

Replaced Full Stops: If the recommended option *Keep Asterisks for Tokenization* is enabled, asterisks that replace full stops will be retained for usage in the subsequent tokenization phase. Otherwise, all asterisks will be replaced by full stops before this module terminates.

3.2.4 Tokenizing Text Units

After creating text units, tokenizing text units is the second phase of text pre-processing. Tokenization segments text units into individual words and tokens, respectively. Additionally, text units are normalized in order to map for example various different date literals (e.g., “1 Jan 2003” and “1.1.2003”) onto a canonical date representation (e.g., “01.01.2003”). Moreover, multi-token terms such as “for example” are identified to subsequently process them as single tokens. Select *File* → *Create Text Units* and input the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Tokenize Regex File	`${PARAMETER_HOME}/createTextUnits/de/TokenizeRegexDE.txt`
Normalize Regex File	`${PARAMETER_HOME}/createTextUnits/de/NormalizeRegexDE.txt`
Multi Token Words File	`${PARAMETER_HOME}/createTextUnits/de/MultiTokenWordsDE.txt`
Replaced Full Stops	`Replace Asterisks in Text Units`

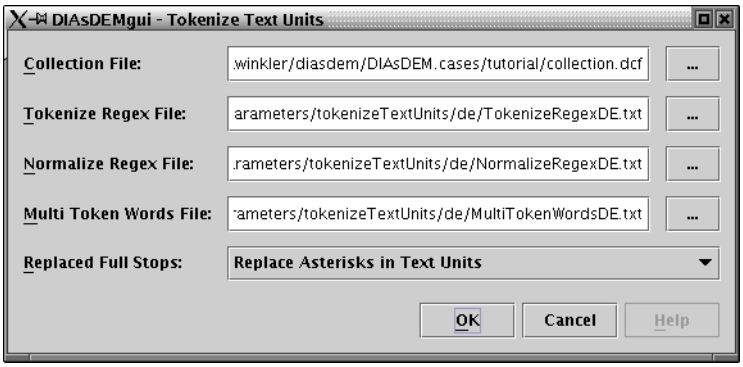

Figure 3.6: *Tokenize Text Units* window of DIAsDEM Workbench 2.0

Click on *OK* to start the process of text unit tokenization, text unit normalization and identification of multi-token terms in text units. Each processing step can be fully parameterized by editing regular expressions in the corresponding parameter file. The heuristic normalization algorithm does not separate asterisks from their surrounding characters, because an asterisk corresponds to a previously replaced full stop. Therefore, *Tokenize Regex File* should not include regular expressions matching asterisks. Text units are normalized by applying regular expressions and substituting matching sequences with the corresponding replacement string. Additionally, blank spaces in known multi-token terms (e.g., "for example") are replaced by underscores (e.g., "for_example"). Finally, all asterisks that are found in the already existing section `<textUnits>` as well as the new section `<textUnitsTokenized>` are replaced by full stops.

Check the contents of the file `${PROJECT_HOME}/xml/file10780.txt.xml`. This intermediate XML document has been extended by the new section `<textUnitsTokenized>` whose elements `<textUnitTokenized>` mark up single, tokenized and normalized sentences.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE diasdemDocument SYSTEM "DiasdemDocument.dtd">
<diasdemDocument> ...
  <text>Der Handel mit Waren aller Art sowie Import und Export. ...</text>
  <textUnits> ...
```

```
    <textUnit>Stammkapital: 50.000 DM.</textUnit>
    <textUnit>Gesellschaft mit beschränkter Haftung.</textUnit> ...
    <textUnit>Marion Marcella Adolph geb. Priester, 22.03.1957, Offenbach, ist zur
    Geschäftsführerin bestellt.</textUnit> ...
    <textUnit>Nicht eingetragen: Die Bekanntmachungen der Gesellschaft erfolgen im
    Bundesanzeiger.</textUnit>
  </textUnits>
  <textUnitsTokenized> ...
    <textUnitTokenized>Stammkapital : 50000 DM .</textUnitTokenized>
    <textUnitTokenized>Gesellschaft_mit_beschränkter_Haftung .</textUnitTokenized> ...
    <textUnitTokenized>Marion Marcella Adolph geb. Priester , 22.03.1957 , Offenbach ,
    ist zur Geschäftsführerin bestellt .</textUnitTokenized> ...
    <textUnitTokenized>Nicht eingetragen : Die Bekanntmachungen der Gesellschaft
    erfolgen im Bundesanzeiger .</textUnitTokenized>
  </textUnitsTokenized>
</diasdemDocument>
```

The module works as follows: Firstly, all regular expressions listed in *Tokenization Regex File* are matched against each text unit. For example, 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 replacement string "$1\$2" which results in the following tokenized text: "This is a sentence . There". Secondly, each regular expression listed in *Normalization Regex File* is matched against the text units. Analogously, matching character subsequences are substituted by the corresponding replacement string. Thirdly, multi-token terms contained in *Multi Token Words File* are looked up in all text units. Each identified multi-token term is reduced to a single token by replacing existing term-separating blank spaces with underscores.

Tokenize Text Units: Summary

Module: *File → Tokenize Text Units*

Use Case: The user must pre-process all imported texts as part of the DIAsDEM KDD process for semantic tagging of domain-specific texts archives. Tokenizing text units is pre-processing phase 2 of 5.

Prerequisites: Each XML file must conform to the XML DTD `DiasdemDocument.dtd` that is described in section 4.1 on page 75. Specifically, it must contain the non-mandatory section `<textUnits>`.

Result: Each intermediate XML file of the collection is extended by a new section `<textUnitsTokenized>` whose elements `<textUnitTokenized>` mark up

tokenized and normalized text units. Identified multi-token are modified by replacing blank spaces with underscores. Any previously existing section `<textUnitsTokenized>` will be completely replaced by a new one.

Additionally, the DIAsDEMgui preferences *Default Collection File*, *Default Project Directory*, *Default Normalize Regex File*, *Default Tokenize Regex File* and *Default Multi Token Words File* are set and updated, respectively.

Remarks: Tokenizing text units is a prerequisite for the remaining pre-processing phases 3 (i.e., named entity extraction) through 5 (i.e., lemmatization).

Tokenize Text Units: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

Tokenize Regex File: Valid local file name of existing file that contains regular expressions in the format described in section 4.2.2 on page 77; file extension: `.txt`; default value: DIAsDEMgui preference *Default Tokenize Regex File*;

Normalize Regex File: Valid local file name of existing file that contains regular expressions in the format described in section 4.2.2 on page 77; file extension: `.txt`; default value: DIAsDEMgui preference *Default Normalize Regex File*

Multi Token Words File: Valid local file name of existing file that contains known multi-token terms in the format described in section 4.2.2 on page 77; file extension: `.txt`; default value: DIAsDEMgui preference *Default Multi Token Words File*

Replaced Full Stops: If the recommended option *Replace Asterisks in Text Units* is enabled, all asterisks contained in sections `<textUnits>` and `<textUnitsTokenized>` will be replaced by full stops before this module terminates. Otherwise, asterisks contained in these section will not be replaced by full stops.

3.2.5 Replacing Named Entities

After creating and tokenizing text units, identifying named entities (e.g., persons or companies) and replacing them by placeholders constitutes the third phase of text pre-processing. Instances of extracted named entities might thereafter serve as attribute values in semantic XML tags. For example, “Karsten Winkler” is an instance of named entity type “person”. Based on lists and regular expressions, DIAsDEM Workbench can

currently identify named entities of the following types: “person”, “company”, “place”, “date”, “amount_of_money”, “paragraph”, “email” and “url”. Select *File* → *Replace Named Entities* and type in the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Forenames File	`${PARAMETER_HOME}/replaceNamedEntities/de/default/ForenamesDE.txt`
Surnames File	`${PARAMETER_HOME}/replaceNamedEntities/de/default/SurnamesDE.txt`
Surname Suffixes File	`${PARAMETER_HOME}/replaceNamedEntities/de/default/SurnameSuffixesDE.txt`
Middle Initials File	`${PARAMETER_HOME}/replaceNamedEntities/de/default/MiddleInitialsDE.txt`
Titles File	`${PARAMETER_HOME}/replaceNamedEntities/de/default/TitlesDE.txt`
Places File	`${PARAMETER_HOME}/replaceNamedEntities/de/default/PlacesDE.txt`
Organizations Start File	`${PARAMETER_HOME}/replaceNamedEntities/de/default/OrganizationsStartDE.txt`
Organizations End File	`${PARAMETER_HOME}/replaceNamedEntities/de/default/OrganizationsEndDE.txt`
Composite NE File	`${PARAMETER_HOME}/replaceNamedEntities/de/commercialRegister/CommercialRegisterExtentedCompositeNE.csv`
Regex NE File	`${PARAMETER_HOME}/replaceNamedEntities/de/commercialRegister/CommercialRegisterRegexNE.csv`
Advanced Options	`Disabled: Skip Named Entity Replacement, but Copy Text Units`

Click on *Save* to set or update DIAsDEMgui preferences that correspond to the current parameter settings. Click the *OK* button to identify and replace named entities. Thereafter, check the contents of the file `${PROJECT_HOME}/xml/file10780.txt.xml`. This intermediate XML document has been extended by two new sections. The first new section `<namedEntities>` contains one element `<namedEntity>` for each identified basic, composite and canonical named entity. `<textUnitsNamedEntities>` is the second new section and consists of elements `<textUnitsNamedEntity>` that mark up tokenized and normalized sentences containing placeholders for identified named entities.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE diasdemDocument SYSTEM "DiasdemDocument.dtd">
<diasdemDocument> ...
  <textUnitsTokenized> ...
    <textUnitTokenized>Stammkapital : 50000 DM .</textUnitTokenized>
    <textUnitTokenized>Gesellschaft_mit_beschränkter_Haftung .</textUnitTokenized> ...
    <textUnitTokenized>Marion Marcella Adolph geb. Priester , 22.03.1957 , Offenbach ,
    ist zur Geschäftsführerin bestellt .</textUnitTokenized> ...
    <textUnitTokenized>Nicht eingetragen : Die Bekanntmachungen der Gesellschaft
    erfolgen im Bundesanzeiger .</textUnitTokenized>
  </textUnitsTokenized>
  <namedEntities> ...
```

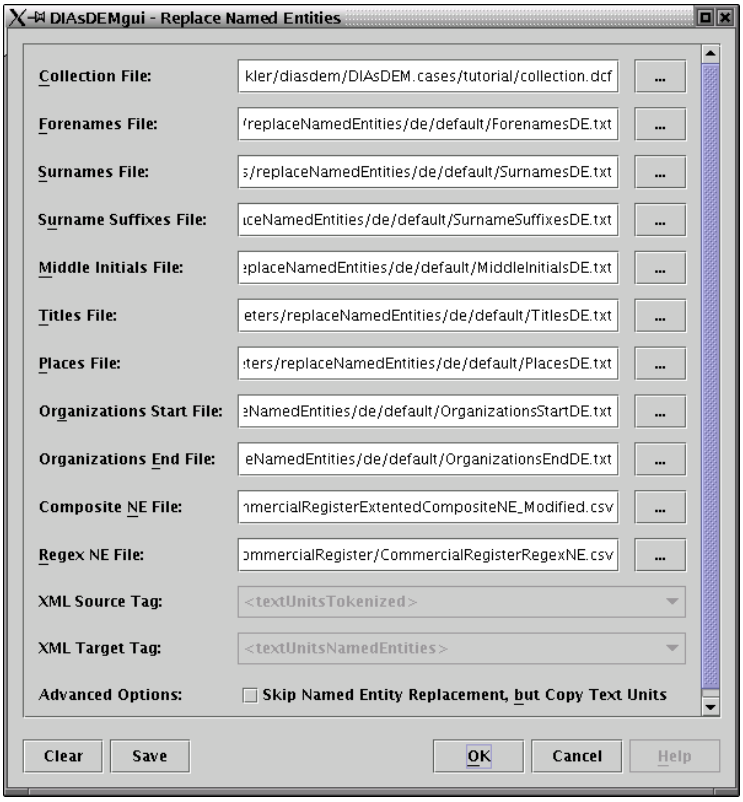

Figure 3.7: *Replace Named Entities* window of DIAsDEM Workbench 2.0

```
  <namedEntity id="1001" type="amount_of_money">50000 DM</namedEntity> ...
  <namedEntity id="1012" type="date">22.03.1957</namedEntity>
  <namedEntity id="1013" type="forename">Marion</namedEntity>
  <namedEntity id="1014" type="forename">Marcella</namedEntity>
  <namedEntity id="1015" type="forename surname">Adolph</namedEntity>
  <namedEntity id="1016" type="place surname">Priester</namedEntity>
  <namedEntity id="1017" type="place">Offenbach</namedEntity>
  <namedEntity id="1018" type="person">1018|null|person|null|Adolph|Marion Marcella|
  22.03.1957|null|null|null|null|Priester|Offenbach|null</namedEntity>
  <namedEntity id="1019" type="person">1019|null|person|1018|Adolph|Marion Marcella|
  22.03.1957|null|null|null|null|Priester|Offenbach|null</namedEntity>
</namedEntities>
<textUnitsNamedEntities> ...
  <textUnitNamedEntities>Stammkapital : <namedEntityRef id="1001" />
  . </textUnitNamedEntities>
  <textUnitNamedEntities>Gesellschaft_mit_beschränkter_Haftung
  . </textUnitNamedEntities> ...
  <textUnitNamedEntities><namedEntityRef id="1019" /> , ist zur Geschäftsführerin
```

```
    bestellt . </textUnitNamedEntities> ...
    <textUnitNamedEntities>Nicht eingetragen : Die Bekanntmachungen der Gesellschaft
    erfolgen im Bundesanzeiger . </textUnitNamedEntities>
  </textUnitsNamedEntities>
</diasdemDocument>
```

The named entity extractor NEEX can be fully parameterized by editing the corresponding parameter files. NEEX is a heuristic named entity extraction module of the DIAsDEM Workbench that works as follows:

1. Firstly, regular expressions listed in *Regex NE File* are matched against each text unit to identify instances of the basic named entities "number", "date", "time", "amount_of_money", "paragraph", "email" and "url". For example, named entity 1001 is an instance "50000 DM" of named entity type "amountOfMoney" that occurs in the tokenized and normalized sentence "Stammkapital : 50000 DM ."

2. Secondly, all tokens of each text unit are looked up in a dictionary to identify instances of the basic named entities "place", "organization_abbreviation", "forename", "surname", "title" and "middle_initial". Basically, this dictionary comprises the contents of *Places File*, *Organizations End File*, *Forenames File*, *Surnames File*, *Titles File* and *Middle Initials File*. Additionally, *Surname Suffixes File* is used to identify rare surnames due to frequently occuring surname suffixes such as "isz". Organizations are extracted by utilizing known abbreviations of organizations listed in *Organizations End File*. Additionally, *Organizations Start File* contains terms and phrases that frequently precede names of organizations. All basic named entities are replaced by their respective placeholders in this phase.

   In the example above, "Marion" is an instance of named entity type "forename" and "Offenbach" instantiates the named entity type "place". Note that a single token can instantiate various basic named entities. For example, the term "Adolph" could be either a surname or a forename.

3. Thirdly, rules contained in *Composite NE File* are applied to text units in order to identify instances of composite named entities, i.e., "person" and "company". Each composite named entity consists of certain basic named entities that occur in a context described by rules in *Composite NE File*. For instance, a "person" can be constructed from basic named entities such as "forename" and "surname". If a composite named entity is identified, both textual contents and basic named entity placeholders matched by the rule will be substituted by the corresponding composite named entity placeholder.

   For instance, the complex rule "<<forename>> <<forename>> <<surname>> geb. <<surname>> , <<date>> , <<place>>" listed in *Composite NE File* results in the instantiation of the complex named entity with ID 1018. This named

entity maps the text "Marion Marcella Adolph geb. Priester , 22.03.1957 , Offenbach" onto an instance of composite named entity type "person".

4. Finally, NEEX applies heuristics to map various occurrences of identical persons and companies within one text document onto a canonical form. Thereby, all composite named entity placeholders that probably reference the same real world entity are replaced by a new placeholder referencing the canonical form of the corresponding named entities.

   In the example above, named entity with ID 1019 corresponds to the canonical form of person with ID 1018. However, Marion Marcella Adolph is referred to only once in this Commercial Register entry. Therefore, open the file `${PROJECT_HOME}/xml/file10781.txt.xml` that apparently illustrates the concept of canonical complex named entities. In this file, the canonical named entity with ID 1058 replaces all other occurrences of named entities (i.e., IDs 1038 and 1053) referring to the real-world person "Willi Steinberg".

Replace Named Entities: Summary

Module: *File → Replace Named Entities*

Use Case: The user must pre-process all imported texts as part of the DIAsDEM KDD process for semantic tagging of domain-specific texts archives. Identifying and replacing named entities is pre-processing phase 3 of 5.

Prerequisites: Each XML file must conform to the XML DTD `DiasdemDocument.dtd` that is described in section 4.1 on page 75. Specifically, it must contain the non-mandatory sections `<textUnits>` and `<textUnitsTokenized>`.

Result: Each intermediate XML file of the collection is extended by two sections: All identified named entities are contained in the elements `<namedEntity>` of section `<namedEntities>`. The elements `<textUnitNamedEntities>` of the new section `<textUnitsNamedEntities>` mark up text units containing placeholders for extracted named entities. Previously existing sections `<namedEntities>` or `<textUnitsNamedEntities>` will be completely replaced by new ones.

Additionally, the DIAsDEMgui preferences *Default Collection File*, *Default Project Directory*, *Default Forenames File*, *Default Surnames File*, *Default Surname Suffixes File*, *Default Middle Initials File*, *Default Title File*, *Default Places File*, *Default Organizations Start File*, *Default Organizations End File*, *Default Composite NE File* and *Default Regex NE File* are set and updated on request, respectively.

Remarks: Identifying and replacing named entities in text units or actively skipping named entity replacement is a prerequisite for the two remaining pre-processing phases (i.e., stopword removal and lemmatization).

Replace Named Entities: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

Forenames File: Valid local file name of existing file that contains a list of forenames in the format described in section 4.2.3 on page 78; file extension: `.txt`; default value: DIAsDEMgui preference *Default Forenames File*

Surnames File: Valid local file name of existing file that contains a list of surnames in the format described in section 4.2.3 on page 78; file extension: `.txt`; default value: DIAsDEMgui preference *Default Surnames File*

Surname Suffixes File: Valid local file name of existing file that contains a list of frequent surname suffixes in the format described in section 4.2.3 on page 78; file extension: `.txt`; default value: DIAsDEMgui preference *Default Surname Suffixes File*

Middle Initials File: Valid local file name of existing file that contains a list of middle initials in the format described in section 4.2.3 on page 78; file extension: `.txt`; default value: DIAsDEMgui preference *Default Middle Initials File*

Title File: Valid local file name of existing file that contains a list of academic and professional titles in the format described in section 4.2.3 on page 78; file extension: `.txt`; default value: DIAsDEMgui preference *Default Titles File*

Places File: Valid local file name of existing file that contains a list of places (i.e., cities) in the format described in section 4.2.3 on page 78; file extension: `.txt`; default value: DIAsDEMgui preference *Default Places File*

Organizations Start File: Valid local file name of existing file that contains terms that are followed by an organization ending with a known abbreviation in the format described in section 4.2.3 on page 78; file extension: `.txt`; default value: DIAsDEMgui preference *Default Organizations Start File*

Organizations End File: Valid local file name of existing file that contains a list of organizational abbreviations in the format described in section 4.2.3 on page 78; file extension: `.txt`; default value: DIAsDEMgui preference *Default Organizations End File*

Composite NE File: Valid local file name of existing file that contains rules for instantiating composite named entities (i.e., persons and companies) in the format described in section 4.2.3 on page 78; file extension: `.csv`; default value: DIAsDEMgui preference *Default Composite NE File*;

Regex NE File: Valid local file name of existing file that contains regular expressions for the identification of basic named entities in the format described in section 4.2.3 on page 78; file extension: `.csv`; default value: DIAsDEMgui preference *Default Regex NE File*

Advanced Options: If the option *Skip Named Entity Replacement, but Copy Text Units* is enabled, named entity identification is completely skipped, but a new section `<textUnitsNamedEntities>` is created by copying the contents of `<textUnitsTokenized>`. This option enables users to quickly meet the prerequisites of the following pre-processing step (i.e., stopword removal) without having to bother with named entity extraction.

3.2.6 Removing Stopwords

DIAsDEM Workbench is capable of either removing meaningless stopwords or skipping stopword removal. In the latter case, a new section `<textUnitsStopwords>` is created in each intermediate XML file by simply copying the contents of the existing section `<textUnitsNamedEntities>`. As explained in section 1, the DIAsDEM framework proposes the utilization of a controlled vocabulary (i.e., a domain-specific thesaurus) for dimension reduction. Thus, stopword removal can be skipped in this case study. Select *File → Remove Stopwords* and type in the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Stopword File	
Advanced Options	`Enabled: Skip Stopword Removal, but Copy Text Units`

Click on *OK* to start the process of skipping stopword removal. Check the contents of the file `${PROJECT_HOME}/xml/file10780.txt.xml` which has been extended by the new section `<textUnitsStopwords>`. In this case study, each element `<textUnitStopwords>` is an exact copy of corresponding element `<textUnitNamedEntities>`.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE diasdemDocument SYSTEM "DiasdemDocument.dtd">
<diasdemDocument> ...
  <textUnitsNamedEntities> ...
    <textUnitNamedEntities>Stammkapital : <namedEntityRef id="1001" />
    . </textUnitNamedEntities>
    <textUnitNamedEntities>Gesellschaft_mit_beschränkter_Haftung
```

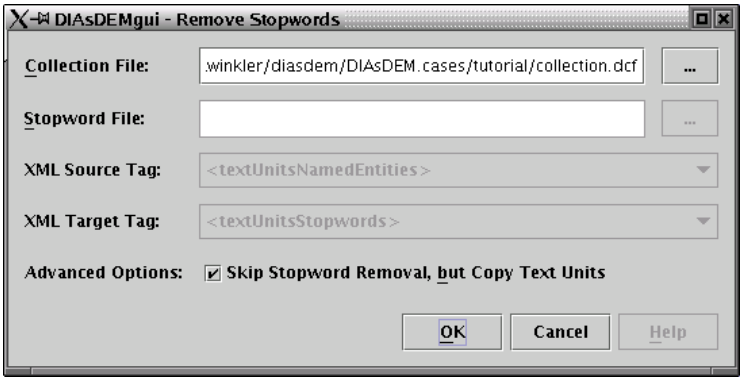

Figure 3.8: *Remove Stopwords* window of DIAsDEM Workbench 2.0

```
    . </textUnitNamedEntities> ...
    <textUnitNamedEntities><namedEntityRef id="1019" /> , ist zur Geschäftsführerin
    bestellt . </textUnitNamedEntities> ...
    <textUnitNamedEntities>Nicht eingetragen : Die Bekanntmachungen der Gesellschaft
    erfolgen im Bundesanzeiger . </textUnitNamedEntities>
  </textUnitsNamedEntities>
  <textUnitsStopwords> ...
    <textUnitStopwords>Stammkapital : <namedEntityRef id="1001" />
    . </textUnitStopwords>
    <textUnitStopwords>Gesellschaft_mit_beschränkter_Haftung
    . </textUnitStopwords> ...
    <textUnitStopwords><namedEntityRef id="1019" /> , ist zur Geschäftsführerin
    bestellt . </textUnitStopwords> ...
    <textUnitStopwords>Nicht eingetragen : Die Bekanntmachungen der Gesellschaft
    erfolgen im Bundesanzeiger . </textUnitStopwords>
  </textUnitsStopwords>
</diasdemDocument>
```

If stopword removal had been enabled, all stopwords listed in *Stopword File* would have been removed from the elements of section `<textUnitsStopwords>`. The text file `${PARAMETER_HOME}/removeStopwords/de/StopwordsDE.txt` contains a default German stopword list that can be modified according to domain-specific needs.

Remove Stopwords: Summary

Module: *File → Remove Stopwords*

Use Case: The user must pre-process all imported texts as part of the DIAsDEM KDD process for semantic tagging of domain-specific texts archives. Removing stopwords is pre-processing phase 4 of 5.

Prerequisites: Each XML file must conform to the XML DTD `DiasdemDocument.dtd` that is described in section 4.1 on page 75. Specifically, it must contain the non-mandatory sections `<textUnits>`, `<textUnitsTokenized>` and `<textUnitsNamedEntities>`.

Result: Each intermediate XML file of the collection is extended by a new section `<textUnitsStopwords>`. Unless *Skip Stopword Removal, but Copy Text Units* is enabled, elements `<textUnitStopwords>` will not contain terms listed in *Stopwords File*. Any previously existing section `<textUnitsStopwords>` will be completely replaced by a new one.

Additionally, the DIAsDEMgui preferences *Default Collection File*, *Default Project Directory* and *Default Stopwords File* are set and updated, respectively.

Remarks: Removing stopwords or actively skipping stopword removal is a prerequisite for the remaining pre-processing phase (i.e., lemmatization).

Remove Stopwords: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

Stopwords File: Valid local file name of existing file that contains stopwords in the format described in section 4.2.4 on page 83; file extension: `.txt`; default value: DIAsDEMgui preference *Default Stopwords File*

Advanced Options: If *Skip Stopword Removal, but Copy Text Units* is enabled, stopword removal is completely skipped, but a new section `<textUnitsStopwords>` is created by copying contents from the existing section `<textUnitsNamedEntities>`. This option enables users to quickly meet the prerequisites of the following pre-processing step (i.e., lemmatization) without having to remove stopwords.

3.2.7 Creating Lemma Forms

Lemmatization of terms is the final text pre-processing step. During this step, the grammatical roots of terms (i.e., their lemma form) are determined and each term is replaced with its lemma form. For example, each inflected verb form (e.g., "went") is mapped onto its infinite form (e.g., "go"). This pre-processing step drastically reduces the number of distinct terms occuring in the collection. Hence, lemmatization also facilitates both the establishment and the usage of archive-specific thesauri which are required by DIAsDEM Workbench for controlled dimension reduction.

DIAsDEM Workbench supports two different methods of creating lemma forms. They can either be automatically determined by TreeTagger or each term can be looked up in a user-supplied list of known lemma forms. Note that using TreeTagger is the preferred method of lemmatization. TreeTagger is a multilingual part-of-speech tagger developed by Helmut Schmid [Sch94]. Currently, TreeTagger for Linux and Solaris can be used for research purposes free of charge. However, the list-based method of determining lemma forms is applied in this case study to avoid any problems with installing the part-of-speech tagger. In contrast to 'real' part-of-speech tagging, this lexicon-based method has the following main disadvantage: Lemma forms can only be determined for terms whose grammatical root forms are listed in the file of known lemma forms. However, select *File* → *Create Lemma Forms* and type in the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Lemmatization Algorithm	`Look Up Lemma Form in List`
TreeTagger Input File	
TreeTagger Output File	
Known Lemma Forms	`${PARAMETER_HOME}/lemmaForms/de/Case1LemmaFormsTreeTagger.txt`
Unknown Lemma Forms	`${PARAMETER_HOME}/lemmaForms/de/Case1LemmaFormsTreeTagger_NewTerms.txt`
Advanced Options	`Disabled: Skip Lemmatization, but Copy Text Units` `Disabled: Create New Known Lemma Forms File`

Click the *OK* button to start the process of lemmatization. Thereafter, check the contents of the file `${PROJECT_HOME}/xml/file10780.txt.xml` which has been extended by the new section `<textUnitsLemmaForms>`. Its elements `<textUnitLemmaForms>` mark up lemma forms and named entity placeholders of the corresponding text units.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE diasdemDocument SYSTEM "DiasdemDocument.dtd">
<diasdemDocument> ...
  <textUnitsStopwords> ...
    <textUnitStopwords>Stammkapital : <namedEntityRef id="1001" />
    . </textUnitStopwords>
    <textUnitStopwords>Gesellschaft_mit_beschränkter_Haftung
    . </textUnitStopwords> ...
    <textUnitStopwords><namedEntityRef id="1019" /> , ist zur Geschäftsführerin
    bestellt . </textUnitStopwords> ...
    <textUnitStopwords>Nicht eingetragen : Die Bekanntmachungen der Gesellschaft
    erfolgen im Bundesanzeiger . </textUnitStopwords>
  </textUnitsStopwords>
  <textUnitsLemmaForms> ...
    <textUnitLemmaForms>Stammkapital : <namedEntityRef id="1001" />
    . </textUnitLemmaForms>
    <textUnitLemmaForms>Gesellschaft_mit_beschränkter_Haftung_unknown
```

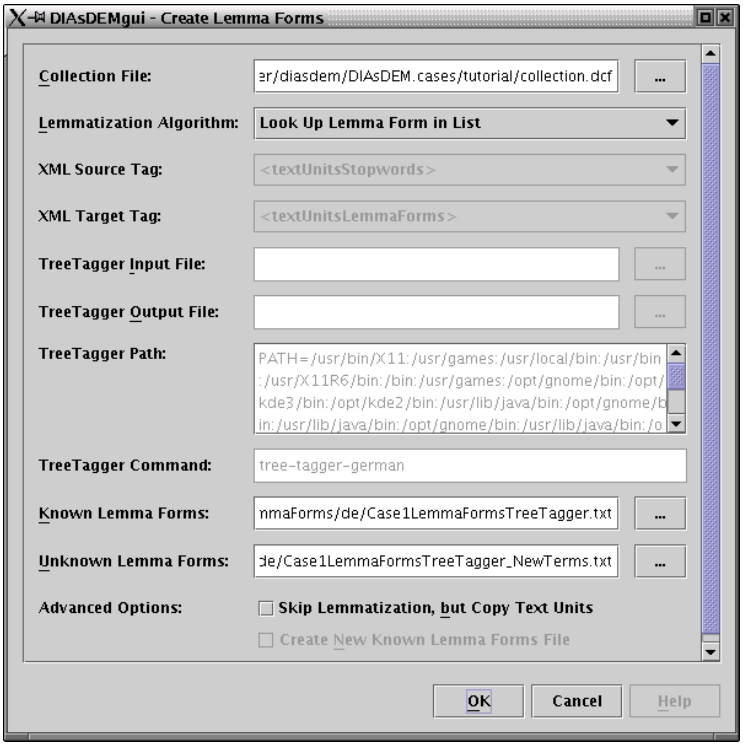

Figure 3.9: *Create Lemma Forms* window of DIAsDEM Workbench 2.0

```
    . </textUnitLemmaForms> ...
    <textUnitLemmaForms><namedEntityRef id="1019" /> , sein zur Geschäftsführerin
    bestellen . </textUnitLemmaForms> ...
    <textUnitLemmaForms>nicht eintragen : d Bekanntmachung d Gesellschaft
    erfolgen im Bundesanzeiger . </textUnitLemmaForms>
  </textUnitsLemmaForms>
</diasdemDocument>
```

Note, the list of known lemma forms had been created using TreeTagger. In the file shown above, the inflected verb form "ist" occuring in `<textUnitsStopwords>` is mapped onto its infinitive form "sein" in the corresponding `<textUnitLemmaForms>`. TreeTagger has been unable to determine the grammatical root form for terms whose lemma forms are suffixed by "_unknown" .

During the iterative clustering phase, text unit vectors are clustered based on similarity of their contents in order to discover semantic XML tags. Note that text unit vectors are created by mapping all elements of the section `<textUnitsLemmaForms>` onto vectors. Thereby, vector dimensions correspond to so-called text units descriptors which are

defined in a domain-specific thesaurus. Consequently, each thesaurus should only contain case-sensitive lemma forms that truely occur in the section `<textUnitsLemmaForms>`.

Create Lemma Forms: Summary

Module: *File → Create Lemma Forms*

Use Case: The user must pre-process all imported texts as part of the DIAsDEM KDD process for semantic tagging of domain-specific texts archives. Creating lemma forms is pre-processing phase 5 of 5.

Prerequisites: Each XML file must conform to the XML DTD `DiasdemDocument.dtd` that is described in section 4.1 on page 75. Specifically, it must contain the non-mandatory sections `<textUnits>`, `<textUnitsTokenized>`, `<textUnitsNamedEntities>` and `<textUnitsStopwords>`.

Result: Each intermediate XML file of the collection is extended by a new section `<textUnitsLemmaForms>`. Unless *Skip Lemmatization, but Copy Text Units* is enabled, elements `<textUnitLemmaForms>` will contain grammatical root forms of the corresponding elements in section `<textUnitStopwords>`. Any previously existing section `<textUnitsLemmaForms>` will be completely replaced by a new one.

Additionally, the DIAsDEMgui preferences *Default Collection File*, *Default Project Directory*, *Default TreeTagger Input File*, *Default TreeTagger Output File*, *Default Known Lemma Forms File* and *Default Unknown Lemma Forms File* are set and updated, respectively.

Remarks: Creating lemma forms or actively skipping lemmatization is a prerequisite for the iterative clustering phase.

Create Lemma Forms: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

Lemmatization Algorithm: If *Use TreeTagger to Determine Lemma Form* is enabled, the external part-of-speech tagger is employed. In this case, TreeTagger must have been successfully installed and the local environment variable `PATH` must include the TreeTagger subdirectories `/bin` and `/cmd` before DIAsDEM Workbench is started. If *Look Up Lemma Form in List* is enabled, a list of a priory known grammatical root forms is used for lemmatization.

TreeTagger Input File: Must be set if *Use TreeTagger to Determine Lemma Form* is enabled; valid local file name of new or existing file that will be replaced; this temporary file is created by DIAsDEM Workbench and includes text to be POS-tagged by TreeTagger; file extension: `.txt`; default value: DIAsDEMgui preference *Default TreeTagger Input File*

TreeTagger Output File: Must be set if *Use TreeTagger to Determine Lemma Form* is enabled; valid local file name of new or existing file that will be replaced; this temporary file is created by TreeTagger and includes the results of POS-tagging; file extension: `.txt`; default value: DIAsDEMgui preference *Default TreeTagger Output File*

Known Lemma Forms: Must be set if *Look Up Lemma Form in List* is enabled; valid local file name of existing file that contains terms along with their lemma forms in the format described in section 4.2.5 on page 83; file extension: `.txt`; default value: DIAsDEMgui preference *Default Known Lemma Forms File*

Unknown Lemma Forms: Must be set if *Look Up Lemma Form in List* is enabled; valid local file name of new or existing file that is created or extended by DIAsDEM Workbench ; includes terms occuring in the collection that are not listed in *Known Lemma Forms* as well as the context of their occurrence (i.e., the sentence); can be used to update *Known Lemma Forms*; format described in section 4.2.5 on page 83; file extension: `.txt`; default value: DIAsDEMgui preference *Default Unknown Lemma Forms File*

Advanced Options: If *Skip Lemmatization, but Copy Text Units* is enabled, creating lemma forms is completely skipped, but a new section `<textUnitsLemmaForms>` is created by copying the contents of `<textUnitsStopwords>`. This option enables users to quickly meet the prerequisites of the following clustering phase without having to create lemma forms. If *Create New Lemma Forms File* is enabled along with *Use TreeTagger to Determine Lemma Form*, all terms and the corresponding lemma forms determined by TreeTagger are saved for later usage as a file of *Known Lemma Forms*.

3.3 Iterative Clustering

3.3.1 Creating Word Statistics

During the clustering phase, DIAsDEM Workbench requires a controlled vocabulary in the form of a domain-specific thesaurus. Text units are mapped onto vectors whose dimensions correspond to thesaurus descriptors. Creating the word frequency statistics

for a collection is the first step in establishing or updating a thesaurus for subsequent use in clustering. Word frequency statistics give an insight into the specific word frequency distribution prevalent in a document collection. Based on word frequency statistics, an initial thesaurus can either be created or an existing thesaurus can be updated by adding, editing or removing terms of interest. Although there exists a prepared thesaurus for this case study, creating and inspecting word frequency statistics is described in this tutorial for reasons of completeness. Therefore, select *File* → *Create Word Statistics* and provide the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
XML Source Tag	`textUnitsLemmaForms`
Word Statistics File	`${PROJECT_HOME}/lemmaForms.dws`
Advanced Options	`Enabled: Export Original Texts in CSV Format`
	`Enabled: Export Words Statistics in CSV Format`
	`Enabled: Export Words Statistics in HTML Format`

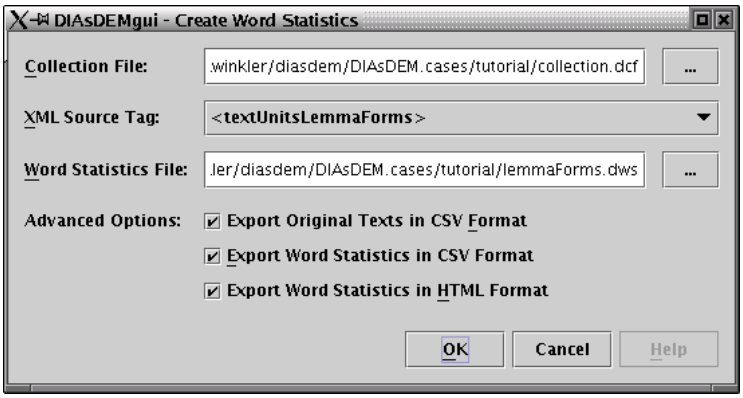

Figure 3.10: *Create Word Statistics* window of DIAsDEM Workbench 2.0

Click on *OK* to start the process of creating word statistics. Thereafter, check the contents of the directory `${PROJECT_HOME}` which now contains the DIAsDEM-specific word statistics file `lemmaForms.dws`. It can be displayed by *Tools* → *Word Statistics Viewer*. Additionally, this directory contains word frequency statistics files in CSV (`lemmaForms.csv`) and HTML (`lemmaForms.html`) format. Due to the settings of advanced options, all original texts of this collection have also been exported into two CSV files, `lemmaForms.orig.csv` and `lemmaForms.lemma.csv`. The former contains the file name of each document and its original text as stored in section `<textUnits>`. The latter file contains lemmatized texts (i.e., the contents of the section `<textUnitsLemmaForms>`) along with their file names. Both export files can be input to further text analysis and text mining activities employing third-party software.

Create Word Statistics: Summary

Module: *File → Create Word Statistics*

Use Case: The user wants to analyze the word frequency distribution prevalent in a document collection to get insight into the particularities of the specific vocabulary. Additionally, the user might want to create an initial thesaurus for the collection or may want to edit an existing thesaurus based on collection-specific term frequencies.

Prerequisites: Each XML file must conform to the XML DTD `DiasdemDocument.dtd` that is described in section 4.1 on page 75. Specifically and according to the parameter settings, XML files must contain the non-mandatory sections `<textUnitsTokenized>` or `<textUnitsLemmaForms>`.

Result: *Word Statistics File* contains all terms and their absolute frequencies that occur in the specified sections of XML documents in collection *Collection File*. Additionally, the DIAsDEMgui preferences *Default Collection File* and *Default Word Statistics File* are set and updated, respectively.

Create Word Statistics: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

XML Source Tag: If *textUnitsTokenized* is enabled, term frequencies are computed for the contents of section `<textUnitsTokenized>`. If *textUnitsLemmaForms* is enabled, term frequencies are instead computed for the contents of section `<textUnitsLemmaForms>`. The latter setting should be used in case of creating word frequency statistics for thesaurus establishment or update, because only lemmatized text units are mapped onto vectors for subsequent clustering. Hence, thesauri to be employed by DIAsDEM Workbench must only contain lemma forms of both descriptors and non-descriptors.

Word Statistics File: Valid local file name of new or existing file that is created or replaced by DIAsDEM Workbench; file extension: `.dws`; default value: DIAsDEMgui preference *Default Word Statistics File*

Advanced Options: If *Export Original Texts in CSV Format* is enabled, two CSV files are created in `${PROJECT_HOME}` that contain the file name and the textual contents of each document. If *Export Word Statistics in CSV Format* is enabled, a CSV file is created in `${PROJECT_HOME}` that contains all terms

and their respective absolute frequencies. If *Export Word Statistics in HTML Format* is enabled, an HTML file is created in `${PROJECT_HOME}` that contains terms and their absolute frequencies.

3.3.2 Viewing Word Statistics

Select *Tools* → *View Word Statistics* to display the previously created statistics file. Thereafter, click on *Open Statistics* and choose the word statistics file `${PROJECT_HOME}/lemmaForms.dws`. After entering the minimum term frequency of words to be displayed (e.g., 5), its contents are shown in the left panel as illustrated in Figure 3.11. Word statistics can either be sorted by decreasing frequency or by ascending term. To sort the list of terms, click the buttons *Sort by Freq.* and *Sort by Term*, respectively.

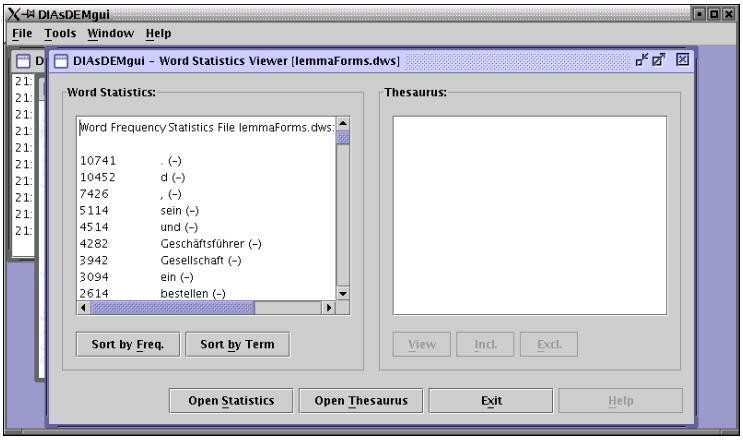

Figure 3.11: *Word Statistics Viewer* of DIAsDEM Workbench 2.0

Terms appearing in the left panel of *Word Statistics Viewer* can be compared with an existing DIAsDEM-specific thesaurus file. To proceed, click the *Open Thesaurus* button and choose the thesaurus file `${PARAMETER_HOME}/thesauri/de/CommercialRegisterThesaurus.dth`.

As illustrated in Figure 3.12, the entire thesaurus is initially displayed in the right panel. Each line corresponds to one thesaurus term that can either be a descriptor or a non-descriptor referencing an associated descriptor term. For example, the thesaurus entry `Ablehnung (D; Case2)` corresponds to descriptor ("D") term "Ablehnung" which is a valid text unit descriptor only in case study 2. Note again, valid text unit descriptors correspond to dimensions of text unit vectors that are subsequently clustered. According to thesaurus entry `<<person>> (D; Case1)`, the named entity type "Person" is a valid descriptor in case study 1. For each identified instance of named entity type "Person" in a text unit, the descriptor counter will be incremented. Finally, the thesaurus entry

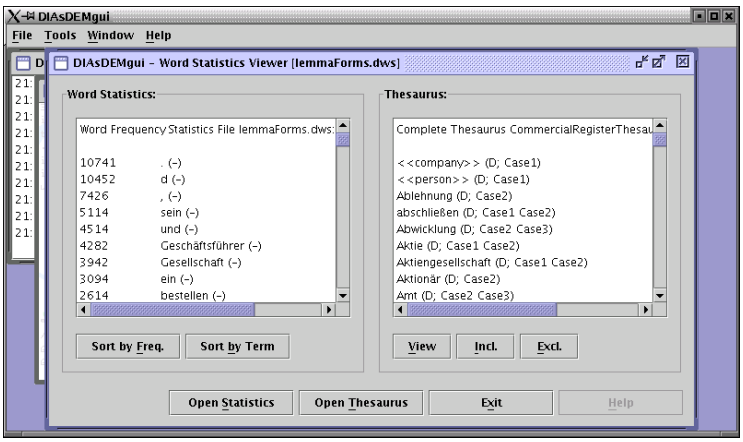

Figure 3.12: *Word Statistics Viewer* of DIAsDEM Workbench 2.0

`beginnen (N; Beginn)` states that the non-descriptor ("N") term "beginnen" is mapped onto its descriptor term "Beginn". If the term "beginnen" occurs in a text unit, the counter of its descriptor term "Beginn" will be incremented. However, this term is a valid descriptor only in case study 1 according to `Beginn (D; Case1)`.

Click the *Incl.* button in the right panel to filter word statistics terms that are also descriptor or non-descriptor thesaurus terms. Additionally, the term frequency in the current collection is shown for each thesaurus term in the right panel. In order to filter word statistics terms that are not contained in the thesaurus, click the *Excl.* button in the right panel. Frequently occuring and semantically important terms that are not listed in the thesaurus are candidates for thesaurus updates. In contrast, infrequently occuring terms could be removed from the thesaurus in order to reduce the dimensionality of text units vectors. However, do not remove important concepts such as "Tätigkeit" from the thesaurus that are descriptor terms for less important non-descriptors. Note, the frequent term `<<1001>>` is a placeholder for every first named entity in a text. Do not include named entity placeholders in the thesaurus, because they replace instances of various named entity types, e.g. "Person" or "Date".

After analyzing the statistics, you might consider to add the frequently occuring term "Bundesanzeiger" as a descriptor to the thesaurus. Moreover, the rather frequent term "Bauvorhaben" should be a non-descriptor term pointing to the important descriptor "Tätigkeit". Additionally, thesaurus term "Aktionär" should be removed from the thesaurus due to its infrequent occurrence in the collection. Thesaurus updates will be explained in the next section. Therefore, do not close the *Word Statistics Viewer* yet.

3.3.3 Editing Domain-Specific Thesauri

DIAsDEM Workbench includes two German thesauri in the subdirectories of directory `${PARAMETER_HOME}/thesauri/de`. They contain application-specific vocabularies for case studies related to Commercial Register entries and corporate news, respectively. Thesauri that should be employed in different application domains can be created by *Tools* → *Create Initial Thesaurus* as described in section 3.5.1. However, the remainder of this section focuses on the process of updating an existing, DIAsDEM-specific thesaurus by adding, editing and removing terms. Select *Tools* → *Thesaurus Editor*, click on *Open* and choose the thesaurus file `CommercialRegisterThesaurus.dth` in the directory `${PARAMETER_HOME}/thesauri/de/commercialRegister`.

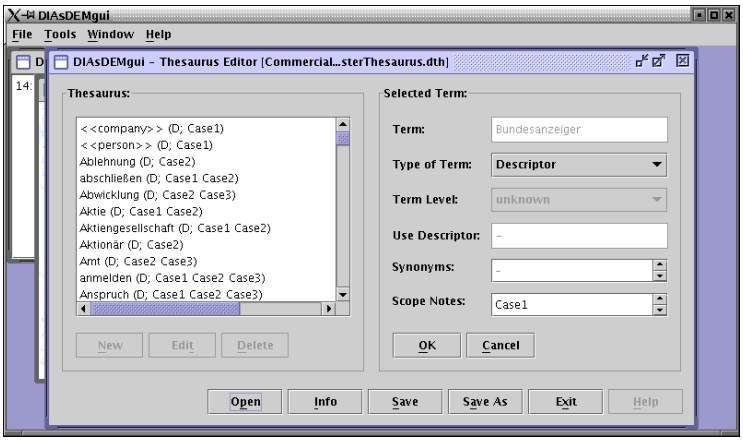

Figure 3.13: *Thesaurus Editor* of DIAsDEM Workbench 2.0

To add the first new term, click on *New* and enter "Bundesanzeiger" which is thereafter displayed in the right editor pane as the selected, editable thesaurus term. Similarly to many Windows applications, you might also use the system clipboard to transfer textual contents for example from *Word Statistics Viewer* to *Thesaurus Editor*. Using the mouse, select the term "Bundesanzeiger" in *Word Statistics Viewer*. The highlighted can be copied to the clipboard by the keyboard shortcut *CTRL-C*. Afterwards, the contents of the clipboard can be pasted into other documents by placing the cursor at the desired position and using the keyboard shortcut *CTRL-V*. Moreover, the keyboard shortcut *CTRL-X* can be used to cut (i.e., remove) selected text from the source document after copying it to the clipboard.

Each thesaurus term must either be a descriptor or a non-descriptor that references an associated descriptor term. Due to its frequent occurrence and semantic importance, the German word "Bundesanzeiger" should be a text unit vector dimension and must thus be a descriptor. Therefore, change the *Type of Term* from *unknown* to *Descriptor*.

The contents of the field *Scope Notes* can be used to filter valid descriptors during the process of creating text unit vectors. Hence, an established thesaurus can be employed in different case studies. The term "Bundesanzeiger" should be a valid descriptor in the first case study only. Thus, case-sensitively enter "Case1" in the field *Scope Notes*. Finally, the new term is added to the current thesaurus by clicking on *OK*. Otherwise, click the *Cancel* button to discard any modifications of the selected term. Figure 3.13 depicts the *Thesaurus Editor* before committing the insertion of the new term "Bundesanzeiger".

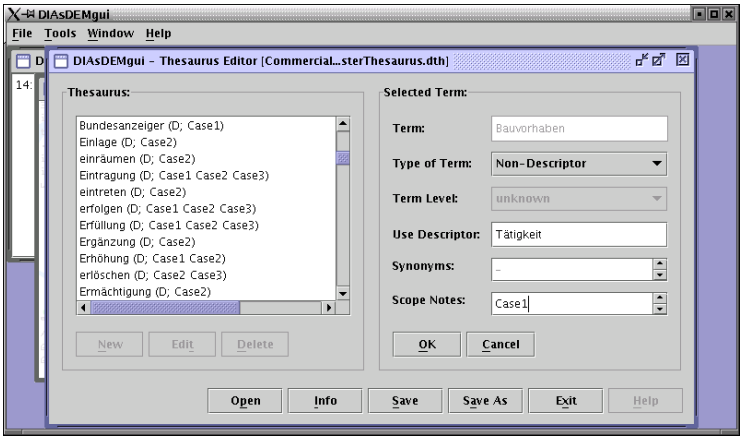

Figure 3.14: *Thesaurus Editor* of DIAsDEM Workbench 2.0

Insert the second new term "Bauvorhaben" into the thesaurus and change its *Type of Term* from *unknown* to *Non-Descriptor*. For each non-descriptor, an associated descriptor must be specified in the field *Use Descriptor*. Hence, type the descriptor term "Tätigkeit" in this field. Note, the *Use Descriptor* field must not contain other non-descriptors or terms that are not included in the current thesaurus. As before, input "Case1" in the field *Scope Notes* as well. The fields *Term Level* and *Synonyms* are not used in the current version of DIAsDEM Workbench. Figure 3.14 illustrates *Thesaurus Editor* before committing the insertion of the term "Bauvorhaben".

Existing terms such as "Bundesanzeiger" and "Bauvorhaben" can be modified by clicking the *Edit* button and afterwards entering the term of interest. Alternatively, a term in the left editor panel can be selected using the mouse. Thereafter, clicking on *Edit* will automatically open the corresponding term in the right editor panel for update. Analogously, terms can be removed from the thesaurus by clicking the *Delete* button. However, make sure not to delete descriptor terms that are referenced by remaining non-descriptors. Finally, remove the German term "Aktionär", because it occurs only once in the entire collection of Commercial Register entries.

Click the *Info* button and look at the brief thesaurus summary that lists the num-

ber of term, descriptors and non-descriptors in the current thesaurus. Keep in mind that the number of descriptors should be kept as low as possible, because DIAsDEM Workbench does not employ uncontrolled techniques for dimensionality reduction such as singular value decomposition. As a rule of thumb, the number of descriptors should not exceed 500 terms. Click the *Save* button in order to commit the entire thesaurus update. After saving, inspect the contents of the directory `${PARAMETER_HOME}/thesauri/de/commercialRegister`. In addition to updating the DIAsDEM-specific thesaurus file, saving a thesaurus always results in the creation of thesaurus files in CSV and HTML format in the same directory. The latter contains information about all thesaurus terms and an explicit mapping of descriptors onto their associated non-descriptors. Finally, click the respective *Exit* buttons to close *Thesaurus Editor* and *Word Statistics Viewer*.

3.3.4 Creating Text Unit Vectors in Iteration 1

Concerning the clustering of text unit vectors, DIAsDEM Workbench implements both a plug-in and a plug-out concept that enables the usage of various clustering algorithms. Users can either employ one of three built-in Weka [WF99] clustering algorithms (i.e., k-means, Cobweb and EM) or utilize algorithms supplied by external data mining applications. To ensure this flexibility, DIAsDEM Workbench is capable of exporting text unit vector files in three different formats. As the k-means clustering algorithm provided by the Java-based Weka data mining library is employed is this case study, vectors are exported in the Weka-specific ARFF format only. However, all three formats are briefly described in section 4.3.1 on page 85. To export text unit vectors for the first clustering iteration, select *File* → *Create Text Unit Vectors* and enter the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Iteration	`First Clustering Iteration`
Vector File Format	`ARFF: Weka Data Mining Project`
Text Unit Vectors File	`${PROJECT_HOME}/vectors1.arff`
Thesaurus File	`${PARAMETER_HOME}/thesauri/de/commercialRegister/CommercialRegisterThesaurus.dth`
Text Unit Descriptors	`Descriptors whose Scope Notes Contain String Case1`
Term Frequency	`Boolean Term Frequency`
Term Weighting	`TFxIDF Term Weighting`
Advanced Options	`Disabled: Create File for Discovery of Term Associations` `Enabled: Create Meta-Data File for Text Unit Vectors File`

Click the *OK* button to export text unit vectors according these parameters. In the first clustering iteration, each text unit in section `<textUnitsLemmaForms>` is mapped onto its vector representation. Let D be the set of descriptors. The dimensionality of text unit vectors corresponds to $|D| = 74$ descriptors in `CommercialRegisterThesaurus.dth` that contain the string "Case1" in their respective scope notes.

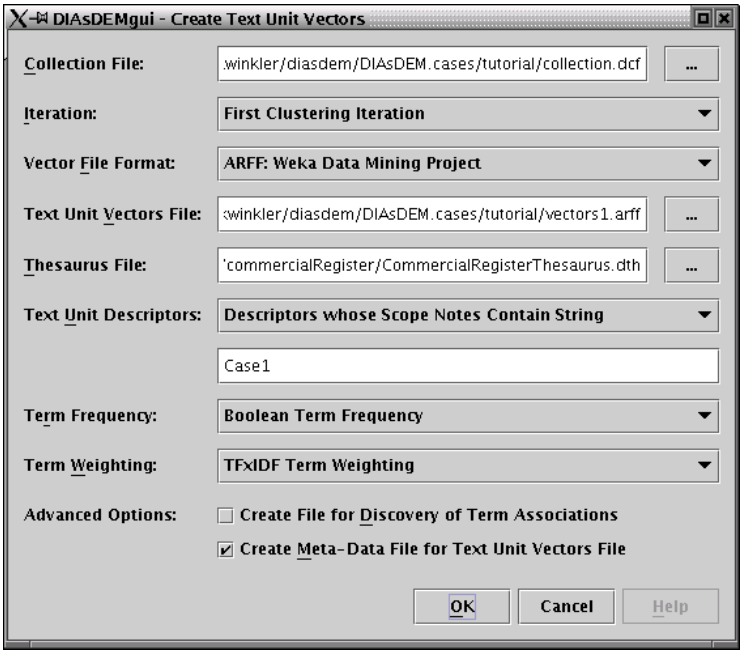

Figure 3.15: *Create Text Unit Vectors* window of DIAsDEM Workbench 2.0

The mapping of text units onto text unit vectors works as follows: Firstly, a boolean vector is created for each text unit. Each vector component $i = 1, \ldots, |D|$ represents the boolean term frequency of descriptor d_i in its text unit. Vector component i is 1 if descriptor d_i occurs in the corresponding text unit or 0 otherwise. Secondly, boolean vectors are weighted by multiplying each vector component i and the inverse document frequency of descriptor d_i. Let U be the set of text units in the collection and let $freq(d_i)$ be the absolute frequency of descriptor d_i in the same collection. The inverse document frequency of descriptor d_i is here defined as $log(|U|/freq(d_i))$. This weighting schema favors terms that occur in relatively few text units, because these terms have a higher discriminative power than terms occuring in almost all text units. To sum up, each vector component i represents the product of boolean term frequency of descriptor d_i in the corresponding sentence and inverse document frequency of descriptor d_i within the entire collection. Open the meta-data file `${PROJECT_HOME}/vectors1.arff.meta` that lists the term frequency as well as the inverse document frequency for each descriptor:

```
...
D1_Aktie = Aktie; Term Frequency = 38; Term Weight = 5.64144
D2_Gesellschafter = Gesellschafter; Term Frequency = 201; Term Weight = 3.97572
...
D74_Anspruch = Anspruch; Term Frequency = 9; Term Weight = 7.08180
```

Note that descriptor term "Aktie" and its associated non-descriptors such as "Namensaktie" occur 38 times in the collection of Commercial Register entries. Its term weight which here equals its inverse document frequency is greater than the term weight of "Gesellschafter", because "Aktie" occurs less frequently in this collection. According to the applied TFxIDF weighting schema, "Aktie" has a greater discriminative power than "Gesellschafter" due to its relatively infrequent occurrence in the collection.

The text unit vector file `${PROJECT_HOME}/vectors1.arff` contains all vectors to be clustered in the Weka-specific ARFF format [WF99]. ARFF-files include metadata about the relation and its attributes (i.e., their names and domains) as well as the actual data below `@data`. For example, the second vector corresponds to the following second text unit of file `${PROJECT_HOME}/xml/file10144.txt.xml`: "Persönlich haftende Gesellschafterin: AGE Glas Vertrieb GmbH, Sitz: Garbsen." The descriptor "Gesellschafter" in the form of its associated non-descriptor "Gesellschafterin" occurs in this text unit. Hence the second vector component represents a term weight greater than zero. However, neither the descriptor term "Aktie" nor a related non-descriptor occurs in this sentence. The first component of the second vector thus equals zero.

```
@relation 'DIAsDEM'
@attribute DocumentType string
@attribute Document string
@attribute TextUnit string
@attribute D1_Aktie real
@attribute D2_Gesellschafter real
...
@attribute D74_Anspruch real
@data
null,/file10144.txt.xml,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.42154,...
null,/file10144.txt.xml,2,0,3.97572,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...
...
```

The text unit vector file `${PROJECT_HOME}/vectors1.arff` is input to the first clustering iteration which is described in the next section.

Create Text Unit Vectors: Summary

Module: *File → Create Text Unit Vectors*

Use Case: The user wants to cluster pre-processed text units of imported texts as part of the DIAsDEM KDD process for semantic tagging of domain-specific texts archives. Creating text unit vectors precedes the clustering step.

Prerequisites: Each XML file must conform to the XML DTD `DiasdemDocument.dtd` that is described in section 4.1 on page 75. Specifically, it must contain the non-mandatory section `<textUnitsLemmaForms>` in the first clustering iteration and `<textUnitsTagged>` in subsequent clustering iterations, respectively.

Remarks: Creating and exporting text unit vectors is a prerequisite for the actual clustering phase.

Create Text Unit Vectors: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

Iteration: If *First Clustering Iteration* is enabled, text unit vectors are created for each text unit in sections `<textUnitsLemmaForms>`. If *Subsequent Clustering Iteration* is enabled, text unit vectors are created for text units in sections `<textUnitsTagged>` that have not been semantically named in a previous clustering iteration. These vectors have been assigned to qualitatively unacceptable clusters in the previous iteration as explained in section 1.

Vector File Format: Choice of vector file format as described in section 4.3.1 on page 85 between comma separated values (CSV file), fixed width values (TXT file) and the Weka-specific ARFF file format; default value: DIAsDEMgui preference *Default Vector File Format*

Text Unit Vectors File: Valid local file name of file to be created or replaced by DIAsDEM Workbench; file extension depends on choice of *Vector File Format*: `.csv`, `.txt` or `.arff`; default value: DIAsDEMgui preference *Default Text Unit Vectors File*

Thesaurus File: Valid local file name of existing DIAsDEM-specific thesaurus file as described in section 4.3.1 on page 85; file extension: `.dth`; default value: DIAsDEMgui preference *Default Thesaurus File*

Text Unit Descriptors: If *All Descriptors in Thesaurus* is enabled, all descriptor terms in *Thesaurus File* are valid. If *Descriptors whose Scope Note Contain String*

is enabled, only descriptor terms in *Thesaurus File* are valid whose scope notes contain the exact string entered below. If *Descriptors whose Scope Note Don't Contain String* is enabled, only descriptor terms in *Thesaurus File* are valid whose scope notes do not contain the exact string entered below.

Term Frequency: If *Raw Term Frequency* is enabled, the term frequency of valid descriptor d in text unit u equals the number of times d occurs in u. If *Boolean Term Frequency* is enabled, the term frequency of d in u is 1 if d occurs in u and 0 otherwise.

Term Weighting: If *No Term Weighting* is enabled, the term frequency of valid descriptor d in text unit u is not weighted at all. If *TFxIDF Term Weighting* is enabled, the term frequency of valid descriptor d in text unit u is multiplied by the inverse document frequency of d in the collection as defined above.

Advanced Options: If *Create File for Discovery of Term Associations* is enabled, an additional file named analogously to *Text Unit Vectors File* but suffixed `.assoc` is created. It can be used for discovery of association rules between descriptor terms in text units. If *Create Meta-Data File for Text Unit Vectors File* is enabled, an additional file named analogously to *Text Unit Vectors File* but suffixed `.meta` is created that contains mappings of abbreviated attribute names onto their respective unabbreviated descriptors along with their term frequencies and term weights.

3.3.5 Clustering Text Unit Vectors in Iteration 1

DIAsDEM Workbench supports the export of text unit vectors into different, mostly standardized file formats. Hence, various external clustering algorithms could be employed to group vectors based on their contents for subsequent semantic labeling. This plug-out concept has been successfully tested in case studies employing commercial data mining applications such as IBM Intelligent Miner for Data and SAS Enterprise Miner, respectively [WS01c, WS02c]. Additionally, DIAsDEM Workbench implements a plug-in concept that wraps the Java-based data mining library Weka. Along with various data pre-processing and machine learning algorithms, Weka contains three clustering algorithms (i.e., k-means, Cobweb and EM) that have been integrated into DIAsDEM Workbench. The discussion of these algorithms and their parameters is beyond the scope of this tutorial. See [WF99] for an excellent description of these algorithms, their parameters and their implementation in the open source Weka library. For moderate amounts of data, all three Weka algorithms should be capable of clustering text unit vectors without memory- or runtime-related problems. However, in this case study,

the simple k-means clustering algorithm will be employed only. To start the clustering process, select *File* → *Cluster Text Unit Vectors* and enter the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Text Unit Vectors File	`${PROJECT_HOME}/vectors1.arff`
Clustering Algorithm	`weka.clusterers.SimpleKMeans`
Clustering Mode	`Clustering Phase (Create New Clustering Model)`

Clustering Parameters	`1) Number of Clusters = 100`
	`2) Acuity =`
	`3) Cutoff =`
	`4) Max. Iterations =`
	`5) Random Number Seed =`
	`6) Min. Std. Deviation =`

Clustering Results File	`${PARAMETER_HOME}/results1.csv`
Text Unit Clusterer File	`${PARAMETER_HOME}/clusterer1.wskm`

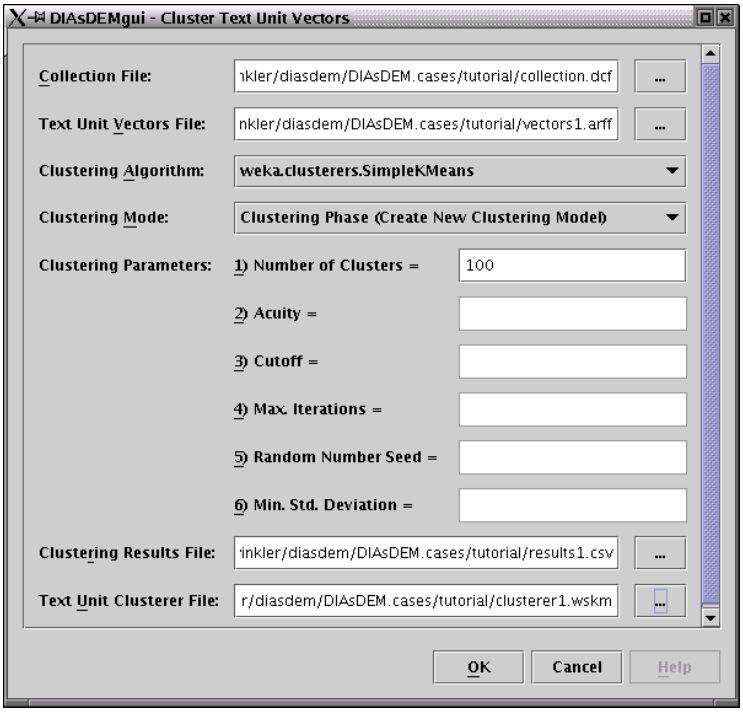

Figure 3.16: *Cluster Text Unit Vectors* window of DIAsDEM Workbench 2.0

Click the *OK* button to start the first clustering iteration. According to the parameters, the simple k-means algorithm is executed to create exactly $k = 100$ text unit vector

clusters some of whom may of course be empty. *Number of Clusters* is the only parameter of this algorithm, whereas *Acuity* and *Cutoff* are two parameters of the Cobweb clusterer. The EM algorithm can be parameterized by *Max. Iterations*, *Random Number Seed* and *Min. Std. Deviation*. All three algorithms require text unit vector files that conform to the Weka-specific ARFF-format. Again, see [WF99] for a detailed discussion of these parameters. The progress of clustering cannot be displayed due to the missing support of progress measurement in Weka. Figure 3.17 depicts an animated, so-called indeterminate progress bar that is shown while the clustering algorithm is being executed. Running Java 1.4.0 for Linux on a Notebook equipped with an 1.06 GHz Intel Mobile Celeron processor and 256 MB memory, clustering of 10,711 text unit vectors requires approx. 15 minutes.

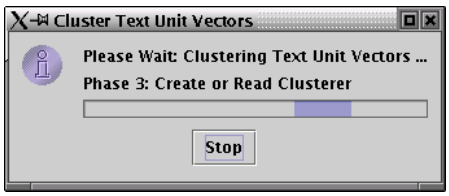

Figure 3.17: Indeterminate progress bar of task *Cluster Text Unit Vectors*

DIAsDEM Workbench post-processes proprietary output generated by Weka clusterers (e.g., `${PROJECT_HOME}/results1.csv.temp`). It creates clustering result files in CSV-format that can easily be processed by *File → Monitor Cluster Quality* and *File → Tag Text Units*, respectively. After clustering has finished, inspect the *Clustering Results File* `${PROJECT_HOME}/results1.csv`. Each line contains a file name within the collection, a text unit identified by its sequence number (i.e., the second attribute) and an associated cluster ID which is the third attribute. Consider for example the intermediate XML file `/file10144.txt.xml`: Its first text unit is assigned to cluster 2, whereas the second one is assigned to cluster 35. Note that cluster 2 also contains the first text unit of the file `/file11136.txt.xml`.

```
/file10144.txt.xml,1,2
/file10144.txt.xml,2,35
/file10144.txt.xml,3,68
/file10144.txt.xml,4,77
/file10144.txt.xml,5,71
/file11136.txt.xml,1,2
```

As described in section 3.3.6, the contents of text units clusters can be visualized by the module *File → Monitor Cluster Quality*. Note, the output file `${PROJECT_HOME}/clusterer1.wskm` contains a serialized instance of the Java class `weka.clusterers.`

`SimpleKMeans`. This so-called text unit clusterer can be employed to cluster text unit vectors during the application phase of the DIAsDEM framework as explained in section 1. In contrast to the clustering phase exemplified by this case study, *Text Unit Clusterer File* is an input file during the application phase. Running DIAsDEM Workbench in application mode can be simulated by applying `${PROJECT_HOME}/clusterer1.wskm` to the same text unit vectors in `${PROJECT_HOME}/vectors1.arff` using the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Text Unit Vectors File	`${PROJECT_HOME}/vectors1.arff`
Clustering Algorithm	`weka.clusterers.SimpleKMeans`
Clustering Mode	`Application Phase (Apply Existing Clustering Model)`

Clustering Parameters	`1) Number of Clusters =`
	`2) Acuity =`
	`3) Cutoff =`
	`4) Max. Iterations =`
	`5) Random Number Seed =`
	`6) Min. Std. Deviation =`

Clustering Results File	`${PROJECT_HOME}/results1.csv`
Text Unit Clusterer File	`${PROJECT_HOME}/clusterer1.wskm`

Compared to training a text unit clusterer, a significant runtime improvement can be noticed in application mode. When applying an existing clusterer to text unit vectors, the parameters of the algorithm cannot be altered due to obvious reasons. After clustering text unit vectors, monitoring the cluster quality is the next step in this case study (i.e., in clustering mode) as described in the next section. In contrast, clustering is directly followed by text unit tagging when DIAsDEM Workbench is running in application mode.

Cluster Text Unit Vectors: Summary

Module: *File → Cluster Text Unit Vectors*

Use Case: The user wants to cluster text unit vectors of imported texts as part of the DIAsDEM KDD process for semantic tagging of domain-specific texts archives.

Prerequisites: Vectors to be clustered in *Text Unit Vectors File* must conform to the Weka-specific ARFF file format exported by the module *File → Create Text Unit Vectors*. This file format is described in section 4.3.1 on page 85.

Result: In clustering mode, text unit vectors are clustered by the selected algorithm and the resulting clusterer is saved for subsequent usage in application mode. Given an existing text unit clusterer, text units can quickly be assigned to their respective clusters in application mode.

Additionally, the DIAsDEMgui preferences *Default Collection File*, *Default Text Unit Vectors File*, *Default Clustering Algorithm*, *Default Clustering Mode*, *Default Clustering Parameters*, *Default Clustering Results File* and *Default Text Unit Clusterer File* are set and updated, respectively.

Remarks: Instead of employing the Weka-based internal clustering algorithms, any other algorithm can also be used provided that its results can be exported or converted into a file format supported by DIAsDEM Workbench.

Cluster Text Unit Vectors: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

Text Unit Vectors File: Valid local file name of existing file; file extension: `.arff`; default value: DIAsDEMgui preference *Default Text Unit Vectors File*

Clustering Algorithm: One of three algorithms supported by the Java-based Weka library [WF99] must be selected: *weka.clusterers.SimpleKMeans*, *weka.clusterers.Cobweb* or *weka.clusterers.EM*.

Clustering Mode: If *Clustering Phase (Create New Clustering Model)* is enabled, a new text unit clusterer is trained according to the parameter settings and output as *Text Unit Clusterer File*. If *Application Phase (Apply Existing Clustering Model)* is enabled, the existing clusterer *Text Unit Clusterer File* is applied to the contents of *Text Unit Vectors File*.

Clustering Parameters: If *Clustering Phase (Create New Clustering Model)* is enabled, the selected algorithm can be parameterized [WF99]. *weka.clusterers.SimpleKMeans*: *Number of Clusters*; *weka.clusterers.Cobweb*: *Acuity* and *Cutoff*; *weka.clusterers.EM*: *Max. Iterations*, *Random Number Seed* and *Min. Std. Deviation*.

Clustering Results File: Valid local file name of file to be created or replaced by DIAsDEM Workbench; file extension: `.csv` default value: DIAsDEMgui preference *Default Clustering Results File*

Text Unit Clusterer File: Valid local file name of file to be created or replaced by DIAsDEM Workbench, if *Clustering Mode* is set to *Clustering Phase*; valid local file name of existing file, if *Clustering Mode* is set to *Application Phase*; file extension: `.wskm`; default value: DIAsDEMgui preference *Default Text Unit Clusterer File*

3.3.6 Monitoring Cluster Quality in Iteration 1

As described in section 1, the set of text unit clusters discovered during clustering must be analyzed to separate qualitatively "acceptable" clusters from "unacceptable" ones. Recall that members of the former are semi-automatically assigned a semantic label, whereas all text unit vectors assigned to qualitatively "unacceptable" clusters are re-clustered in the next iteration. A discussion of cluster quality criteria is beyond the scope of this tutorial. However, the applied cluster quality criteria are described in detail in [GSW01]. In DIAsDEM Workbench, the Cluster Quality Monitor computes descriptive statistics for clusters, visualizes cluster contents in HTML files and creates a default cluster label file which contains default semantic labels for qualitatively "acceptable" clusters only. Default cluster labels are composed of text unit descriptors that prevail in the respective clusters. Start the Cluster Quality Monitor by selecting *File → Monitor Cluster Quality*. Thereafter, submit the following parameters and click on *OK*.:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Iteration	`First Clustering Iteration (Text Units: Text)`
Result File Format	`CSV: Comma Separated Values`
Cluster Result File	`${PROJECT_HOME}/results1.csv`
Cluster Result Directory	`${PROJECT_HOME}/iteration1`
Cluster Label File	`${PROJECT_HOME}/labels1.dcl`
Max. Cluster ID	`100`
Thesaurus File	`${PARAMETER_HOME}/thesauri/de/commercialRegister/CommercialRegisterThesaurus.dth`
Text Unit Descriptors	`Descriptors whose Scope Notes Contain String Case1`

Cluster Quality Criteria	`1) Min. Cardinality = 50`
	`2) Max. Distinct Ratio = 0.75`
	`3) Min. Frequent Ratio = 0.25`

Advanced Options	`Disabled: Ignore First Line of Cluster Result File`
	`Enabled: Ignore Empty Clusters in Cluster Index HTML File`
	`Enabled: Launch Web Browser with Cluster Index HTML File`
	`Enabled: Launch Cluster Label Editor with Cluster Label File`

Although available, both *Iteration* options *First Clustering Iteration (Text Units: N-Grams)* and *Subsequent Clustering Iteration (Text Units: N-Grams)* are alpha features that cannot be discussed in this tutorial. Contact Henner Graubitz (graubitz@iti.cs.uni-magdeburg.de) for details about how to employ DIAsDEM Workbench for semantic tagging of n-grams such as 7-words or 2-sentences. In the current version, only text units containing regular textual contents (e.g., sentences) can be semantically annotated. Hence, select *First Clustering Iteration (Text Units: Text)* in the first iteration and *Subsequent Clustering Iteration (Text Units: Text)* in subsequent ones.

For obvious reasons, the settings of *Thesaurus File* and *Text Unit Descriptors* must exactly correspond to the parameters entered in the module *File → Create Text Unit*

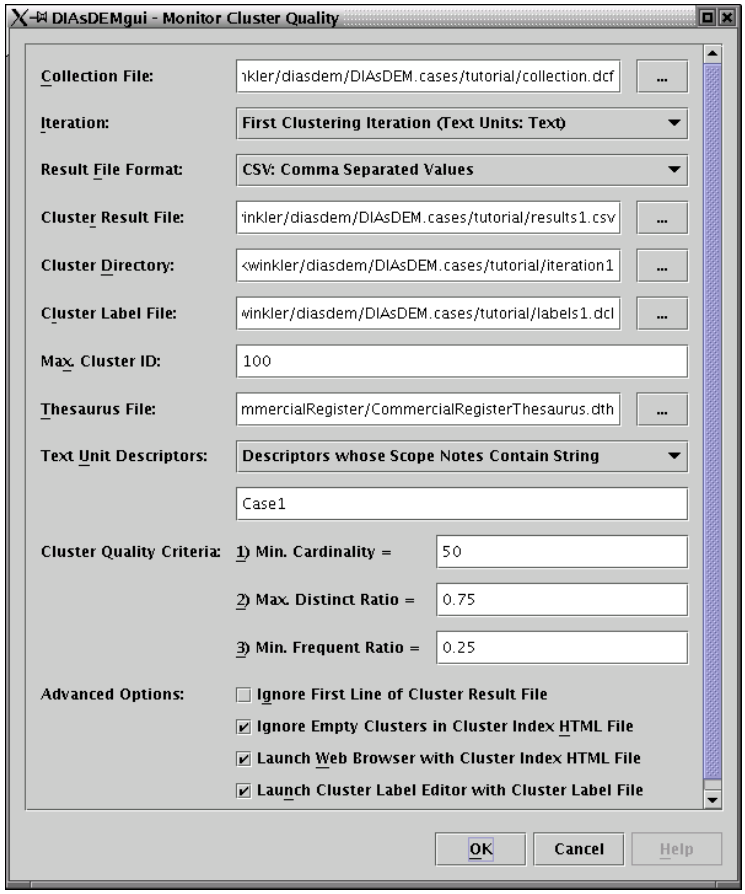

Figure 3.18: *Monitor Cluster Quality* window of DIAsDEM Workbench 2.0

Vectors in the current clustering iteration. *Max. Cluster ID* must equal the greatest integer that has been used as a cluster identifier in the current clustering run. In this first iteration, the Weka simple k-means algorithm was parameterized to discover $k = 100$ clusters. In contrast to other algorithms, the greatest cluster ID equals k in this Weka algorithm.

Please refer to [GSW01] for a detailed description of *Cluster Quality Criteria*. However, decreasing *Min. Cardinality* or *Min. Frequent Ratio* as well as increasing *Max. Distinct Ratio* tends to result in a greater number of qualitatively "acceptable" clusters which are automatically assigned a default label in *Cluster Result File*.

After monitoring cluster quality, your preferred Web browser pops up and displays the HTML file `${PROJECT_HOME}/iteration1/index.html`. As illustrated in Figure 3.19, it references all non-empty cluster files in the directory `${PROJECT_HOME}/iteration1`.

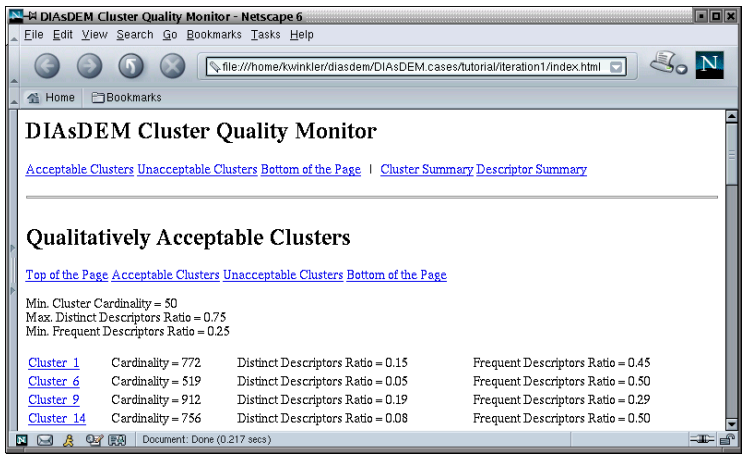

Figure 3.19: Cluster index file created by task *Monitor Cluster Quality*

If the browser cannot be launched, check the current settings in the *Applications* tab of the window *Tools* → *DIAsDEMgui Preferences*. Figure 3.20 depicts *Cluster Label Editor* that is also launched within DIAsDEM Workbench. This editor enables you to modify the automatically created cluster label file `${PROJECT_HOME}/labels1.dcl` by both altering default cluster labels and naming clusters having no default label. However, using *Cluster Label Editor* to customize the file `${PROJECT_HOME}/labels1.dcl` is described in section 3.3.7. Close *Cluster Label Editor* and the browser displaying `${PROJECT_HOME}/iteration1/index.html`.

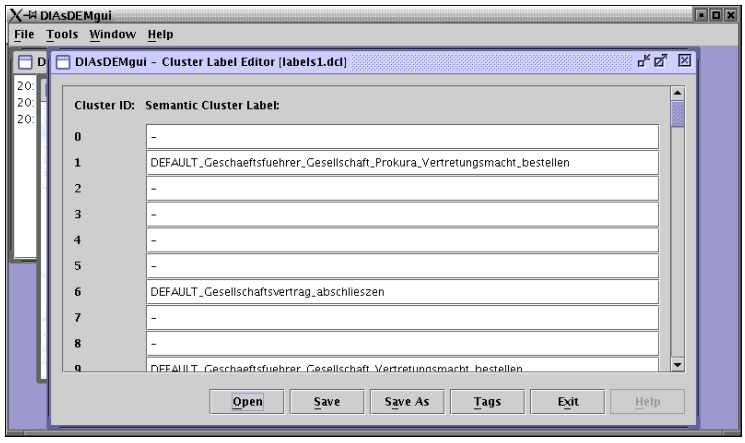

Figure 3.20: *Cluster Label Editor* of DIAsDEM Workbench 2.0

Monitor Cluster Quality: Summary

Module: *File → Monitor Cluster Quality*

Use Case: The user wants to separate qualitatively "acceptable" text unit vector clusters from "unacceptable" ones after clustering as part of the DIAsDEM KDD process for semantic tagging of domain-specific texts archives.

Prerequisites: Clustering results in *Cluster Result File* must conform either to the DIAsDEM-specific CSV or to the DIAsDEM-specific TXT file format which are described in section 4.3.3 on page 89.

Result: The contents of all text unit vector clusters are visualized as HTML files in *Clustering Directory*. Additionally, *Cluster Label File* contains default semantic labels for qualitatively "acceptable" clusters according to *Cluster Quality Criteria*.

Additionally, the DIAsDEMgui preferences *Default Collection File*, *Default Result File Format*, *Default Cluster Result File*, *Default Clustering Directory*, *Default Cluster Label File*, *Default Max. Cluster ID*, *Default Thesaurus File*, *Default Text Unit Descriptors*, *Default Min. Cardinality*, *Default Max. Distinct Ratio* and *Default Min. Frequent Ratio* are set and updated, respectively.

Remarks: This module must only be executed in the KDD phase of the DIAsDEM framework as explained in section 1. In this phase, monitoring cluster quality and thereby creating *Cluster Label File* is a prerequisite for subsequently tagging text units using *File → Tag Text Units*. *Thesaurus File* and *Text Unit Descriptors* must exactly correspond to the parameters entered in *File → Create Text Unit Vectors* in the current clustering iteration.

Monitor Cluster Quality: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

Iteration: *First Clustering Iteration (Text Units: Text)* has to be enabled in the first iteration, whereas *Subsequent Clustering Iteration (Text Units: Text)* must be enabled thereafter. Note, the options *First Clustering Iteration (Text Units: N-Grams)* and *Subsequent Clustering Iteration (Text Units: N-Grams)* are alpha features that should not be used without appropriate knowledge.

Result File Format: Choice of clustering result file format as described in section 4.3.3 on page 89 between comma separated values (CSV-file) and fixed width values (TXT-file); default value: DIAsDEMgui preference *Default Result File Format*

Cluster Result File: Valid local file name of existing file; file extension depends on choice of *Result File Format*: `.csv` or `.txt`; default value: DIAsDEMgui preference *Default Cluster Result File*

Clustering Directory: Valid local file name of existing directory or directory to be created by DIAsDEM Workbench; *Clustering Directory* should be empty; default value: DIAsDEMgui preference *Default Clustering Directory*

Cluster Label File: Valid local file name of file to be created or replaced by DIAsDEM Workbench; file extension: `.dcl`; default value: DIAsDEMgui preference *Default Cluster Label File*

Max. Cluster ID: integer greater than zero; corresponds to the greatest cluster identifier assigned by the clusterer in the current iteration; default value: DIAsDEMgui preference *Default Max. Cluster ID*

Thesaurus File: Valid local file name of existing DIAsDEM-specific thesaurus file as described in section 4.3.1 on page 85; file extension: `.dth`; default value: DIAsDEMgui preference *Default Thesaurus File*

Text Unit Descriptors: If *All Descriptors in Thesaurus* is enabled, all descriptor terms in *Thesaurus File* are valid. If *Descriptors whose Scope Note Contain String* is enabled, only descriptor terms in *Thesaurus File* are valid whose scope notes contain the exact string entered below. If *Descriptors whose Scope Note Don't Contain String* is enabled, only descriptor terms in *Thesaurus File* are valid whose scope notes do not contain the exact string entered below.

Cluster Quality Criteria: Three floating-point thresholds as described in [GSW01]; default values: DIAsDEMgui preferences *Default Min. Cardinality*, *Default Max. Distinct Ratio* and *Default Min. Frequent Ratio*

Advanced Options: If the first line of *Cluster Result File* contains attribute names, *Ignore First Line of Cluster Result File* must be enabled. If *Ignore Empty Clusters in Cluster Index HTML File* is enabled, the index file in *Clustering Directory* does not contain links to HTML files of empty clusters. If *Launch Web Browser with Cluster Index HTML File* is enabled, the browser specified in *Tools* → *DIAsDEMgui Preferences* is launched to display the index file of *Clustering Directory* after monitoring cluster

quality. Analogously, if *Launch Cluster Label Editor with Cluster Label File* is enabled, *Tools* → *Cluster Label Editor* is launched to edit *Cluster Label File*.

3.3.7 Editing Cluster Label Files in Iteration 1

After clustering text unit vectors and monitoring cluster quality, the default *Cluster Label File* should be manually inspected by a domain specialist. The objective of this task is to assign each qualitatively "acceptable" cluster an appropriate semantic label. Semantic cluster labels should provide a concise and content-based description of the respective text units, because they finally serve as elements of the XML document type definition to be derived. Text units whose vectors are assigned to semantically labeled clusters will be annotated by an XML tag that corresponds to the respective cluster label. All remaining text unit vectors are input to the clusterer in the next iteration.

Firstly, all qualitatively "acceptable" clusters should be checked that are automatically assigned a default cluster label. In this case study, default German cluster labels are manually replaced by English labels. Moreover, certain "acceptable" clusters may contain rather inhomogeneous text units according to the human sense of semantic similarity. For example, two opposite semantic concepts such as "to appoint" and "to dismiss" a managing director might be prevailing in a cluster. In these cases, default labels must be deleted in *Cluster Label File* in order to enforce the re-clustering of all corresponding text unit vectors in the next clustering iteration. Secondly, qualitatively "unacceptable" clusters should be inspected as well, because the current cluster quality criteria cannot capture all cases of semantic similarity. For example, a cluster might contain text units that belong to a common semantic concept, although there are no statistically prevailing text unit descriptors.

Select *Tools* → *Cluster Label Editor* and open the file `${PROJECT_HOME}/labels1.dcl` by clicking on *Open* and choosing this cluster label file. As depicted in Figure 3.19, it contains default cluster labels assigned during the first iteration. Additionally, open the index HTML file `${PROJECT_HOME}/iteration1/index.html`. For each cluster, there exists an HTML file in the directory `${PROJECT_HOME}/iteration1` which visualizes its textual contents and provides descriptive statistics of occuring text unit descriptors.

16 qualitatively "acceptable" clusters and 36 non-empty "unacceptable" ones have been automatically discovered by monitoring cluster quality. For example, Figures 3.21 and 3.22 illustrate the HTML file visualizing the qualitatively "acceptable" cluster 1. It has been automatically assigned the label "DEFAULT_Geschaeftsfuehrer_Gesellschaft_Prokura_Vertretungsmacht_bestellen". Note, this default label is the concatenation of all prevailing text unit descriptors within cluster 1 which are highlighted in Figure 3.22. Change the label of cluster 1 into its English equivalent "IfAppointmentOfManyManagingDirectors_JointPowerToRepresent" in *Cluster Label Editor* and click the *Save* button. Consider cluster 3 which is listed in the section "Qualitatively Unacceptable Clus-

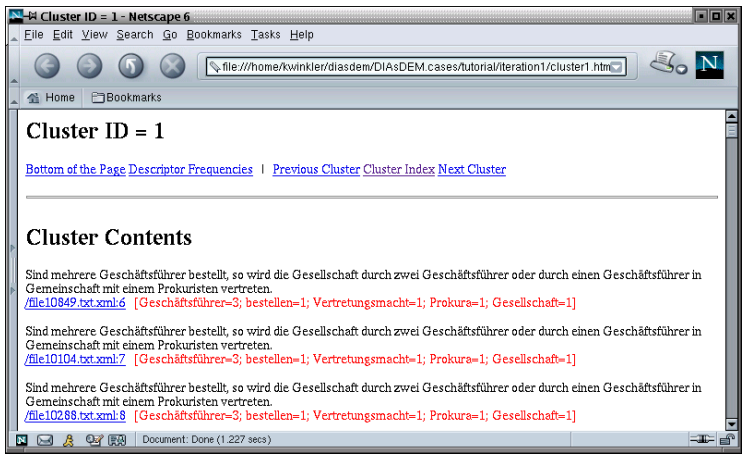

Figure 3.21: Top of HTML file visualizing the contents of cluster 1

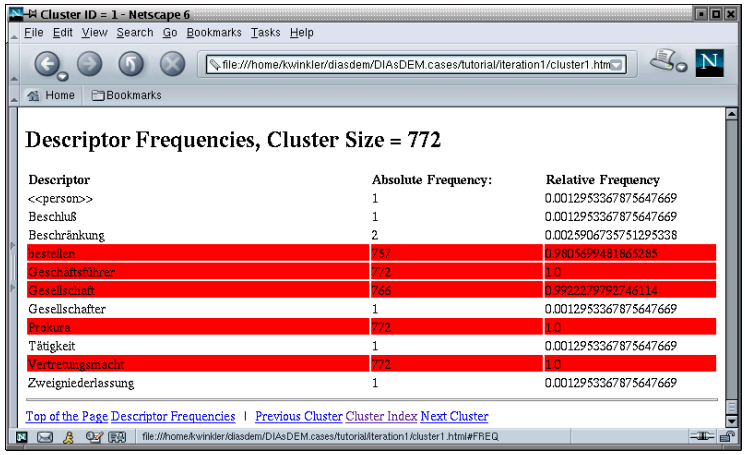

Figure 3.22: Descriptor frequencies at the bottom of cluster 1 HTML file

ters" of index file `${PROJECT_HOME}/iteration1/index.html`. The German concept "Tätigkeit" occurs in all members of cluster 3. Therefore, this cluster can be labeled with its English equivalent "PurposeOfCompany" as well. Inspect the remaining clusters and modify their respective semantic labels in *Cluster Label File* according to Table 3.2. Finally, close the Web browser, save the current *Cluster Label File* by clicking on *Save* and close *Cluster Label Editor* by clicking the *Exit* button.

Cluster ID	Semantic Cluster Label
1	IfAppointmentOfManyManagingDirectors_JointPowerToRepresent
3	PurposeOfCompany
6	ConclusionOfPartnershipAgreement
9	IfAppointmentOfOneManagingDirector_SolePowerToRepresent
14	AppointmentOfManagingDirector
22	SolePowerToRepresent_PowerToContractWithOneself
32	PublicationMediaOfCommercialRegisterEntries
38	SolePowerToRepresent_PowerToContractWithOneself
40	ShareCapital
48	ResolutionByShareholders_ChangeOfPlaceOfDomicile
52	NameOfMerchant
62	ChangeOfFirmName
72	NumberOfLimitedPartners
78	CommencementOfPartnership
90	PowerToContractWithOneself
91	DismissalOfManagingDirector

Table 3.2: Summary of semantic cluster labels in the first iteration

3.3.8 Tagging Text Units in Iteration 1

After clustering text unit vectors (i.e., creating *Cluster Result File*), monitoring cluster quality and manually editing the resulting *Cluster Label File*, all intermediate XML files of *Collection File* have to be updated. Specifically, each text unit whose vector has been input to the current clustering iteration should be annotated with the numerical identifier of the cluster it has been assigned to. Additionally, members of qualitatively "acceptable" and thus labeled clusters have to be annotated with the respective semantic label as specified in *Cluster Label File*. Tagging text units is a prerequisite for exporting text unit vectors in the next iteration as well as for tagging entire documents (i.e., creating semantically annotated XML documents) after the final clustering iteration. Hence, select *File* → *Tag Text Units* and type in the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Iteration	`First Clustering Iteration (Text Units: Text)`
Result File Format	`CSV: Comma Separated Values`
Cluster Result File	`${PROJECT_HOME}/results1.csv`
Cluster Label File	`${PROJECT_HOME}/labels1.dcl`
Advanced Options	`Disabled: Ignore First Line of Cluster Result File`

The parameters *Iteration*, *Result File Format* and *Cluster Result File* have been discussed in the previous section 3.3.6 in the context of monitoring cluster quality. Moreover, *Cluster Label File* corresponds to the file that has been created by the module *File*

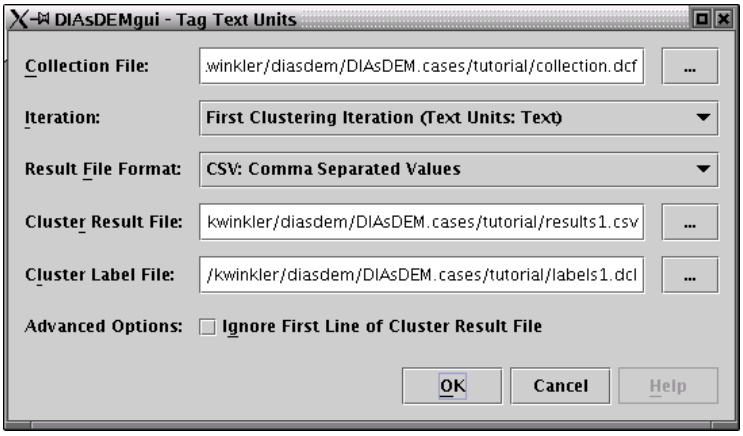

Figure 3.23: *Tag Text Units* window of DIAsDEM Workbench 2.0

→ *Monitor Cluster Quality*. Click the *OK* button to tag all text units of *Collection File* according to *Cluster Result File* and *Cluster Label File*. Thereafter, check the contents of the file `${PROJECT_HOME}/xml/file10780.txt.xml` which has been extended by the new section `<textUnitsTagged>` whose elements `<textUnitTagged>` mark up the same contents as the corresponding `<textUnitLemmaForms>`. However, each XML tag is extended by the attributes `clusterID` and `clusterName`. The values of the former contain a numerical cluster ID, whereas the latter attribute values either contain "-" for unlabeled clusters or the semantic label associated with the respective cluster. However, the current version of DIAsDEM Workbench does not save information about the iteration in which a text unit has been assigned to a semantically labeled cluster.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE diasdemDocument SYSTEM "DiasdemDocument.dtd">
<diasdemDocument> ...
  <textUnitsTagged> ...
    <textUnitTagged clusterID="40" clusterName="ShareCapital">
      Stammkapital : <namedEntityRef id="1001" /> .
    </textUnitTagged>
    <textUnitTagged clusterID="8" clusterName="-">
      Gesellschaft_mit_beschränkter_Haftung_unknown .
    </textUnitTagged> ...
    <textUnitTagged clusterID="14" clusterName="AppointmentOfManagingDirector">
      <namedEntityRef id="1019" /> , sein zur Geschäftsführerin bestellen .
    </textUnitTagged> ...
    <textUnitTagged clusterID="32"
    clusterName="PublicationMediaOfCommercialRegisterEntries">
      nicht eintragen : d Bekanntmachung d Gesellschaft erfolgen im Bundesanzeiger .
    </textUnitTagged>
```

```
  </textUnitsTagged>
</diasdemDocument>
```

Consider the first text unit shown in the file excerpt shown above which corresponds to the original sentence "Stammkapital: 50.000 DM." Its text unit vector has been assigned to cluster 40 that in turn has been labeled "ShareCapital". Thus, this sentence will subsequently be tagged as "`<ShareCapital>` Stammkapital: 50.000 DM. `</ShareCapital>`". In contrast, the second text unit has been assigned to cluster 8 which remains unlabeled after the first clustering iteration. Recall, this text unit vector will be input to the second clustering iteration. Finally, the text unit vector corresponding to the sentence "Marion Marcella Adolph geb. Priester, 22.03.1957, Offenbach, ist zur Geschäftsführerin bestellt." has been assigned to cluster 14 which has been semantically labeled "AppointmentOfManagingDirector". Note that all text unit vectors corresponding to annotated text units will not be re-clustered in the next iteration. Once a semantic label has been attached to a text unit, it cannot be changed anymore in the current version of DIAsDEM Workbench.

The execution of the second clustering iteration is concisely described in section 3.3.9. Thereafter, section 3.4.1 introduces the module *File → Tag Documents* which creates the final, semantically tagged XML documents according to the results of the case study.

Tag Text Units: Summary

Module: *File → Tag Text Units*

Use Case: The user wants to annotate text units in intermediate XML files according to the results of monitoring cluster quality as part of the DIAsDEM KDD process for semantic tagging of domain-specific texts archives.

Prerequisites: Each XML file must conform to the XML DTD `DiasdemDocument.dtd` that is described in section 4.1 on page 75. Specifically, it must contain the non-mandatory section `<textUnitsLemmaForms>` in the first clustering iteration and `<textUnitsTagged>` in subsequent clustering iterations, respectively. Moreover, clustering results in *Cluster Result File* must conform either to the DIAsDEM-specific CSV or to the DIAsDEM-specific TXT file format which are described in section 4.3.3 on page 89.

Result: In the first clustering iteration, a new section `<textUnitsTagged>` is created whose elements `<textUnitTagged>` mark up the same contents as the corresponding `<textUnitLemmaForms>`. In subsequent iterations, the section `<textUnitsTagged>` is updated. In both cases, all text units contained in *Cluster Result File* are annotated with the respective cluster ID and the corresponding label according to *Cluster Label File*.

Additionally, the DIAsDEMgui preferences *Default Collection File*, *Default Result File Format*, *Default Cluster Result File* and *Default Cluster Label File* are set and updated, respectively.

Remarks: Tagging text units is a prerequisite for the next clustering iteration and for finally executing the module *File → Tag Documents*. In the first clustering iteration, any previously existing section `<textUnitsTagged>` will be replaced by a new one.

Tag Text Units: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

Iteration: *First Clustering Iteration (Text Units: Text)* has to be enabled in the first iteration, whereas *Subsequent Clustering Iteration (Text Units: Text)* must be enabled thereafter. Note, the options *First Clustering Iteration (Text Units: N-Grams)* and *Subsequent Clustering Iteration (Text Units: N-Grams)* are alpha features that should not be used without appropriate knowledge.

Result File Format: Choice of clustering result file format as described in section 4.3.3 on page 89 between comma separated values (CSV-file) and fixed width values (TXT-file); default value: DIAsDEMgui preference *Default Result File Format*

Cluster Result File: Valid local file name of existing file; file extension depends on choice of *Result File Format*: `.csv` or `.txt`; default value: DIAsDEMgui preference *Default Cluster Result File*

Cluster Label File: Valid local file name of existing file created by DIAsDEM Workbench in *File → Monitor Cluster Quality* and possibly modified by *Tools → Cluster Label Editor*; file extension: `.dcl`; default value: DIAsDEMgui preference *Default Cluster Label File*

Advanced Options: If the first line of *Cluster Result File* contains attribute names, *Ignore First Line of Cluster Result File* must be enabled.

3.3.9 Summary of Clustering Iteration 2

All text units whose vectors have not been assigned to a qualitatively "acceptable" cluster in the first iteration are re-clustered in iteration 2. Consequently, text unit vectors that correspond to currently unlabeled sentences need to be exported and clustered. After

monitoring cluster quality and creating a new *Cluster Label File*, text units have to be tagged according to the results of the second iteration. These steps of the DIAsDEM KDD process have been discussed in detail in sections 3.3.4 through 3.3.8. Hence, this section only summarizes parameter settings and briefly explains particularities.

Firstly, text unit vectors that constitute the input data set to iteration 2 should be exported. Please select *File* → *Create Text Unit Vectors*, type in the following parameters and click on *OK*.

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Iteration	`Subsequent Clustering Iteration`
Vector File Format	`ARFF: Weka Data Mining Project`
Text Unit Vectors File	`${PROJECT_HOME}/vectors2.arff`
Thesaurus File	`${PARAMETER_HOME}/thesauri/de/commercialRegister/CommercialRegisterThesaurus.dth`
Text Unit Descriptors	`Descriptors whose Scope Notes Contain String Case1`
Term Frequency	`Boolean Term Frequency`
Term Weighting	`TFxIDF Term Weighting`
Advanced Options	`Disabled: Create File for Discovery of Term Associations` `Enabled: Create Meta-Data File for Text Unit Vectors File`

Compared to the first iteration, *Text Unit Vectors File* contains approx. one third of text unit vectors. Note additionally, collection-based term weights such as inverse document frequency are always computed on the basis of the remaining text unit vectors. For example, compare the different term weights for iteration 1 and 2 which are listed in the meta-data files `${PROJECT_HOME}/vectors1.arff.meta` and `vectors2.arff.meta`, respectively. In order to cluster all exported text unit vectors, select *File* → *Cluster Text Unit Vectors*, enter the following parameters and click on the *OK* button. In contrast to iteration 1, the maximum number of clusters to be discovered by k-means is 50.

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Text Unit Vectors File	`${PROJECT_HOME}/vectors2.arff`
Clustering Algorithm	`weka.clusterers.SimpleKMeans`
Clustering Mode	`Clustering Phase (Create New Clustering Model)`

Clustering Parameters	`1) Number of Clusters = 50`
	`2) Acuity =`
	`3) Cutoff =`
	`4) Max. Iterations =`
	`5) Random Number Seed =`
	`6) Min. Std. Deviation =`

Clustering Results File	`${PROJECT_HOME}/results2.csv`
Text Unit Clusterer File	`${PROJECT_HOME}/clusterer2.wskm`

Analogously to iteration 1, cluster text unit vectors is followed by monitoring cluster quality according to the DIAsDEM cluster quality criteria. Hence, select *File* → *Monitor*

Cluster Quality, submit the following parameters and click on *OK*. Compared with the first iteration, the cluster cardinality threshold has been decreased from 50 to 25.

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Iteration	`Subsequent Clustering Iteration (Text Units: Text)`
Result File Format	`CSV: Comma Separated Values`
Cluster Result File	`${PROJECT_HOME}/results2.csv`
Cluster Result Directory	`${PROJECT_HOME}/iteration2`
Cluster Label File	`${PROJECT_HOME}/labels2.dcl`
Max. Cluster ID	`50`
Thesaurus File	`${PARAMETER_HOME}/thesauri/de/commercialRegister/CommercialRegisterThesaurus.dth`
Text Unit Descriptors	`Descriptors whose Scope Notes Contain String Case1`

Cluster Quality Criteria	`1) Min. Cardinality = 25`
	`2) Max. Distinct Ratio = 0.75`
	`3) Min. Frequent Ratio = 0.25`

Advanced Options	`Disabled: Ignore First Line of Cluster Result File`
	`Enabled: Ignore Empty Clusters in Cluster Index HTML File`
	`Enabled: Launch Web Browser with Cluster Index HTML File`
	`Enabled: Launch Cluster Label Editor with Cluster Label File`

After monitoring cluster quality, the Web browser pops up and displays the HTML file `${PROJECT_HOME}/iteration2/index.html` that references all non-empty cluster files in `${PROJECT_HOME}/iteration2`. Additionally, *Cluster Label Editor* is launched within DIAsDEM Workbench. DIAsDEM Workbench has automatically discovered three qualitatively "acceptable" clusters only. Please have a look at all clusters and modify `${PROJECT_HOME}/labels2.dcl` in *Cluster Label Editor* according to Table 3.3.

After editing `${PROJECT_HOME}/labels2.dcl`, select *File* → *Tag Text Units*, type in the following parameters and click the *OK* button to annotate text units accordingly:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Iteration	`Subsequent Clustering Iteration (Text Units: Text)`
Result File Format	`CSV: Comma Separated Values`
Cluster Result File	`${PROJECT_HOME}/results2.csv`
Cluster Label File	`${PROJECT_HOME}/labels2.dcl`
Advanced Options	`Disabled: Ignore First Line of Cluster Result File`

Open the file `${PROJECT_HOME}/xml/file10780.txt.xml` and consider the original sentence "Gesellschaft mit beschränkter Haftung." which has not been tagged in iteration 1. However, the corresponding text unit vector has been assigned to the qualitatively "acceptable" cluster 10 in the second clustering iteration. Consequently, "Gesellschaft mit beschränkter Haftung." has been annotated with the semantic label "LimitedLiabilityCompany" of cluster 10. Note again that annotations and cluster IDs assigned in

Cluster ID	Semantic Cluster Label
3	JointStockCompany
9	SolePowerToRepresentCanBeGranted
10	LimitedLiabilityCompany
11	FullyLiablePartner
14	LimitedPartnership
18	PurposeOfCompany
19	FullyLiablePartner
25	PurposeOfCompany
26	ConclusionAndModificationOfPartnershipAgreement
27	PowerToRepresent_ManagingBoard
34	ConfermentOfProkura
36	PurposeOfCompany
37	SolePowerToRepresent
41	AppointmentOfManagingDirector

Table 3.3: Summary of semantic cluster labels in the second iteration

the first iteration remain constant. For example, the original sentence "Stammkapital: 50.000 DM." is still assigned to the first iteration cluster 40 labeled "ShareCapital".

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE diasdemDocument SYSTEM "DiasdemDocument.dtd">
<diasdemDocument> ...
  <textUnitsTagged> ...
    <textUnitTagged clusterID="40" clusterName="ShareCapital">
      Stammkapital : <namedEntityRef id="1001" /> .
    </textUnitTagged>
    <textUnitTagged clusterID="10" clusterName="LimitedLiabilityCompany">
      Gesellschaft_mit_beschränkter_Haftung_unknown .
    </textUnitTagged> ...
    <textUnitTagged clusterID="14" clusterName="AppointmentOfManagingDirector">
      <namedEntityRef id="1019" /> , sein zur Geschäftsführerin bestellen .
    </textUnitTagged> ...
    <textUnitTagged clusterID="32"
    clusterName="PublicationMediaOfCommercialRegisterEntries">
      nicht eintragen : d Bekanntmachung d Gesellschaft erfolgen im Bundesanzeiger .
    </textUnitTagged>
  </textUnitsTagged>
</diasdemDocument>
```

In this case study, only two clustering iterations are performed in order to exemplify the interactive and iterative DIAsDEM KDD process. When applying the DIAsDEM KDD process to real document archives, the iterative clustering should be continued until further qualitatively "acceptable" clusters cannot be discovered.

3.4 XML Tagging of Texts

3.4.1 Tagging Documents

After finishing the final clustering iteration, text documents have to be converted into an archive of semantically tagged XML documents in order to reach the objectives of the DIAsDEM framework. To that purpose, an XML document type definition needs to be derived firstly. It concisely describes frequently occuring, collection-specific semantic concepts in the form of DTD elements which can be either XML tags or attributes of XML tags. The latter correspond to named entity types whose instances exceed a relative frequency threshold within all text units annotated with the respective tag. Secondly, XML documents are created by assembling both tagged (i.e., text units whose vectors have been assigned to a semantically labeled cluster) and untagged text units in the order of their occurrence. Besides the derived XML document type definition and the generated text unit cluster for subsequent batch processing, semantically tagged XML documents constitute the main output of the DIAsDEM KDD process. To proceed with the last but one task of this case study, select *File* → *Tag Documents* and enter the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Unstructured DTD File	`${PROJECT_HOME}/unstructured.dud`
DTD Root Element	`CommercialRegisterEntry`
Min. Attribute Support	`0.1`
Random Sample File	`${PROJECT_HOME}/sample5percent.dts`
Random Sample Size	`0.05`
Text Units	`Text`
Advanced Options	`Enabled: Create CSV-File Containing Tag-by-Document-Matrix`
	`Disabled: Create Log Files for Subsequent Analysis with WUM`
	`Enabled: Create HTML File with Random Sample of XML Documents`

Note that *Unstructured DTD File* is a DIAsDEM-specific file that contains meta-data about the XML DTD to be derived for internal usage. According to *DTD Root Element*, the term `CommercialRegisterEntry` will be the root element of the XML document type definition. Hence, each XML document to be created will start with the same root tag as well. Due to *Min. Attribute Support*, named entity type e such as "Date" only qualifies as an attribute of XML tag t, if instances of e (e.g., "2003-03-31" and "2003-04-01") occur in at least 10% of all text units annotated with t. Furthermore, the module *File* → *Tag Documents* draws a 5% random sample of all text units in the current collection. This sample of text units is saved in the DIAsDEM-specific *Random Sample File* for subsequent evaluation of tagging quality.

The option *N-Grams* of parameter *Text Units* corresponds to an alpha feature that cannot be discussed in this tutorial. Again, contact Henner Graubitz (graubitz@iti.cs.uni-magdeburg.de) for details about how to employ DIAsDEM Workbench for semantic

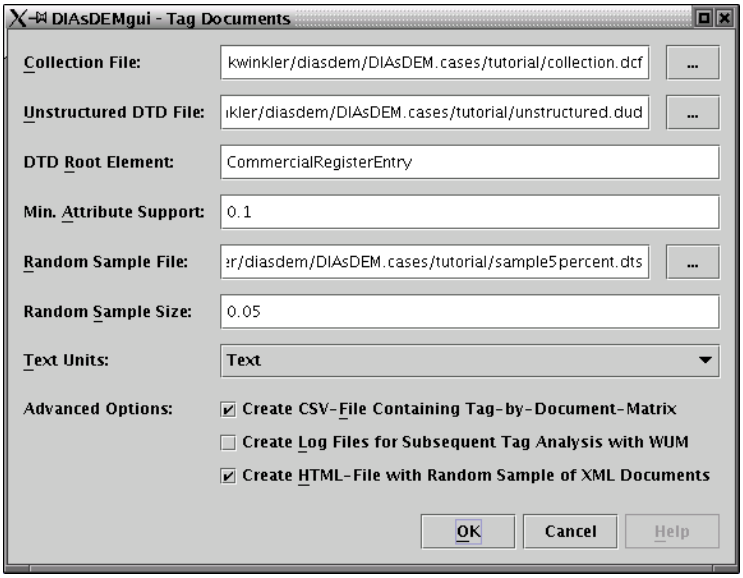

Figure 3.24: *Tag Documents* window of DIAsDEM Workbench 2.0

tagging of n-grams. In the current version of DIAsDEM Workbench, only text units containing normal textual content can be annotated. Hence, always select the parameter option *Text*. Click the *OK* button to start the semantic tagging of documents. For each intermediate XML document in the directory `${PROJECT_HOME}/xml`, a new file suffixed `.tagged.xml` is created which contains semantically annotated content of the corresponding text. The XML document `${PROJECT_HOME}/xml/file10780.txt.tagged.xml` is partly depicted below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE CommercialRegisterEntry SYSTEM 'CommercialRegisterEntry.dtd'>
<CommercialRegisterEntry>
  <PurposeOfCompany> Der Handel mit Waren aller Art sowie Import und Export.
  </PurposeOfCompany> <PurposeOfCompany> Der Dienstleistungsbereich bezieht
  sich auf Vermittlung, Beratung und Schulungen. </PurposeOfCompany>
  <ShareCapital AmoutOfMoney="50000 DM"> Stammkapital: 50.000 DM. </ShareCapital>
  <LimitedLiabilityCompany> Gesellschaft mit beschränkter Haftung.
  </LimitedLiabilityCompany> ... <SolePowerToRepresentCanBeGranted>
  Einzelvertretungsbefugnis kann erteilt werden. </SolePowerToRepresentCanBeGranted>
  <AppointmentOfManagingDirector Person="1019; Adolph; Marion Marcella; null; null;
  22.03.1957; Priester; Offenbach"> Marion Marcella Adolph geb. Priester, 22.03.1957,
  Offenbach, ist zur Geschäftsführerin bestellt. </AppointmentOfManagingDirector>
  <PowerToContractWithOneself> Sie ist befugt, Rechtsgeschäfte mit sich selbst
  oder mit sich als Vertreter Dritter abzuschließen. </PowerToContractWithOneself>
  <PublicationMediaOfCommercialRegisterEntries> Nicht eingetragen: Die
```

```
  Bekanntmachungen der Gesellschaft erfolgen im Bundesanzeiger.
  </PublicationMediaOfCommercialRegisterEntries>
</CommercialRegisterEntry>
```

The current version of DIAsDEM Workbench derives an unstructured, rather preliminary XML document type definition which simply enumerates valid DTD elements (i.e., XML tags) and attributes associated with XML tags. Currently, neither the discovery of nested XML tags nor the identification of frequently occuring sequences of XML tags within annotated documents are supported by DIAsDEM Workbench. Open the DTD file `${PROJECT_HOME}/xml/CommercialRegisterEntry.dtd` whose file name always matches *DTD Root Element*.

In this document type definition, the listing of unordered DTD elements is followed by definitions of all valid XML tags that can occur anywhere within textual content. Thereafter, valid attributes of XML tags are defined as well. For example, the XML tag `AppointmentOfManagingDirector` has two optional attributes `Date` and `Person`. Due to *Min. Attribute Support*, instances of named entity types "Date" and "Person" occur in at least 10% of all text units annotated with `AppointmentOfManagingDirector`. Note that attributes of XML tags cannot be semantically named in the current release of DIAsDEM Workbench.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT CommercialRegisterEntry ( #PCDATA
 | AppointmentOfManagingDirector | ChangeOfFirmName | ChangeOfPlaceOfDomicile
 | CommencementOfPartnership | ConclusionAndModificationOfPartnershipAgreement
 | ConclusionOfPartnershipAgreement | ConfermentOfProkura | ...
 | PurposeOfCompany | ResolutionByShareholders_ChangeOfPlaceOfDomicile
 | ShareCapital | ... | SolePowerToRepresent_PowerToContractWithOneself
)* >
<!ELEMENT AppointmentOfManagingDirector (#PCDATA)>
<!ELEMENT ChangeOfFirmName (#PCDATA)>
<!ELEMENT ChangeOfPlaceOfDomicile (#PCDATA)>
<!ELEMENT CommencementOfPartnership (#PCDATA)> ...
<!ELEMENT SolePowerToRepresent_PowerToContractWithOneself (#PCDATA)>
<!ATTLIST AppointmentOfManagingDirector Date CDATA #IMPLIED>
<!ATTLIST AppointmentOfManagingDirector Person CDATA #IMPLIED>
<!ATTLIST ChangeOfFirmName Company CDATA #IMPLIED>
<!ATTLIST ChangeOfPlaceOfDomicile Date CDATA #IMPLIED> ...
<!ATTLIST ShareCapital AmoutOfMoney CDATA #IMPLIED>
```

Consider two files that have been created due to *Advanced Options*. In contrast to *Random Sample File* containing a random sample of text units, `${PROJECT_HOME}/sample-5percent.html` references a random sample of XML documents. This HTML file is always located in the same directory as *Random Sample File* and is named in accordance

with *Random Sample File* as well. Using a browser capable of appropriately displaying XML files, `${PROJECT_HOME}/sample5percent.html` can be used to inspect a sample of final, semantically tagged XML documents as well as the corresponding intermediate XML files. Open the second additionally created file `${PROJECT_HOME}/xml/TagByDocumentMatrix.csv` which is partly shown below. Each line contains a relational representation of semantic XML tags that occur in a certain text file. As shown above, the XML tag `AppointmentOfManagingDirector` for example occurs in the XML file created from source document `/.../file10780.txt`, whereas the tag `ChangeOfFirmName` does not occur in this file. In the current version of DIAsDEM Workbench, the attribute `DocumentID` always has the `null` value.

```
DocumentFileName,DocumentID,AppointmentOfManagingDirector,ChangeOfFirmName,...
/home/.../commercialRegister1/file10144.txt,null,0,0,0,1,0,0,0,0,1,0,0,0,0,...
/home/.../commercialRegister1/file11136.txt,null,0,0,0,1,0,0,0,0,1,0,0,0,0,...
/home/.../commercialRegister1/file10318.txt,null,0,0,0,1,0,0,0,0,1,0,0,0,0,...
/home/.../commercialRegister1/file10780.txt,null,1,0,0,0,1,0,0,0,0,1,1,0,1,...
```

Tag Documents: Summary

Module: *File → Tag Documents*

Use Case: The user wants to create semantically tagged XML documents from intermediate XML files as part of the DIAsDEM KDD process for semantic tagging of domain-specific texts archives.

Prerequisites: Each intermediate XML file must conform to `DiasdemDocument.dtd` that is described in section 4.1 on page 75. Specifically, it must contain the non-mandatory sections `<textUnits>` and `<textUnitsTagged>`. Elements of the latter must have been processed by the module *File → Tag Text Units* at least once.

Result: A collection-specific XML document type definition is firstly derived that enumerates valid XML tags and their attributes. For each intermediate XML file, a semantically annotated XML file is thereafter created and saved in the same directory. File names of result XML files are suffixed `.tagged.xml`.

Additionally, the DIAsDEMgui preferences *Default Collection File*, *Default Unstructured DTD File*, *Default DTD Root Element*, *Default Min. Attribute Support*, *Default Random Sample File* and *Default Random Sample Size* are set and updated, respectively.

Remarks: After tagging result documents, only one task remains to be done: The quality of semantic tags should be evaluated using the module *Tools* → *Tagging Quality Evaluation*.

Tag Text Units: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: DIAsDEMgui preference *Default Collection File*

Unstructured DTD File: Valid local file name of file to be created or replaced by DIAsDEM Workbench; file extension: `.dud`; default value: DIAsDEMgui preference *Default Unstructured DTD File*

DTD Root Element: ISO-8859-1 encoded string without blank spaces; default value: DIAsDEMgui preference *Default DTD Root Element*

Min. Attribute Support: floating point threshold in the interval $[0; 1]$: named entity type e only qualifies as attribute of XML tag t, if instances of e occur in at least the specified portion of all text units annotated with t; default value: DIAsDEMgui preference *Default Min. Attribute Support*

Random Sample File: Valid local file name of file to be created or replaced by DIAsDEM Workbench; file extension: `.dts`; default value: DIAsDEMgui preference *Default Random Sample File*

Random Sample Size: floating point number in the interval $[0; 1]$ which corresponds to the portion of text units to be randomly drawn for quality evaluation; default value: DIAsDEMgui preference *Default Random Sample Size*

Text Units: option *Text* has to be enabled; option *N-Grams* corresponds to an alpha feature that should not be used without appropriate knowledge

Advanced Options: If *Create CSV-File Containing Tag-by-Document-Matrix* is enabled, a file `TagByDocumentMatrix.csv` is created in the collection directory that contains a relational mapping of source file names onto discovered XML tags. If *Create Log File for Subsequent Tag Analysis with WUM* is enabled, all sequences of XML tags in result files are exported into a log file for subsequent sequence mining and association rules discovery using WUM: The Web Utilization Miner. If *Create HTML-File with Random Sample of XML Documents* is enabled, a file named according to *Random Sample File* but suffixed `.html` is created which references a portion of finally tagged XML documents equal to *Random Sample Size* as well as their corresponding intermediate XML files.

3.4.2 Evaluating the Tagging Quality

Finally, the quality of semi-automatically created semantic XML mark up has to be evaluated. Due to the absence of pre-tagged documents, a random sample of all text units (i.e., both tagged and untagged ones) must be drawn for quality assessment. Thereafter, a domain specialist should verify the annotations of randomly chosen text units with respect to the following two error types:

- *Error type I (false positive):* A text unit is annotated with a wrong XML tag, i.e. the tag does not properly reflect the content of the text unit.
- *Error type II (false negative):* A text unit is not annotated at all, although there exists an XML tag in the derived DTD reflecting the content of the text unit.

In contrast to this quality assessment of semantic annotations, the accuracy of the DIAsDEM Named Entity Extractor cannot be evaluated in the current release of DIAsDEM Workbench. However, start the quality evaluation by selecting *Tools* → *Tagging Quality Evaluation*, clicking the *Start* button and choosing the following four parameter files one after the other:

Parameter	Value
Existing Text Unit Sample File	`${PROJECT_HOME}/sample5percent.dts`
New or Existing File of Evaluated Text Units	`${PROJECT_HOME}/evaluatedTextUnits.det`
Text Unit Sample File to be Created for Next Evaluation	`${PROJECT_HOME}/sample5percentB.dts`
Existing Unstructured DTD File	`${PROJECT_HOME}/unstructured.dud`

Both parameter files *Existing Text Unit Sample File* and *Existing Unstructured DTD File* have been created by the module *File* → *Tag Documents*. The contents of these files are concisely described in section 4.4.1. Evaluating tagging quality can be a lengthy task even for rather small text units samples. Hence, DIAsDEM Workbench supports the assessment of tagging quality in multiple sessions. In the first assessment session, a new file *New or Existing File of Evaluated Text Units* is created. It contains the domain specialist's decision for each text unit as well as the text unit itself. After clicking the *Stop* button, text units that remain to be evaluated in subsequent sessions are copied into *Text Unit Sample File to be Created for Next Evaluation*. In the next evaluation session, this file *Text Unit Sample File to be Created for Next Evaluation* must be chosen as *Existing Text Unit Sample File*.

Figure 3.25 depicts the *Tagging Quality Evaluation* window after opening the *Existing Text Unit Sample File* `${PROJECT_HOME}/sample5percent.dts` in the first assessment session. In the left panel, the current text unit to be assessed is displayed along with its semantic tag if present. The entire set of XML tags as contained in the derived, collection-specific XML DTD is shown in the right panel. Note that you probably have

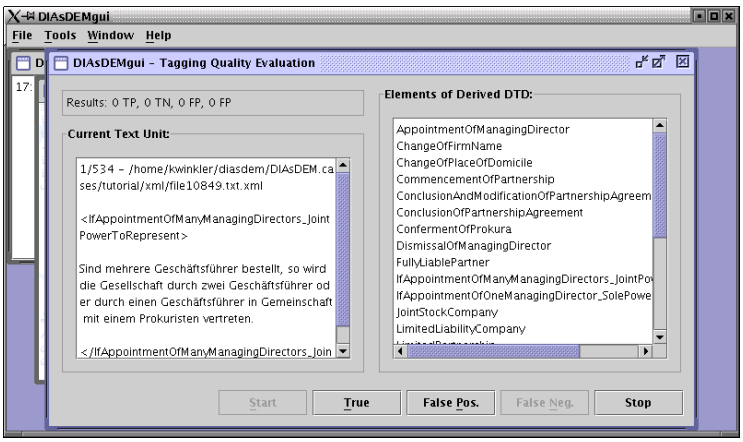

Figure 3.25: *Tagging Quality Evaluation* window of DIAsDEM Workbench 2.0

to evaluate different sentences, since text units are randomly chosen. For obvious reasons, tagged sentences can either be true positives (i.e., having a correct XML tag) or false positives (i.e., having a false XML tag). On the other hand, text units that have not been semantically annotated by DIAsDEM Workbench can either be true negatives (i.e., appropriate XML tags are not contained in DTD) or a false negatives (i.e., appropriate XML tags are actually part of DTD). Please assess 10 text units by clicking the appropriate buttons *True*, *False Pos.* and *False Neg.*, respectively. Thereafter, click on *Stop* and open the file `${PROJECT_HOME}/evaluatedTextUnits.det` which is exemplified by four lines shown below:

```
# TP,TN,FP,FN,Type,TextUnit
1,0,0,0,"TP","/.../file10849.txt.xml <PublicationMediaOfCommercialRegisterEntries>
  Nicht eingetragen: Die Bekanntmachungen der Gesellschaft erfolgen im
  Bundesanzeiger.</PublicationMediaOfCommercialRegisterEntries>"
0,1,0,0,"TN","/.../file10421.txt.xml Die Gesellschafterversammlung vom 17. Oktober
  1997 hat die Erhöhung des Stammkapitals um 50.000 DM auf 100.000 DM und die Änderung
  des Gesellschaftsvertrages in § 3 (Stammkapital, Stammeinlagen) beschlossen."
1,0,0,0,"TP","/.../file10034.txt.xml <NumberOfLimitedPartners>2 Kommanditisten.
  </NumberOfLimitedPartners>"
```

Close the *Tagging Quality Evaluation* window to simulate the end of the current session. Thereafter, the second assessment session can be started by selecting again *Tools* → *Tagging Quality Evaluation*, clicking the *Start* button and choosing the four parameter files one after the other:

Parameter	Value
Existing Text Unit Sample File	`${PROJECT_HOME}/sample5percentB.dts`
New or Existing File of Evaluated Text Units	`${PROJECT_HOME}/evaluatedTextUnits.det`
Text Unit Sample File to be Created for Next Evaluation	`${PROJECT_HOME}/sample5percentC.dts`
Existing Unstructured DTD File	`${PROJECT_HOME}/unstructured.dud`

Figure 3.26 illustrates the *Tagging Quality Evaluation* window at the beginning of the second assessment session. In the left panel, the results of all previous sessions as contained in `${PROJECT_HOME}/evaluatedTextUnits.det` are displayed as well. If you have time, you might assess the remaining 524 text units by clicking the appropriate buttons *True*, *False Pos.* and *False Neg.*, respectively.

The results of each assessment session are appended to `${PROJECT_HOME}/evaluatedTextUnits.det`. After completing the evaluation of tagging quality, this file can be renamed `${PROJECT_HOME}/evaluatedTextUnits.csv` and imported into any spreadsheet application for detailed analysis. If you need the exact number of text units contained in the entire collection for statistical reasons, open *Existing Unstructured DTD File* with any common text editor and search the attribute value `NUMBER_OF_TEXT_UNITS`.

Congratulations, you have now reached the end of this introductory case study! The entire DIAsDEM research group is looking forward to get your feedback concerning the DIAsDEM framework and DIAsDEM Workbench. Bug reports and feature suggestions should be e-mailed to Karsten Winkler (kwinkler@ebusiness.hhl.de) and Henner Graubitz (graubitz@iti.cs.uni-magdeburg.de). Contact Myra Spiliopoulou (myra@iti.cs.uni-magdeburg.de), if your are interested in cooperating with the DIAsDEM group.

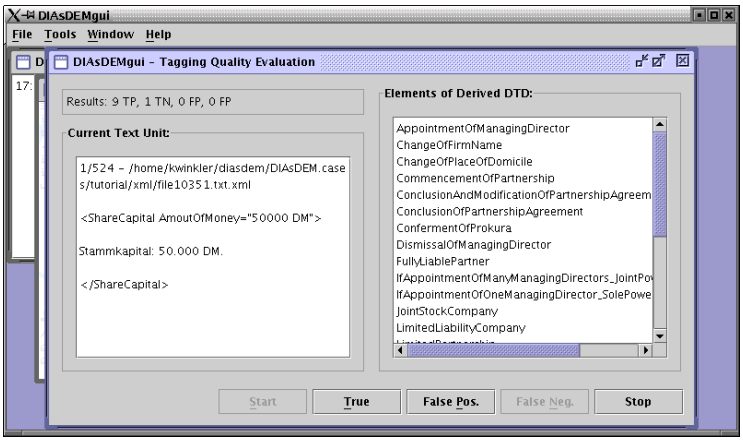

Figure 3.26: *Tagging Quality Evaluation* window of DIAsDEM Workbench 2.0

3.5 Auxiliary Tasks

3.5.1 Create Initial Thesaurus

Before applying the DIAsDEM Workbench to a new collection of text documents, an application-specific thesaurus has to be created. The module described in this section establishes an initial thesaurus based on previously created word frequency statistics. All terms whose collection-specific absolute frequency is greater than or equal to a minimum and less than or equal to a maximum threshold are inserted into a new thesaurus as descriptor terms. However, the resulting initial thesaurus must be manually refined by removing less frequent or semantically unimportant terms. Moreover, the semantics of terms and concepts should be taken into account by defining relations between less important non-descriptors and associated descriptors of importance. Select *Tools* → *Create Initial Thesaurus* and enter the following parameters:

Parameter	Value
Word Statistics File	`${PROJECT_HOME}/lemmaForms.dws`
Initial Thesaurus File	`${PROJECT_HOME}/InitialThesaurus.dth`
Min. Word Frequency	`30`
Max. Word Frequency	`4500`

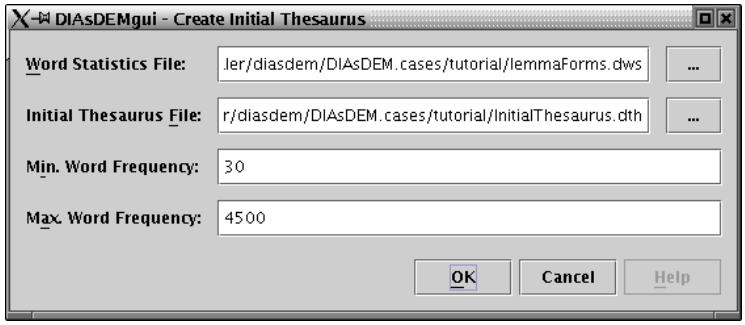

Figure 3.27: *Create Initial Thesaurus* window of DIAsDEM Workbench 2.0

Recall that the input word frequency statistics `${PROJECT_HOME}/lemmaForms.dws` has been created in this case study as described in section 3.3.1. Additionally, note that stop word removal has been skipped in this case study, because of an existing domain-specific thesaurus. However, it is strongly recommended to remove stopwords before creating an initial thesaurus for a new application domain. If semantically less important stopwords are not removed, the resulting thesaurus will contain them as well.

Click on *OK* to start the process of creating an initial thesaurus. Thereafter, inspect and manually refine this thesaurus by selecting *Tools* → *Thesaurus Editor* and opening

the new thesaurus file `${PROJECT_HOME}/InitialThesaurus.dth`. In order to quickly remove less important terms, thesaurus files can also be edited using any common text editor: Simply delete all lines that correspond to terms to be removed.

Create Initial Thesaurus: Summary

Module: *Tools → Create Initial Thesaurus*

Use Case: The user wants to create an application-specific initial thesaurus that is based on collection-specific term frequencies. This thesaurus must subsequently be manually refined using *Tools → Thesaurus Editor*.

Prerequisites: Using *Tools → Create Word Statistics*, a collection-specific term frequency file must have been created for the section `<textUnitsLemmaForms>`.

Result: Descriptors in *Initial Thesaurus File* correspond to terms in *Word Statistics File* whose term frequency is greater than or equal to *Min. Word Frequency* and less than or equal to *Max. Word Frequency*. The remaining terms are not inserted into *Initial Thesaurus File*.

Create Initial Thesaurus: Parameters

Word Statistics File: Valid local file name of existing file; file extension: `.dws`; default value: DIAsDEMgui preference *Default Word Statistics File*

Initial Thesaurus File: Valid local file name of file to be created or replaced by DIAsDEM Workbench; file extension: `.dth`

Min. Word Frequency: Minimum absolute term frequency of descriptors in *Initial Thesaurus File*

Max. Word Frequency: Maximum absolute term frequency of descriptors in *Initial Thesaurus File*

4 Technical Specification

4.1 DIAsDEM Documents

DIAsDEM Workbench processes intermediate XML files that conform to the following XML document type definition `DiasdemDocument.dtd`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- 2002-10-25, kwinkler -->

<!ELEMENT diasdemDocument ( metaData*, text,
  textUnits?, textUnitsTokenized?,
  namedEntities?, textUnitsNamedEntities?,
  textUnitsStopwords?, textUnitsLemmaForms?,
  textUnitsTagged? )>

<!ELEMENT metaData (name, content)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT textUnits (textUnit+)>
<!ELEMENT textUnitsTokenized (textUnitTokenized+)>
<!ELEMENT namedEntities (namedEntity+)>
<!ELEMENT textUnitsNamedEntities (textUnitNamedEntities+)>
<!ELEMENT textUnitsStopwords (textUnitStopwords+)>
<!ELEMENT textUnitsLemmaForms (textUnitLemmaForms+)>
<!ELEMENT textUnitsTagged (textUnitTagged+)>

<!ELEMENT name (#PCDATA)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT textUnit (#PCDATA)>
<!ELEMENT textUnitTokenized (#PCDATA)>
<!ELEMENT namedEntity (#PCDATA)>
<!ELEMENT textUnitNamedEntities (#PCDATA | namedEntityRef)*>
<!ELEMENT textUnitStopwords (#PCDATA | namedEntityRef)*>
<!ELEMENT textUnitLemmaForms (#PCDATA | namedEntityRef)*>
<!ELEMENT textUnitTagged (#PCDATA | namedEntityRef)*>
<!ELEMENT namedEntityRef EMPTY>

<!ATTLIST namedEntity id CDATA #IMPLIED type CDATA #IMPLIED>
<!ATTLIST namedEntityRef id CDATA #IMPLIED>
<!ATTLIST textUnitTagged clusterID CDATA #IMPLIED clusterName CDATA #IMPLIED>
```

4.2 Text Pre-Processing

4.2.1 Module: Create Text Units

Abbreviations File: Valid local file name of existing text file that contains known abbreviations in the following format: Each line of *Abbreviations File* contains exactly one known abbreviation whose capitalization is relevant. The module only matches abbreviations that either occur at the beginning of the text or that are preceded by one of the following characters: blank space, opening parenthesis "(". Comment lines starting with "#" will be ignored and can hence be used to structure the file. Example:

```
Abb.
Abess.
abgeschl.
abgeschloss.
abgest.
Abl.
Ablief.-Gew.
# Note the different capitalization of the following abbreviations:
a.d.
a. d.
a.D.
```

Full Stop Regex File: Valid local file name of existing text file that contains regular expressions in the following format: Each line of *Full Stop Regex File* contains a regular expression matching full stops, exactly one tab stop and thereafter a corresponding replacement string that substitutes each matched full stop with an asterisk. Therefore, the replacement string usually contains references such as `$1` to captured subsequences like `(ges|Ges)` of the corresponding regular expression. Both the regular expression and the replacement string must be Java-compliant constructs as specified in the API documentation of the Java package `java.util.regex`. Comment lines starting with "#" will be ignored and can hence be used to structure the file. Example:

```
# dates
([0-9]{1,2})\.([\ ]*[0-9]{1,2})\.([\ ]*[0-9]{2,4}) $1\*$2\*$3
([0-9]{1,2})\.([\ ]*[0-9]{1,2})\.([\ ]*)   $1\*$2\*$3
# abbreviations that are not preceded by blank space
(str|Str|pl|Pl)\.([\ ]*\d*)  $1\*$2
(ges|Ges)\.([\ ]*mbH) $1\*$2
# complex and context-sensitive abbreviations
(Dipl)\*([\ ]*|[\ ]*\-[\ ]*)([a-zöäüßA-ZÖÄÜ]+)\.   $1\*$2\*$3\*
(Art)\.\ (\d)  $1\*\ $2
# numbers
(\s*[0-9]+)\.(\s*[0-9]+)\.(\s*[0-9]+)\.(\s*[0-9]+) $1\*$2\*$3\*$4
(\s*[0-9]+)\.(\s*[0-9]+)\.(\s*[0-9]+) $1\*$2\*$3
```

```
(\s*[0-9]+)\.(\s*[0-9]+)     $1\*$2
(\s*[0-9]{1,3})\.(\s) $1\*$2
```

Default files of both *Abbreviations File* and *Full Stop Regex File* are provided in the language-specific subdirectories of `${PARAMETER_HOME}/createTextUnits/`.

4.2.2 Module: Tokenize Text Units

Tokenize Regex File: Valid local file name of existing text file that contains regular expressions in the following format: Each line of *Tokenize Regex File* contains a regular expression matching full stops, exactly one tab stop and thereafter a corresponding replacement string that substitutes each matched full stop with an asterisk. Therefore, the replacement string usually contains references such as `$1` to captured subsequences of the corresponding regular expression. Both the regular expression and the replacement string must be Java-compliant constructs as specified in the API documentation of the Java package `java.util.regex`. Comment lines starting with "#" will be ignored and can hence be used to structure the file. Example:

```
# Format: searchRegex<TAB>replaceString
#
(\S)(\.|\!|\?|\(|\)|\{|\}|\[|\]|\-|"|’|‘|’|:|;|,|\+|\/|\\) $1\ $2
(\.|\!|\?|\(|\)|\{|\}|\[|\]|\-|"|’|‘|’|:|;|,|\+|\/|\\)(\S) $1\ $2
```

Normalize Regex File: Valid local file name of existing text file that contains regular expressions in the same format as *Tokenize Regex File* described above. Example:

```
# Format: searchRegex<TAB>replaceString
#
# dates
#
([0-9]{1,2})\.[\ ]*(Januar|Jan[\.]?)[\ ]*([\d]{2,4}) $1.01.$3
([0-9]{1,2})\.[\ ]*(Februar|Feb[r]?[\.]?)[\ ]*([\d]{2,4}) $1.02.$3
([0-9]{1,2})\.[\ ]*(März|M[ä]?r[z]?[\.]?)[\ ]*([\d]{2,4}) $1.03.$3
([0-9]{1,2})\.[\ ]*(April|Apr[\.]?)[\ ]*([\d]{2,4}) $1.04.$3
([0-9]{1,2})\.[\ ]*(Mai)[\ ]*([\d]{2,4})   $1.05.$3
([0-9]{1,2})\.[\ ]*(Juni|Jun[\.]?)[\ ]*([\d]{2,4}) $1.06.$3
([0-9]{1,2})\.[\ ]*(Juli|Jul[\.]?)[\ ]*([\d]{2,4}) $1.07.$3
([0-9]{1,2})\.[\ ]*(August|Aug[\.]?)[\ ]*([\d]{2,4}) $1.08.$3
([0-9]{1,2})\.[\ ]*(September|Sep[t]?[\.]?)[\ ]*([\d]{2,4}) $1.09.$3
([0-9]{1,2})\.[\ ]*(Oktober|Okt[\.]?)[\ ]*([\d]{2,4}) $1.10.$3
([0-9]{1,2})\.[\ ]*(November|Nov[\.]?)[\ ]*([\d]{2,4}) $1.11.$3
([0-9]{1,2})\.[\ ]*(Dezember|Dez[\.]?)[\ ]*([\d]{2,4}) $1.12.$3
([\ ])([0-9]{1})\.[\ ]*([0-9]{1,2})\.[\ ]*([\d]{2,4}) $10$2.$3.$4
([\ ][0-9]{1,2})\.[\ ]*([0-9]{1})\.[\ ]*([\d]{2,4}) $1.0$2.$3
```

Multi Token Words File: Valid local file name of existing text file that contains known multi-token terms in the following format: Each line of *Multi Token Words File* contains exactly one known multi-token word whose capitalization is relevant. Multi-token terms consist of multiple single tokens and blank spaces. Comment lines starting with "#" will be ignored and can hence be used to structure the file. Example:

```
#
# Each line contains exactly on multi-token term.
#
Gesellschaft mit beschränkter Haftung
Offene Handelsgesellschaft
```

The language-specific subdirectories of `${PARAMETER_HOME}/tokenizeTextUnits` contain defaults for *Tokenize Regex File*, *Normalize Regex File* and *Multi Token Words File*.

4.2.3 Module: Replace Named Entities

Forenames File: Valid local file name of existing file that contains a list of forenames in the following format: Each line contains exactly one forename whose capitalization is relevant. Currently, forenames must not be multi-token terms that include blank spaces. However, forenames consisting of multiple tokens separated by either a blank space (e.g., "Hans Joachim") or a hyphen (e.g., "Hans-Joachim") can be identified by user-specific rules in *Composite NE File*. Comment lines starting with "#" will be ignored. Example:

```
# A few frequent forenames ...
Stanka
Cevdet
Wolfgang
Vid
Joaquin
```

Surnames File: Valid local file name of existing file that contains a list of surnames in the following format: Each line contains exactly one surname whose capitalization is relevant. Currently, surnames must not be multi-token terms that include blank spaces. However, surnames consisting of multiple tokens separated by a hyphen (e.g., "Müller-Schmidt") can be identified by user-specific rules in *Composite NE File*. Comment lines starting with "#" will be ignored. Example:

```
# A few frequent surnames ...
Meier
Müller
Schulze
Schmidt
Schmitt
```

Surname Suffixes File: Valid local file name of existing file that contains a list of frequent surname suffixes in the following format: Each line contains exactly one surname suffix whose capitalization is relevant. Only tokens that are preceded by an instance of named entity types "forename" or "middle_initial" are checked against the list of surname suffixes. Comment lines starting with "#" will be ignored. Example:

```
# A few frequent surname suffixes ...
wig
witz
wski
wsky
yer
```

Middle Initials File: Valid local file name of existing file that contains a list of middle initials in the following format: Each line contains exactly one middle initial whose capitalization is relevant. Only tokens that are preceded by an instance of named entity type "forename" are checked against the list of middle initials. Currently, middle initials must not be multi-token terms that include blank spaces. Comment lines starting with "#" will be ignored. Example:

```
# A few middle initials ...
A.
B.
von
De
de
```

Titles File: Valid local file name of existing file that contains a list of academic and professional titles in the following format: Each line contains exactly one title whose capitalization is relevant. Currently, titles must not be multi-token terms that include blank spaces. However, multiple subsequent titles can be merged by user-specific rules in *Composite NE File*. Comment lines starting with "#" will be ignored. Example:

```
# A few titles ...
Prof.
Dr.
Steuerberaterin
Diplom-Bauingenieur
Ing.
```

Places File: Valid local file name of existing file that contains a list of places (i.e., cities) in the following format: Each line contains exactly one place whose capitalization is relevant. Currently, places must not be multi-token terms that include blank spaces. However, multiple subsequent places can be merged by user-specific rules in *Composite NE File*. Comment lines starting with "#" will be ignored. Example:

```
# A few famous places in Germany ...
Berlin
Hamburg
München
Neualbenreuth
Oebles-Schlechtewitz
```

Organizations Start File: Valid local file name of existing file that contains sequences of terms frequently preceding names of organizations as comma-separated values in a format defined by the Java class `diasdem.misc.io.CsvFile`. Except for the first two meta-data lines, each line contains exactly one string attribute value. This attribute value is a sequence of terms that frequently precede organizations in reverse order. Note that tokens must be separated from each others by blank spaces, because named entities are identified in tokenized text units. Comment lines starting with "#" will be ignored. Example:

```
attribute1
String
# Example: Gesellschafterin : <organization>ABC Gmbh</organization>
": Gesellschafterin"
"der"
"die"
"gründenden zu"
":"
```

Organizations End File: Valid local file name of existing file that contains a list of abbreviations for organizations in the following format: Each line contains exactly one abbreviation whose capitalization is relevant. Currently, abbreviations must not be multi-token terms that include blank spaces. However, NEEX aims at identifying complex abbreviations that consist of multiple elementary abbreviations listed in *Organizations End File*. Comment lines starting with "#" will be ignored. Example:

```
# A few common abbreviations of organizations ...
AG
aA
e.G.
```

```
KG.
GmbH
OHG
```

Composite NE File: Valid local file name of existing file that contains rules for instantiating composite named entities as comma-separated values in a format defined by the Java class `diasdem.misc.io.CsvFile`. Except for the first two meta-data lines, each line contains exactly two string attribute values.

The first attribute value is a simple, DIAsDEM-specific regular expression that must be matched by a tokenized text unit in order to instantiate a composite named entity, i.e., "person" or "company". This expression can include case-sensitive tokens (e.g., "mit", "Sitz") and generic placeholders for basic named entities (e.g., "<<organization>>", "<<surname>>") as defined in the Java class `diasdem.neex.NamedEntity`. The second attribute value contains the corresponding named entity constructor that either creates a "person" or a "company". Each constructor references terms and generic placeholders of its corresponding regular expression that are attribute values of the new composite named entity (e.g., "person"). Note that tokens in both expressions must be separated from each others by blank spaces, because named entities are identified in tokenized text units. *Composite NE File* must not contain empty lines. Comment lines starting with "#" will be ignored.

Note: Users only edit or create csv-files containing so-called initial composite named entity rules. Thereafter, the DIAsDEM Workbench module *Tools → Extend Composite NE File* should be employed to extend these initial rules by applying a heuristic permutation algorithm that automatically extends user-supplied initial rules and thus creates the final *Composite NE File*. Extended composite named entity rules cater for the fact that many tokens can be instances of multiple basic named entities. Thus, additional rules are created by extending plain instances of the basic named entities "place", "forename" and "surname" with possible combinations of them. For example, the initial rule "<<organization>> mit Sitz in <<place>>" contains the generic basic named entity placeholder "<<place>>". Therefore, three additional extended rules are generated and added to *Composite NE File* that substitute "<<place>>" with placeholders for the following combinations of named entity types: "<<place forename>>", "<<place surname>>" and "<<place forename surname>>".

The following example depicts a comma-separated values file containing initial composite named entity rules as defined by the user:

```
Attribute0;Attribute1
String;String
# company( Name , Place )
"<<organization>>";"company( 0 , null )"
"<<organization>> mit Sitz in <<place>>";"company( 0 , 4 )"
```

```
"<<organization>> Sitz , <<place>> <<place>> ";"company( 0 , 3 4 )"
"<<organization>> Sitz , <<place>> / <<place>> ";"company( 0 , 3 \"/\" 5 )"
# person( Surname , Forename , Title , MiddleInitial , DoB , MothersName , Place )
"<<forename>> <<surname>>";"person( 1 , 0 , null , null , null , null , null )"
"<<title>> <<surname>> - <<surname>>";"person( 1 \"-\" 3 , null , 0 , null , ... )"
```

Below, an excerpt of *Composite NE File* contains automatically extended composite named entity rules that correspond to the first two initial rules shown above:

```
Attribute0;Attribute1
String;String
"<<organization>>";"company( 0 , null )"
"<<organization>> mit Sitz in <<place>>";"company( 0 , 4 )"
"<<organization>> mit Sitz in <<place forename>>";"company( 0 , 4 )"
"<<organization>> mit Sitz in <<place surname>>";"company( 0 , 4 )"
"<<organization>> mit Sitz in <<place forename surname>>";"company( 0 , 4 )"
```

The DIAsDEM Workbench module *Tools* → *Extend Composite NE File* can be employed to automatically create *Composite NE File* after editing a CSV-file containing domain-specific initial composite named entity rules. Note that the file corresponding to the parameter *Basic NE File* must not be edited. After modifying *Initial Composite NE File*, type in the following parameters to create *Extended Composite NE File* which is commonly referred to as *Composite NE File*:

Parameter	Value
Basic NE File	`${PARAMETER_HOME}/replaceNamedEntities/de/default/BasicNE.csv`
Initial Composite NE File	`${PARAMETER_HOME}/replaceNamedEntities/de/commercialRegister/CommercialRegisterInitialCompositeNE.csv`
Extended Composite NE File	`${PARAMETER_HOME}/replaceNamedEntities/de/commercialRegister/CommercialRegisterExtentedCompositeNE_Modified.csv`

Regex NE File: Valid local file name of existing file that contains regular expressions for the identification of basic named entities as comma-separated values in a format defined by the Java class `diasdem.misc.io.CsvFile`. Except for the first two metadata lines, each line of *Regex NE File* contains two attribute values: The first attribute value is a regular expression matching an instance of the following basic named entities: "number", "date", "time", "amount_of_money", "paragraph", "email" and "url". The second attribute value contains the corresponding named entity type as defined by the Java class `diasdem.neex.NamedEntity`.

Regular expressions must be Java-compliant constructs as specified in the API documentation of the Java package `gnu.regexp`. Comment lines starting with "#" will be ignored and can hence be used to structure the file. Example:

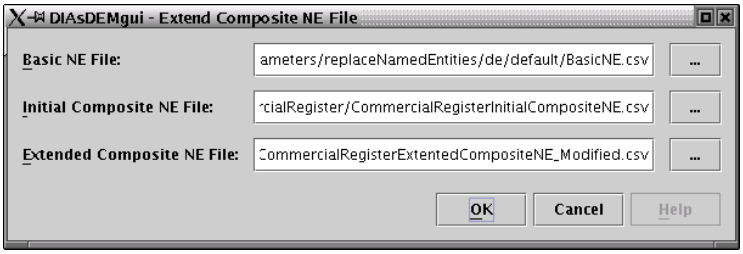

Figure 4.1: *Extend Composite NE File* window of DIAsDEM Workbench 2.0

```
RegularExpression;BasicNamedEntityType
String;String
"\d{1,}[,.\d]{1,}\sDM";amount_of_money
"\d{1,}[,.\d]{1,}\sEUR";amount_of_money
"\d\d.\d\d.\d\d\d\d";date
"\§\s\d+[\s\d\w\.]*BGB";paragraph
"\§\§\s[\d\W]*\s";paragraph
```

The language-specific subdirectories of `${PARAMETER_HOME}/replaceNamedEntities` contain defaults for all parameter files described above.

4.2.4 Module: Remove Stopwords

Stopword File: Valid local file name of existing text file that contains stopwords in the following format: Each line lists exactly one stopword whose capitalization is irrelevant. Stopwords must not be multi-token terms that include blank spaces. Comment lines starting with "#" will be ignored and can hence be used to structure the file. Example:

```
# Each line contains exactly one stopword.
dem
den
denen
denn
der
```

Defaults for *Stopword File* are provided in the language-specific subdirectories of the directory `${PARAMETER_HOME}/removeStopwords`.

4.2.5 Module: Create Lemma Forms

TreeTagger Input File: The name of this temporary file must be set, if *Use TreeTagger to Determine Lemma Form* is enabled. It must be a valid local file name of either a new

or an existing file that will be replaced by the module. This file is created by DIAsDEM Workbench and includes text to be POS-tagged by TreeTagger. Example:

```
<document_/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/xml/file10144.txt.xml>
<textUnitLemmaForms_0>
( Gegenstand : Einzelhandel nebst Montage von Fahrzeugscheiben sowie deren Reparatur ) .
</textUnitLemmaForms_0>
<textUnitLemmaForms_1>
Persönlich haftende Gesellschafterin : <<1004>> .
</textUnitLemmaForms_1>
<textUnitLemmaForms_2>
Kommanditgesellschaft .
</textUnitLemmaForms_2>
<textUnitLemmaForms_3>
Die Gesellschaft hat am <<1003>> begonnen .
</textUnitLemmaForms_3>
<textUnitLemmaForms_4>
1 Kommanditist .
</textUnitLemmaForms_4>
</document_/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/xml/file10144.txt.xml> ...
```

TreeTagger Output File: The name of this temporary file must be set, if *Use TreeTagger to Determine Lemma Form* is enabled. It must be a valid local file name of either a new or an existing file that will be extended by the module. This file is created by TreeTagger and includes the results of POS-tagging for subsequent parsing the DIAsDEM Workbench. Example:

```
<document_/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/xml/file10144.txt.xml>
<textUnitLemmaForms_0>
(       $(      (
Gegenstand      NN      Gegenstand
:       $.      :
Einzelhandel    NN      Einzelhandel
nebst   APPR    nebst
Montage NN      Montag|Montage
von     APPR    von
Fahrzeugscheiben        NN      <unknown>
sowie   KON     sowie
deren   PRELAT d
Reparatur       NN      Reparatur
)       $(      )
.       $.      .
</textUnitLemmaForms_0>
<textUnitLemmaForms_1> ...
```

Known Lemma Forms: The name of this parameter file must be set, if *Look Up Lemma Form in List* is enabled. It must be a valid local file name of an existing file that contains terms along with their lemma forms in the following format: The first line contains the attribute names separated by a single tab stop. Thereafter, each line lists one term, exactly one tab stop and thereafter the corresponding grammatical root form. Terms and lemma forms must not be multi-token terms that include blank spaces. However, blank spaces in multi-token terms can be replaced with underscores (e.g., "for_example"). Note that capitalization of terms is irrelevant, but the capitalization of lemma forms is retained when replacing the corresponding terms. Comment lines starting with "#" will be ignored and can hence be used to structure the file. Example:

```
TERM   LemmaTerm
# Each line contains a term, a tab stop and the lemma form.
jetzt  jetzt
Investitionsgütern    Investitionsgut
dem    d
mobilen mobil
Gesellschaft_mit_beschränkter_Haftung Gesellschaft_mit_beschränkter_Haftung_unknown
Anlegen anlegen
Rödelheim     Rödelheim_NE
```

Unknown Lemma Forms: The name of this temporary file must be set, if *Look Up Lemma Form in List* is enabled. It must be a valid local file name of either a new or an existing file that will be extended by the module. This file is created or extended by DIAsDEM Workbench and includes terms occuring in the collection that are not listed in the file of *Known Lemma Forms* as well as the context of their occurrence (i.e., the sentence) separated by exactly one tab stop. This file could be used to update the file *Known Lemma Forms* with new terms. Example:

```
.Kaufm. Dipl. - .Kaufm. <<1099>> , <<1064>> , ist zum Geschäftsführern bestellt .
.Wirt. <<1031>> , Speditionskaufmann , <<1005>> ; <<1032>> , Dipl. <<1008>> ...
Art.   Der Gesellschaftsvertrag ist am <<1001>> abgeschlossen und am <<1002>> ...
Dip.   Dip. - Ing .
lit.   1. Der An - und Verkauf von Immobilien sowie die Beteiligung an ...
```

4.3 Iterative Clustering

4.3.1 Module: Create Text Unit Vectors

Vector File Format: DIAsDEM Workbench supports the export of three file formats: comma separated values (CSV-files), fixed width values (TXT-files) and ARFF-files in the Weka-specific format described in [WF99]. See below an example of a text unit vector file in comma separated values format:

```
DocumentType,Document,TextUnit,D1_Aktie,D2_Gesellschafter,D3_Inhaber,...,D74_Anspruch
"null","/file10144.txt.xml",1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.42154,0,...
"null","/file10144.txt.xml",2,0,3.97572,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...
"null","/file10144.txt.xml",3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...
"null","/file10144.txt.xml",4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.32319,0,0,0,0,0,0,0,...
```

In the current version of DIAsDEM Workbench, "DocumentType" is always "null" due to legacy reasons. The attribute "Document" contains file names of intermediate XML files relative to the collection directory that corresponds to the case-specific *Collection File*. Currently, all collection files must reside in the same directory. Values of the attribute "TextUnit" uniquely identify text units within a given document by their sequence number. Note that "Document" and "TextUnit" constitute a composite primary key for text units in the scope of the corresponding *Collection File*. The following meta-data file summarize information about all attributes of the CSV-file above:

```
DocumentType
Document
TextUnit
D1_Aktie = Aktie; Term Frequency = 38; Term Weight = 5.64144
D2_Gesellschafter = Gesellschafter; Term Frequency = 201; Term Weight = 3.97572
...
D74_Anspruch = Anspruch; Term Frequency = 9; Term Weight = 7.08180
```

Below is an example of a text unit vector file in fixed width values format:

```
DocumentType␣␣␣␣␣␣␣␣Document␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣TextUnit␣␣D1_Aktie␣␣␣␣␣␣␣␣␣␣␣␣D2_Gese...
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/file10144.txt.xml␣␣␣␣␣␣␣␣1␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣...
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/file10144.txt.xml␣␣␣␣␣␣␣␣2␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣3.97572...
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/file10144.txt.xml␣␣␣␣␣␣␣␣3␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣...
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/file10144.txt.xml␣␣␣␣␣␣␣␣4␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣...
```

As indicated above, simple blank spaces separate attribute values from each others. The following meta-data file corresponds to the TXT-file above and additionally contains information about the width of each attribute. Note that file names of intermediate XML files cannot exceed 25 characters. Currently, the width of attributes cannot be changed by the user.

```
1-20   DocumentType
21-45  Document
46-55  TextUnit
56-75  D1_Aktie = Aktie; Term Frequency = 38; Term Weight = 5.64144
76-95  D2_Gesellschafter = Gesellschafter; Term Frequency = 201; Term Weight = 3.9...
...
1516-1535     D74_Anspruch = Anspruch; Term Frequency = 9; Term Weight = 7.08180
```

See below an example of a Weka-specific text unit vector file in ARFF-format:

```
@relation 'DIAsDEM'
@attribute DocumentType string
@attribute Document string
@attribute TextUnit string
@attribute D1_Aktie real
@attribute D2_Gesellschafter real
...
@attribute D74_Anspruch real
@data
null,/file10144.txt.xml,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.42154,...
null,/file10144.txt.xml,2,0,3.97572,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...
...
```

The following meta-data file corresponds to the ARFF-file above:

```
DocumentType
Document
TextUnit
D1_Aktie = Aktie; Term Frequency = 38; Term Weight = 5.64144
D2_Gesellschafter = Gesellschafter; Term Frequency = 201; Term Weight = 3.97572
...
D74_Anspruch = Anspruch; Term Frequency = 9; Term Weight = 7.08180
```

Thesaurus File: The existing DIAsDEM-specific thesaurus file must be identified by a valid local file name. Except for comment lines starting with "#", each line corresponds to exactly one thesaurus entry that can either be a descriptor (i.e., preferred term) or a non-descriptor (i.e., non-preferred term). Non-descriptor terms must always point to an associated descriptor in the same thesaurus file that should be used for indexing and term frequency counting instead. Note that DIAsDEM-specific thesauri must only include grammatical root forms of terms (i.e., their so-called lemma forms) as determined by the module *File → Create Lemma Forms*. Thesauri can be created by *Tools → Create Initial Thesaurus* and modified by *Tools → Thesaurus Editor*. Example:

```
# Terms of Thesaurus /.../thesauri/de/commercialRegister/CommercialRegisterThesaurus.dth
19535 "<<company>>" 1 "TY=D" "HL=-" "SY=-" "BT=-" "NT=-" "UD=-" "SN=Case1" ...
19461 "Ablehnung" 1 "TY=D" "HL=-" "SY=-" "BT=-" "NT=-" "UD=-" "SN=Case2"
10628 "abschließen" 1830 "TY=D" "HL=-" "SY=-" "BT=-" "NT=-" "UD=-" "SN=Case1 Case2" ...
12299 "Änderung" 24 "TY=D" "HL=-" "SY=-" "BT=-" "NT=-" "UD=-" "SN=Case1 Case2" ...
12087 "Zweck" 16 "TY=N" "HL=-" "SY=-" "BT=-" "NT=-" "UD=Tätigkeit" "SN=-"
10859 "Zwecke_|" 10 "TY=N" "HL=-" "SY=-" "BT=-" "NT=-" "UD=Tätigkeit" "SN=-"
10797 "ändern" 396 "TY=N" "HL=-" "SY=-" "BT=-" "NT=-" "UD=Änderung" "SN=-"
```

Thesaurus terms can either be lemma forms of words (e.g., “Ablehnung” and “ändern”) or named entity type placeholders such as “<<company>>” and “<<person>>”. Consider for example the thesaurus entry “Ablehnung”: “19461” is a unique term identifier within the thesaurus, the term type field “TY=D” denotes that “Ablehnung” is a descriptor term and the scope notes field “SN=Case2” can be used to filter valid descriptors in different case studies and clustering iterations, respectively. The use descriptor field (“UD=-”) remains empty for descriptor terms for obvious reasons. Furthermore, consider the thesaurus entry “ändern” which is a non-descriptor (“TY=N”). The descriptor term “Änderung” should be used instead of “ändern”, because of the use descriptor field “UD=Tätigkeit”. Note that hierarchy level (“HL=-”), synonyms (“SY=-”), broader term (“BT=-”) and narrower term (“NT=-”) are neither used for descriptor nor non-descriptors in the current version of DIAsDEM Workbench.

4.3.2 Module: Cluster Text Unit Vectors

Text Unit Vectors File: Valid local file name containing text unit vectors to be clustered in ARFF format as specified above in section 4.3.1. Note that the internal Weka-based clustering algorithms cannot process other file formats.

Clustering Results File: Valid local file name of a file to be created or replaced by DIAsDEM Workbench that contains the mappings of text units onto clusters in CSV format as specified below in section 4.3.3. Note that all internal Weka-based clustering algorithms cannot output other file formats.

Text Unit Clusterer File: Valid local file name of a file to be created or replaced by DIAsDEM Workbenchthat contains a serialized instance of the Java class corresponding to *Clustering Algorithm*. *Text Unit Clusterer File* is an output parameter in clustering mode, but an input parameter in application mode. Note, there must be a correspondance between the specified *Clustering Algorithm* in clustering and application mode. In the latter phase, an instance of the respective text unit clusterer is created as follows:

```
ObjectInputStream modelInputStream = new ObjectInputStream( new FileInputStream(
  Parameter.getClusterModelFileName() ) );
switch ( Parameter.getClusteringAlgorithm() ) {
  case ClusterTextUnitVectorsParameter.WEKA_SIMPLE_KMEANS: {
    clusterer = (SimpleKMeans)modelInputStream.readObject();
    break;
  }
  case ClusterTextUnitVectorsParameter.WEKA_COBWEB: {
    clusterer = (Cobweb)modelInputStream.readObject();
    break;
  }
  case ClusterTextUnitVectorsParameter.WEKA_EM: {
    clusterer = (EM)modelInputStream.readObject();
    break;
  }
```

```
}
modelInputStream.close();
```

4.3.3 Module: Monitor Cluster Quality

Result File Format: *Cluster Result File* maps text unit vectors onto their respective clusters which are identified by integers. Currently, each text unit vector can only be assigned to exactly one cluster. DIAsDEM Workbench can import result files in the following two formats: comma separated values (CSV-files) and fixed width values (TXT-files). In both cases, *Cluster Result File* must contain exactly three attributes for each text unit vector. The file name within the current collection (i.e., *Collection File*) is the first attribute. It is followed by the text unit identifier as the second and the cluster ID associated with the respective text unit vector as the third attribute. The first two attributes (i.e., file name and text unit identifier) correspond to the attributes "Document" and "TextUnit" of text unit vector files as described in section 4.3.1. Valid cluster IDs are integers being greater than zero. Text units vectors in *Cluster Result File* should be ordered as in the corresponding *Text Unit Vectors File*.

Note, DIAsDEM Workbench can only process files that completely conform to the syntax exemplified by the following two file excerpts. Hence, clustering algorithms must either be configured to create appropriate result files or intermediate output files must be post-processed by for example Perl scripts. See below an example of a cluster result file in comma separated values format:

```
/file10144.txt.xml,1,2
/file10144.txt.xml,2,35
/file10144.txt.xml,3,68
/file10144.txt.xml,4,77
/file10144.txt.xml,5,71
/file11136.txt.xml,1,2
```

Below is an example of a text unit vector file in fixed width values format:

```
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/file10144.txt.xml␣␣␣␣␣␣␣1␣␣␣␣␣␣␣␣␣2
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/file10144.txt.xml␣␣␣␣␣␣␣2␣␣␣␣␣␣␣␣␣35
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/file10144.txt.xml␣␣␣␣␣␣␣3␣␣␣␣␣␣␣␣␣68
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/file10144.txt.xml␣␣␣␣␣␣␣4␣␣␣␣␣␣␣␣␣77
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/file10144.txt.xml␣␣␣␣␣␣␣5␣␣␣␣␣␣␣␣␣71
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/file11136.txt.xml␣␣␣␣␣␣␣1␣␣␣␣␣␣␣␣␣2
```

As indicated above, only blank spaces are allowed to separate attribute values from each others. The following meta-data file corresponds to the TXT-file above and contains

information about the width of each attribute. Note that file names of intermediate XML files cannot exceed 25 characters. Currently, the width of attributes cannot be changed by the user. In the current version of DIAsDEM Workbench, "DocumentType" has always the "null" value due to legacy reasons. In contrast to fixed width files, CSV-files must not contain the attribute "DocumentType".

```
1-20   DocumentType
21-45  Document
46-55  TextUnit
56-58  ClusterID
```

Cluster Result File: Valid local file name of a file to be created or replaced by DIAsDEM Workbench that conforms to *Result File Format*.

Thesaurus File: Valid local file name of existing DIAsDEM-specific thesaurus file as described in section 4.3.1.

4.3.4 Module: Tag Text Units

Result File Format: One of two result file formats (i.e., comma separated values and fixed width values) supported by DIAsDEM Workbench that are described in section 4.3.1.

Cluster Result File: Valid local file name of a file to be created or replaced by DIAsDEM Workbench that conforms to *Result File Format*.

Cluster Label File: Valid local file name of existing file created by DIAsDEM Workbench in *File* → *Monitor Cluster Quality* and possibly modified by *Tools* → *Cluster Label Editor*.

4.4 XML Tagging of Texts

4.4.1 Module: Tag Documents

Unstructured DTD File: Valid local file name of file to be created or replaced by DIAsDEM Workbench that contains meta-data about the preliminary XML document type definition, its XML tags and their attribues in a DIAsDEM-specific format. The following *Unstructured DTD File* has been created in this case study:

```
#This is an automatically created file: Please do not edit this file manually!
#Thu Mar 20 21:02:51 CET 2003
NUMBER_OF_UNTAGGED_TEXT_UNITS=777
ELEMENTS_FILE_NAME=/home/.../DIAsDEM.cases/tutorial/unstructured.dud.elements
MIN_ATTRIBUTE_REL_SUPPORT=0.1
ROOT_ELEMENT=CommercialRegisterEntry
NUMBER_OF_TEXT_UNITS=10702
```

```
XML_DTD_FILE_NAME=/home/.../DIAsDEM.cases/tutorial/unstructured.dud.xml
NUMBER_OF_TAGGED_TEXT_UNITS=9925
NUMBER_OF_DOCUMENTS=1146
RANDOM_SAMPLE_SIZE=0.05
ATTRIBUTES_FILE_NAME=/home/.../DIAsDEM.cases/tutorial/unstructured.dud.attributes
```

As shown below, the file `/home/.../DIAsDEM.cases/tutorial/unstructured.dud.elements` contains meta-data about DTD elements (i.e., XML tags):

```
# UnstructuredDtdElement,AbsoluteSupport,RelativeSupport
"AppointmentOfManagingDirector" 885 0.7722513089005235
"ChangeOfFirmName" 60 0.05235602094240838
"ChangeOfPlaceOfDomicile" 35 0.030541012216404886
...
"SolePowerToRepresent_PowerToContractWithOneself" 640 0.5584642233856894
```

The file `/home/.../DIAsDEM.cases/tutorial/unstructured.dud.attributes` contains meta-data about attributes of DTD elements (i.e., XML tags) as exemplified below:

```
# UnstructuredDtdElement,UnstructuredDtdAttribute,MostProbableType,
# AbsoluteSupport,RelativeSupport
"AppointmentOfManagingDirector" "Date" "Date" 255 0.288135593220339
"AppointmentOfManagingDirector" "Person" "Person" 755 0.8531073446327684
"ChangeOfFirmName" "Company" "Company" 60 1.0
...
"ShareCapital" "AmoutOfMoney" "AmoutOfMoney" 893 0.9944320712694877
```

Random Sample File: Valid local file name of file to be created or replaced by DIAsDEM Workbench which contains a random sample from all text units (i.e., both tagged and untagged ones) in a DIAsDEM-specific format. Along with *Unstructured DTD File*, this file is input to the module *Tools* → *Tagging Quality Evaluation*. For example, see below three lines of *Random Sample File* as created in this case study. Note that the second line corresponds to an untagged sentence, whereas the remaining ones contain semantically annotated sentences.

```
/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/xml/file10849.txt.xml
  <PublicationMediaOfCommercialRegisterEntries>Nicht eingetragen: Die
  Bekanntmachungen der Gesellschaft erfolgen im Bundesanzeiger.
  </PublicationMediaOfCommercialRegisterEntries>
/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/xml/file10421.txt.xml Die
  Gesellschafterversammlung vom 17. Oktober 1997 hat die Erhöhung des
  Stammkapitals um 50.000 DM auf 100.000 DM und die Änderung des
  Gesellschaftsvertrages in § 3 (Stammkapital, Stammeinlagen) beschlossen.
/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/xml/file10034.txt.xml
  <NumberOfLimitedPartners>2 Kommanditisten.</NumberOfLimitedPartners>
```

List of Abbreviations

DFG	Deutsche Forschungsgemeinschaft (German Research Society)
DIAsDEM	Datenintegration von Altlastdaten und semistrukturierten Dokumenten mit Mining Verfahren (German project acronym that stands for "integration of legacy data and semi-structured documents with data mining techniques")
DTD	Document Type Definition
FN	false negative
FP	false positive
geb.	geboren (born on a date)
HHL	Handelshochschule Leipzig (Leipzig Graduate School of Management)
ID	identifier
KDD	Knowledge Discovery in Databases
KDT	Knowledge Discovery in Textual Databases
NE	named entity
NEEX	Named Entity Extractor (module of DIAsDEM Workbench)
POS	part-of-speech
Regex	regular expression
TFxIDF	term frequency multiplied by inverse document frequency
TN	true negative
TP	true positive
Weka	Waikato Environment for Knowledge Analysis
XML	Extensible Markup Language

List of Relevant German Vocabulary

The following list contains German nouns and verbs that might be useful to understand the meaning of Commercial Register entries in this case study. This list is based on a translation of the German Commercial Code by Peltzer, Doyle and Voight which includes a concise introduction to the German Commercial Code as well [PDV00, pp. 1–32].

Aktiengesellschaft (AG) German joint stock corporation

Aktionär (Aktionäre) shareholder of German joint stock corporation (AG)

Amtsgericht District Court in Germany; a local Commercial Register is usually maintained by the respective District Court

Änderung change or modification of sth. (e.g., modification of partnership agreement)

Anspruch legal claim against sb.

Bauvorhaben building project; here: purpose of certain companies

Beginn here: commencement of operations

beginnen (beginnt) here: to commence with operations

Bekanntmachung (Bekanntmachungen) information that has to be officially published by companies according to the German Commercial Code

Bundesanzeiger official German newspaper that weekly publishes Commercial Register entries and corporate news

bestellen (bestellt) here: to appoint sb. to a position of responsibility (e.g., to appoint sb. as managing director of a German limited liability company)

eingetragen here: (e.g., legal facts) to be registered with the Commercial Register

Einzelvertretungsbefugnis sole power to legally represent a company (in contrast to joint power to represent a company)

erfolgen (erfolgt) here: to publish information according to the German Commercial Code

Erhöhung increase in sth. (e.g., increase in share capital)

erteilen (erteilt) here: to confer (e.g., Prokura or power to represent a company)

Firma here: 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

Geschäftsführer, Geschäftsführerin managing director of German limited liability company (GmbH)

Gesellschafter, Gesellschafterin partner in German commercial partnership (e.g., OHG and KG) or in German limited liability company (GmbH)

Gesellschaft here: (commercial) partnership and company, respectively

Gesellschaft mit beschränkter Haftung (GmbH) German limited liability company

Gesellschafterversammlung meeting of (commercial) partners and share holders, respectively

Gesellschaftsvertrag commercial partnership agreement

Handel mit Waren trading of goods

Kommanditist (Kommanditisten) fully liable partner in German limited partnership (KG)

Kommanditgesellschaft (KG) German limited partnership

Offene Handelsgesellschaft (OHG) German commercial partnership

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

Stammkapital share capital of German limited liability company (GmbH)

Tätigkeit here: purpose of company

vertreten (vertritt) here: to legally represent a company

Vorstand managing board of German joint stock company (AG)

Zweigniederlassung branch office of a company

Bibliography

[GSW01] Henner Graubitz, Myra Spiliopoulou, and Karsten Winkler. The DIAsDEM framework for converting domain-specific texts into XML documents with data mining techniques. In *Proceedings of the First IEEE International Conference on Data Mining*, pages 171–178, San Jose, CA, USA, November/December 2001.

[GWS01] Henner Graubitz, Karsten Winkler, and Myra Spiliopoulou. Semantic tagging of domain-specific text documents with DIAsDEM. In *Proceeding of the 1st International Workshop on Databases, Documents, and Information Fusion (DBFusion 2001)*, pages 61–72, Magdeburg, Germany, May 2001.

[ISO86] ISO. Documentation: Guidelines for the establishment and development of monolingual thesauri. Technical Report ISO 2788-1986 (E), International Organisation for Standardization, 1986.

[PDV00] 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, 4th revised edition, 2000.

[Sch94] 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. TreeTagger is available at http://www.ims.uni-stuttgart.de/∼schmid, accessed 2003-04-01.

[Sul01] Dan Sullivan. *Document Warehousing and Text Mining.* John Wiley & Sons, New York, Chichester, Weinheim, 2001.

[SW02] Myra Spiliopoulou and Karsten Winkler. Text Mining auf Handelsregistereinträgen: Der SAS Enterprise Miner im Einsatz. In Klaus D. Wilde, Hajo Hippner, and Melanie Merzenich, editors, *Data Mining: Mehr Gewinn aus Ihren Kundendaten*, pages 117–124. Verlagsgruppe Handelsblatt, Düsseldorf, 2002.

[WF99] Ian H. Witten and Eibe Frank. *Data Mining: Practical Mechine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, San

Francisco, 1999. Weka is available at http://www.cs.waikato.ac.nz/∼ml/weka, accessed 2003-04-01.

[WS01a] Karsten Winkler and Myra Spiliopoulou. Extraction of semantic XML DTDs from texts using data mining techniques. In *Proceedings of the K-CAP 2001 Workshop on Knowledge Markup and Semantic Annotation*, pages 59–68, Victoria, BC, Canada, October 2001.

[WS01b] Karsten Winkler and Myra Spiliopoulou. Integrating data and probabilistically structured text documents. In *Proceedings des 5. Workshops "Föderierte Datenbanken" und GI Arbeitstreffen "Konzepte des Data Warehousing" (FDBS 2001)*, pages 16–29, Berlin, Germany, October 2001.

[WS01c] Karsten Winkler and Myra Spiliopoulou. Semi-automated XML tagging of public text archives: A case study. In *Proceedings of EuroWeb 2001 "The Web in Public Administration"*, pages 271–285, Pisa, Italy, December 2001.

[WS02a] Karsten Winkler and Myra Spiliopoulou. Employing text mining for semantic tagging in DIAsDEM. *KI – Künstliche Intelligenz*, 16(2):27–29, 2002.

[WS02b] Karsten Winkler and Myra Spiliopoulou. Structuring domain-specific text archives by deriving a probabilistic XML DTD. In *Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02)*, pages 461–474, Helsinki, Finland, August 2002.

[WS02c] Karsten Winkler and Myra Spiliopoulou. Text Mining in der Wettbewerberanalyse: Konvertierung von Textarchiven in XML-Dokumente. In *Data Mining und Statistik in Hochschule und Wirtschaft: Proceedings der 6. Konferenz der SAS-Anwender in Forschung und Entwicklung (KSFE)*, pages 347–363, Dortmund, Germany, February/March 2002.