Technical Report

Getting Started with DIAsDEM Workbench 2.2: A Case-Based Tutorial

Karsten Winkler

Otto von Guericke University of Magdeburg
Faculty of Computer Science
Research Group ITI/KMD
E-Mail: k@rsten-winkler.de

Contents

1 Introduction

Most organizations are not only 'drowning' in data, they are also 'struggling' to cope with huge amounts of text documents. It is estimated that 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 information retrieval or to enable information integration with related data sources. Unfortunately, most users are not willing to manually create high-quality and consistent semantic metadata due to the efforts and costs involved. Thus, text mining techniques are required that (semi-) automatically create semantic markup.

Using the Extensible Markup Language XML, semantic annotation of text archives results in domain-specific, semantic metadata in the form of XML tags that adhere to a domain-specific XML document type definition (DTD). Semantic metadata 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 semantically tagged XML archives. However, two main problems must be solved to semi-automatically create text annotations: Firstly, a concept-based XML document type definition should be derived for each textual archive. Secondly, all text documents contained in an archive should be semantically tagged according to the previously established 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. Figure 1.1 illustrates the user interface of DIAsDEM Workbench 2.2. The Java-based research prototype supports the entire framework including automated batch processing.

1.1 DIAsDEM Framework

In the research project DIAsDEM, the notion of semantic tagging refers to annotating texts with domain-specific, content-descriptive XML tags that optionally comprise attributes describing extracted named entities (e.g., names of persons). Rather than classifying entire documents or tagging single terms, we aim at semantically annotating

[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.

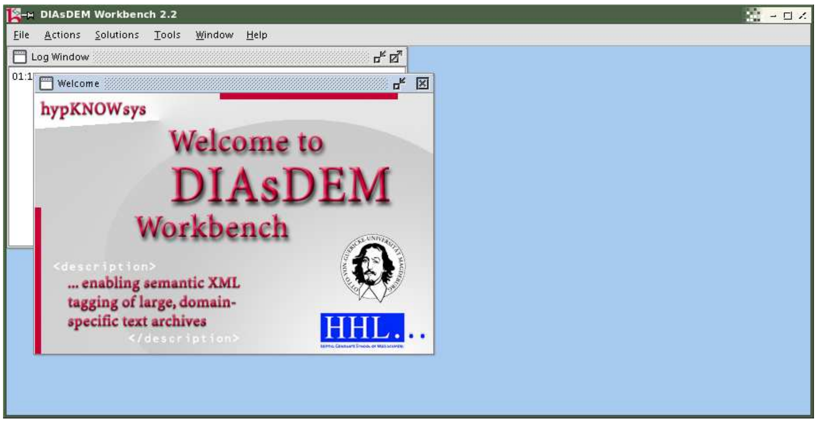


Figure 1.1: Java-based GUI of DIAsDEM Workbench 2.2

structural text units, such as sentences or paragraphs, to make their semantics explicit. The following example illustrates two tagged sentences contained in a German Commercial Register entry such that 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 to discover clusters of semantically similar text units, to derive a concept-based XML DTD describing the archive, and to semantically mark up documents in XML accordingly. The KDT process, which is depicted in Figure 1.2, results in a final set of clusters. Their semantic labels serve as DTD elements and XML tags, respectively. Huge amounts of new documents from the same domain are automatically 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, as well as descriptions of relevant named entities (e.g., names of persons and companies).

Similarly to a conventional KDT process, the process starts with a pre-processing phase that includes basic NLP pre-processing tasks, such as tokenization, normalization, lemmatization, and named entity extraction. Instead of removing stop words, we establish a drastically reduced feature space by selecting a limited set of terms and concepts (i.e., so-called text unit descriptors) from the domain-specific thesaurus. Text unit

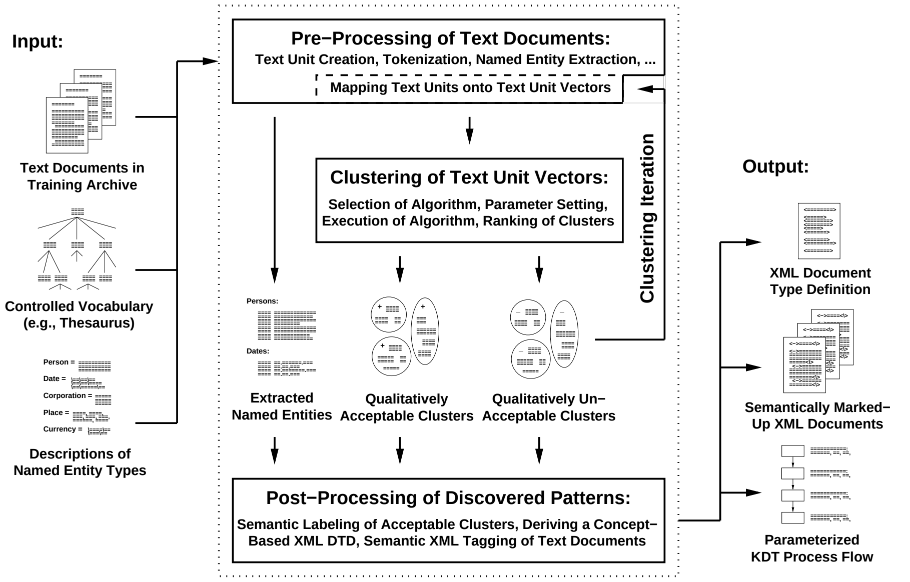

Input:
Text Documents in Training Archive
Controlled Vocabulary (e.g., Thesaurus)
Person =
Date =
Corporation =
Place =
Currency =
Descriptions of Named Entity Types
Pre−Processing of Text Documents:
Text Unit Creation, Tokenization, Named Entity Extraction, ...
Mapping Text Units onto Text Unit Vectors
Clustering of Text Unit Vectors:
Selection of Algorithm, Parameter Setting, Execution of Algorithm, Ranking of Clusters
Clustering Iteration
Persons:
Dates:
Extracted Named Entities
Qualitatively Acceptable Clusters
Qualitatively Un− Acceptable Clusters
Post−Processing of Discovered Patterns:
Semantic Labeling of Acceptable Clusters, Deriving a Concept−Based XML DTD, Semantic XML Tagging of Text Documents
Output:
XML Document Type Definition
Semantically Marked−Up XML Documents
Parameterized KDT Process Flow

Figure 1.2: Iterative and Interactive KDT Process of the DIAsDEM Framework

descriptors are chosen by the knowledge engineer because they must reflect important concepts of the application domain. Text units are initially mapped onto Boolean vectors of this feature space. Thereafter, Boolean text unit vectors are further processed by applying the information retrieval TFxIDF weighting schema.

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 subsequently partitioned into qualitatively acceptable and unacceptable ones according to framework-specific quality criteria. A cluster of text unit vectors is qualitatively acceptable if (i) its cardinality is sufficiently large, (ii) the corresponding text units are homogeneous, and (iii) the text units can be content-descriptively characterized by a small number of text unit descriptors. Members of acceptable clusters are subsequently removed from the data set for later labeling whereas the remaining text unit vectors are input to the clustering algorithm in the next iteration. In each iteration, the cluster similarity threshold value is gradually 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-processing 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, a concept-based XML DTD is derived that coarsely describes the semantic structure of the XML collection by enumerating discovered XML tags. Finally, original documents are annotated by valid XML tags with respect to the derived XML DTD. The archive-specific XML tags optionally include attributes reflecting previously extracted named entities and their values. The following DTD excerpt was created in a case study [WS01c]:

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

1.2 Code Credits, Third-Party Licenses, and Trademarks

Markus Banach, Martin Christian, Henner Graubitz, Ingo Kampe, Heiko Scharff, and Karsten Winkler are code contributors. 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.

DIAsDEM Workbench 2.2 utilizes WEKA 3.4.3, 29 September 2004, Java Programs for Machine Learning. Copyright (C) 1998-2004 University of Waikato. WEKA is distributed under the GNU public license available at http://www.opensource.org/licenses/gpl-license.php.

DIAsDEM Workbench 2.2 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.

DIAsDEM Workbench 2.2 utilizes JGoodies Looks 1.2.2 (http://www.jgoodies.com/freeware/looks/). The BSD License for the JGoodies Looks: Copyright (c) 2001-2004 JGoodies Karsten Lentzsch. 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.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of JGoodies Karsten Lentzsch nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

DIAsDEM Workbench 2.2 utilizes JGoodies Forms 1.0.2 (http://www.jgoodies.com/freeware/forms/). The BSD License for the JGoodies Forms: Copyright (c) 2003 JGoodies Karsten Lentzsch. 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.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of JGoodies Karsten Lentzsch nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission

DIAsDEM Workbench 2.2 utilizes hypKNOWsys Algorithms 0.1, Java Package for Knowledge Discovery and Knowledge Management. hypKNOWsys Algorithms is based on and extends 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. hypKNOWsys Algorithms is distributed under the GNU public license available at http://www.opensource.org/licenses/gpl-license.php.

In the task ‘Search the Web for HTML Files’, DIAsDEM Workbench 2.2 utilizes the run-time library (in the file $DIAsDEM_HOME/lib/googleapi.jar) of the Google Web APIs (TM) exactly as supplied by Google. In order to use the Google Web APIs, users first must register with Google to receive a personal authentication key and declare to abide by Google's terms of use. As of May 2004, this can be done online at http://www.google.com/apis/. Please see the Google's terms of use, a copy of which is available in the file DIAsDEM_HOME/lib/googleapi.license.txt, for more information. Excerpt of Google's terms of use: "INTELLECTUAL PROPERTY You agree not to remove, obscure, or alter Google's copyright notice, trademarks, or other proprietary rights notices affixed to or contained within Google Web APIs. You also acknowledge that Google owns all right, title and interest in and to Google Web APIs, including without limitation all intellectual property rights (the "Google Rights"). The Google Rights include rights to the following: (1) the APIs developed and provided by Google, (2) all software associated with the Google Web APIs server, and (3) the search results and spell checking you obtain when you use Google Web APIs. The Google Rights do not include the following: (1) third party components used as part of Google Web APIs; or (2) software developed by you in conjunction with using Google Web APIs." The complete terms and conditions can be downloaded from the Google Web site (http://www.google.com/apis/api_terms.html).

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 2.2

DIAsDEM Workbench 2.2, Copyright (C) 2000-2006, Henner Graubitz, Myra Spiliopoulou, Karsten Winkler. All rights reserved. DIAsDEM Workbench 2.2 is distributed under the GNU general public license available at http://www.opensource.org/licenses/gpl-license.php.

1.4 Typographical Conventions

Italic is used for emphasis within text and to indicate selectable items in DIAsDEM Workbench dialogs and menus. *Italic* is also used to represent field names (i.e., parameters) in DIAsDEM Workbench dialogs. `Courier` is used to refer to directories, file names, and file extentions. Additionally, `Courier` is used for computer output, XML tags, and the content 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.workbench22`.

- `${PARAMETER_HOME}` denotes the local subdirectory of `${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`.

1.5 Semantic XML Tagging of English Texts

The case study described in this technical report is based on processing German texts. Nevertheless, our workbench is capable of processing English texts. For example, DIAsDEM Workbench includes tasks designed to import texts from two English Reuters corpora; namely, the Reuters-21578 text categorization test collection compiled by David D. Lewis and the new Reuters Corpus, Vol. 1, English Language, 1996-08-20 to 1997-08-19 [RSW02, LYRL04]. Contact the author to obtain English parameter files, which are not yet packaged into DIAsDEM Workbench.

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 higher) or the Java 2 Software Development Kit, Standard Edition, 1.4.0 (or higher) must be installed on the target machine. Visit the Web site http://java.sun.com to download the required Java release. Note, Java releases below 1.4 cannot be used to launch DIAsDEM Workbench because it heavily depends on the support of regular expressions by the package `java.util.regex`. It is strongly recommended using the latest Java release to profit from steady and noticeable performance improvements.

2.2 Unix/Linux

1. Visit the Web site http://www.hypknowsys.org and follow the instructions to download the compressed archive file `DIAsDEM.workbench22.tar.gz`.

2. Create a directory for DIAsDEM Workbench (e.g., `/home/kwinkler/diasdem`) and copy the file `DIAsDEM.workbench22.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 archive by submitting the following two commands at the prompt:

```
/home/kwinkler/diasdem> gunzip DIAsDEM.workbench22.tar.gz
/home/kwinkler/diasdem> tar -xf DIAsDEM.workbench22.tar
```

4. Using any common text editor, modify the environment variables `JAVA_HOME` and `DIAsDEM_HOME` in both shell scripts `DIAsDEM.workbench22/bin/diasdemgui` and `DIAsDEM.workbench22/bin/diasdembatch` (e.g., in the directory `/home/kwinkler/diasdem`) according to your system. For example, if Java is installed in the directory `/usr/lib/j2sdk1.4.0_01` and the file `DIAsDEM.workbench22.tar.gz` was uncompressed in the directory `/home/kwinkler/diasdem`, these two environment variables have to be set as follows:

```
DIAsDEM_HOME=/home/kwinkler/diasdem/DIAsDEM.workbench22
JAVA_HOME=/usr/lib/j2sdk1.4.0_01
```

5. Make sure that both shell scripts `DIAsDEM.workbench22/bin/diasdemgui` and `DIAsDEM.workbench22/bin/diasdembatch` (e.g., in the directory `/home/kwinkler/diasdem`) are executable files:

```
/home/kwinkler/diasdem> chmod a+x
DIAsDEM.workbench22/bin/diasdemgui
/home/kwinkler/diasdem> chmod a+x
DIAsDEM.workbench22/bin/diasdembatch
```

6. Thereafter, the graphical user interface of DIAsDEM Workbench can be launched by executing the shell script `DIAsDEM.workbench22/bin/diasdemgui` (e.g., below the directory `/home/kwinkler/diasdem`).

```
/home/kwinkler/diasdem> DIAsDEM.workbench22/bin/diasdemgui
```

7. The command line batch script processor of DIAsDEM Workbench can be launched by executing the shell script `DIAsDEM.workbench22/bin/diasdembatch` (e.g., below the directory `/home/kwinkler/diasdem`). For example, the following statement has to be submitted at the command prompt to execute the DIAsDEM batch script `/home/kwinkler/diasdem/exampleScript.dsc` in verbose mode:

```
/home/kwinkler/diasdem> DIAsDEM.workbench22/bin/diasdembatch
/home/kwinkler/diasdem/exampleScript.dsc verbose
```

8. Finally, uncompress the text archive for case 1:

```
/home/kwinkler/diasdem> cd DIAsDEM.workbench22/data/samples/de/case1
/home/.../samples/de/case1> gunzip archive.tar.gz
/home/.../samples/de/case1> tar -xf archive.tar
```

2.3 Windows

1. Visit the Web site http://www.hypknowsys.org and follow the instructions to download the compressed archive file `DIAsDEM.workbench22.zip`. This archive includes exactly the same files and directories as `DIAsDEM.workbench22.tar.gz`.

2. Create a directory for DIAsDEM Workbench (e.g., `C:\Programs\diasdem`) and copy the file `DIAsDEM.workbench22.zip` into this directory.

3. Using, for example, WinZip 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 two environment variables `JAVA_HOME` and `DIAsDEM_HOME` in both batch files `DIAsDEM.workbench22\bin\diasdemgui.bat` and `DIAsDEM.workbench22\bin\diasdembatch.bat` (e.g., below the directory `C:\ Programs\diasdem`) according to your system. For example, if Java is installed in `C:\Programs\Java\j2re1.4.0_01` and the file `DIAsDEM.workbench22.zip` was extracted in the directory `C:\Programs\diasdem`, these two environment variables have to be set as follows:

   ```
   DIAsDEM_HOME=C:\Programs\diasdem\DIAsDEM.workbench22
   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.workbench22\bin\diasdemgui.bat` (e.g., below the directory `C:\Programs\diasdem`).

6. The command line batch script processor of DIAsDEM Workbench can be launched by executing the batch file `DIAsDEM.workbench22\bin\diasdembatch.bat` (e.g., below the directory `C:\ Programs\diasdem`). For example, the following statement has to be submitted at the MS-DOS prompt to execute the DIAsDEM batch script `C:\Programs\diasdem\exampleScript.dsc` in verbose mode:

   ```
   C:\Programs\diasdem> DIAsDEM.workbench22\bin\diasdembatch.bat
   C:\Programs\diasdem\exampleScript.dsc verbose
   ```

7. 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, which should afterwards be double-clicked to start DIAsDEM Workbench. The environment memory allocated to the batch file `diasdembatch.bat` must also be increased. Thereafter, DIAsDEM batch script files can be executed by launching the respective PIF file `diasdembatch.pif` instead of `diasdembatch.bat`:

   ```
   C:\Programs\diasdem> DIAsDEM.workbench22\bin\diasdembatch.pif
   C:\Programs\diasdem\exampleScript.dsc verbose
   ```

8. Finally, uncompress the text archive for case 1 using, for example, WinZip. The file `archive.tar.gz` is located in `DIAsDEM.workbench22\data\samples\de\case1`.

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 (e.g., establishment of branch offices, changes of share capital, or mergers and acquisitions) must be reported to the competent 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.

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 `${SAMPLES_HOME}/de/case1` contains 1146 German Commercial Register entries published by the district court Potsdam in 1999 via its Web site (http://www.amtsgericht-potsdam.org). Each entry announces the foundation of a new company in the Potsdam district. Table 3.1 illustrates the content of the file `${SAMPLES_HOME}/de/case1/file10780.training.txt`. Altogether, 985 text files (`*.training.txt`) are input to the first, interactive, and iterative KDT phase of the DIAsDEM framework. The remaining 161 files, which have the extension `.application.txt`, are automati-

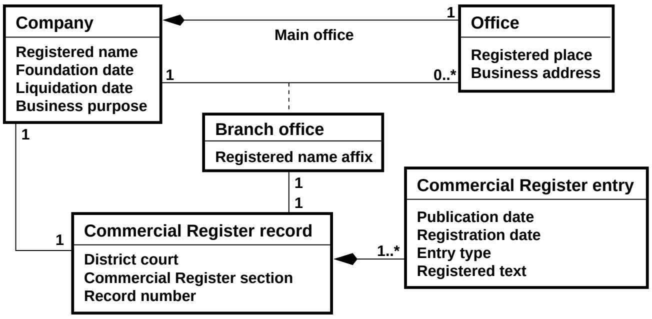


Figure 3.1: Simplified Application Domain (UML Class Diagram)

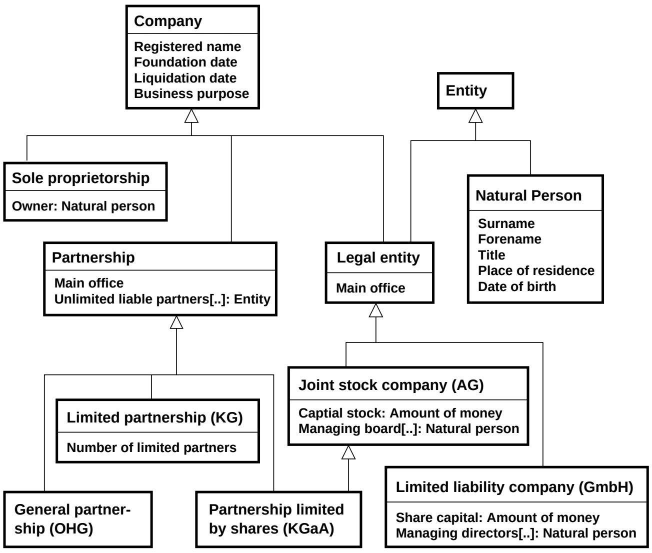


Figure 3.2: Simplified Taxonomy of German Companies (UML Class Diagram)

cally tagged in the second, batch-oriented application phase by applying the previously trained text unit clusterer. All files are ISO-8856-1 encoded and have Unix/Linux line feeds. However, each entry is stored in a single line to avoid line feed related problems. A concise list of relevant German vocabulary based on [PDV00] is available on page 112.

3.2 Text Pre-Processing in the KDT Phase

In the following screen shots of this case study, the directory `/home/kwinkler/diasdem/DIAsDEM.workbench22` corresponds to `${DIAsDEM_HOME}`. `${PROJECT_HOME}` corresponds to `/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject` in the sections describing the interactive and iterative KDT phase of the DIAsDEM framework. All file and directory names are Unix/Linux-based in this case study.

Create a new local directory `${PROJECT_HOME}` on your machine, which can be used to store all files related to this case. Additionally, copy the entire directory template for KDT phase projects, referred to as training projects, into the directory `${PROJECT_HOME}`:

```
DIAsDEM.cases/tutorial> pwd
/home/kwinkler/diasdem/DIAsDEM.cases/tutorial
DIAsDEM.cases/tutorial> cp -R ../../DIAsDEM.workbench22/data/templates/trainingProject .
DIAsDEM.cases/tutorial> ls
trainingProject
```

> 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: German Commercial Register Entry

The template for KDT phase projects comprises empty directories that are populated during this case study. For example, intermediate DIAsDEM documents are stored in the subdirectory `${PROJECT_HOME}/inputCollection`. At the end of this case study, semantically annotated XML documents that correspond to the training text documents are copied into the subdirectory `${PROJECT_HOME}/outputXmlDocuments`.

Make `${DIAsDEM_HOME}` your current working directory and start DIAsDEM Workbench by executing the shell script `${DIAsDEM_HOME}/bin/diasdemgui` on a Unix/Linux system. If you are working on a Windows machine, simply double-click the batch file `diasdemgui.bat`. Figure 1.1 on page 2 depicts DIAsDEM Workbench after startup.

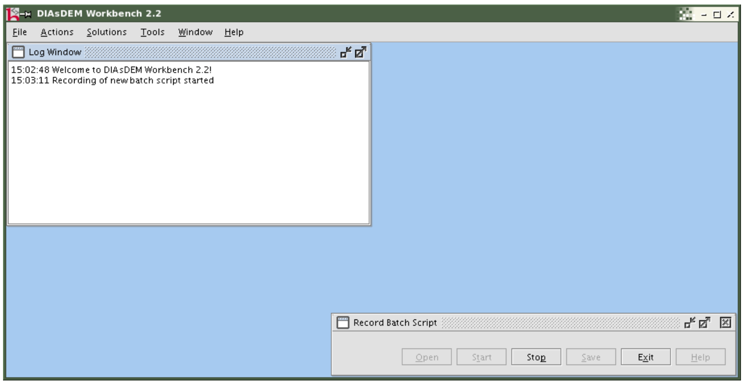
The file `DIAsDEM.plugins` in `${DIAsDEM_HOME}` is a list of Java class names. Each class name corresponds to a DIAsDEM Workbench plug-in that is initialized during the start-up of DIAsDEM Workbench. `DIAsDEM.config` is another file created by DIAsDEM Workbench to store various, project-independent settings. Select *Tools* → *Options* to inspect and alter these settings, respectively. For example, you can choose your preferred Web browser, XML file viewer (e.g., the Web browser), and text file editor in the tab *External Programs*. Be cautious when editing *GUI Properties* because inappropriate parameter values might cause DIAsDEM Workbench to terminate abnormally.

```
DIAsDEM.cases/tutorial> ls trainingProject
applicationParameters kddProcessIteration1 outputGateDocuments outputSqlScripts
batchScripts         kddProcessIteration2 outputNeex21Files outputXmlDocuments
inputCollection      kddProcessIteration3 outputSampleFiles README
DIAsDEM.cases/tutorial> cd /home/kwinkler/diasdem/DIAsDEM.workbench22
diasdem/DIAsDEM.workbench22> ls
DIAsDEM.plugins bin data doc  lib  src
diasdem/DIAsDEM.workbench22> bin/diasdemgui
diasdem/DIAsDEM.workbench22> ls
DIAsDEM.config DIAsDEM.plugins bin data doc lib  src
diasdem/DIAsDEM.workbench22>
```

3.2.1 Starting the Batch Script Recorder

The entire interactive KDT phase is recorded in a DIAsDEM batch script to enable subsequent automated semantic tagging. After slight adjustments, this recorded script can be employed in the application phase to automatically annotate new Commercial Register entries without time-consuming human intervention. DIAsDEM batch scripts are XML documents conforming to the XML document type definition listed in Subsection 4.2 on page 91. Recording a batch script corresponds to saving all performed scriptable tasks along with their parameter settings in a file for subsequent task automation. After recording a script, it can be modified either in the dedicated *Batch Script Editor* (*Solutions* → *Batch Script Processing* → *Edit Batch Script*) or in any common text editor. Using the *Batch Script Editor*, new scripts can also be created from scratch as well. DIAsDEM Workbench allows executing batch scripts within the graphical user interface (*Solutions* → *Batch Script Processing* → *Execute Batch Script*) and on the command prompt by executing the shell script `${DIAsDEM_HOME}/bin/diasdembatch` in Unix/Linux or the script `${DIAsDEM_HOME}\bin\diasdembatch.bat` in a Windows environment.

Start recording all tasks performed hereafter by selecting *Solutions* → *Batch Script Processing* → *Record Batch Script* and thereafter clicking the button *Start*. Alternatively, tasks could be appended to an existing batch script by clicking the button *Open* and subsequently choosing the desired file. Figure 3.3 depicts DIAsDEM Workbench while recording a batch script. Stopping the current recording session, as well as saving, editing, and executing the recorded batch script is explained in Subsection 3.4.4 and Section 3.5.



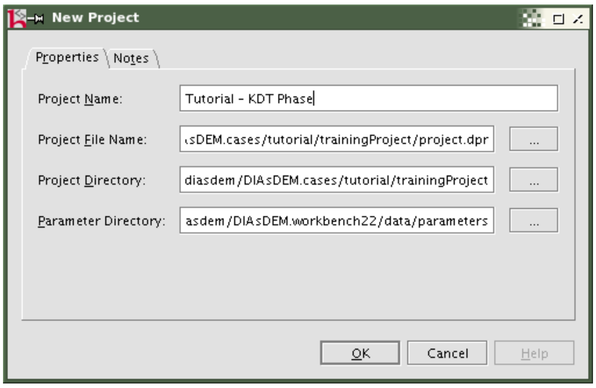
```
15:02:48 Welcome to DIAsDEM Workbench 2.2!
15:03:11 Recording of new batch script started
```

Figure 3.3: DIAsDEM Workbench 2.2 after Starting the Batch Script Recorder

3.2.2 Creating a New Project

Training a text unit clusterer for Commercial Register entries in the KDT phase of our framework constitutes an independent project. All input, intermediate, and output files associated with a project are stored in a single directory (i.e., in `${PROJECT_HOME}`) and its subdirectories to ensure a properly organized file system. Additionally, DIAsDEM Workbench stores project-related data, such as most recently used parameter values, in a DIAsDEM project file with the extension `.dpr` in the directory `${PROJECT_HOME}`. A project must thus either be created or opened before any DIAsDEM task from the menu *Actions*, such as importing texts or clustering text unit vectors, can be performed at all. Consequently, select *File* → *New Project* to create a new DIAsDEM project and enter the following parameters:

Parameter	Value
Project Name	`Tutorial - KDT Phase`
Project File Name	`${PROJECT_HOME}/project.dpr`
Project Directory	`${PROJECT_HOME}`
Parameter Directory	`${PARAMETER_HOME}`



Project Name: Tutorial - KDT Phase

Project File Name: ‹sDEM.cases/tutorial/trainingProject/project.dpr

Project Directory: diasdem/DIAsDEM.cases/tutorial/trainingProject

Parameter Directory: asdem/DIAsDEM.workbench22/data/parameters

Figure 3.4: *New Project* Dialog

Make sure to replace the abbreviations `${PROJECT_HOME}` and `${PARAMETER_HOME}` with the corresponding directories according to your individual installation of DIAsDEM Workbench. Recall, the abbreviation `${PARAMETER_HOME}` denotes the directory `${DIAsDEM_HOME}/data/parameters` in this case study. Therefore, it depends on the installation directory `${DIAsDEM_HOME}` of DIAsDEM Workbench on your machine. Files and directories can always be chosen by clicking the button "..." beside the respective text field and afterwards selecting the desired file using the file dialog. The Java-based file dialog can also be used to create new and rename existing directories, respectively.

Figure 3.4 shows the *New Project* dialog before clicking on *OK*. After clicking the *OK* button, a new project labeled “Tutorial - KDT Phase” is created and immediately opened. Note, the title bar of DIAsDEM Workbench window always indicates the actually opened project.

New Project: Summary

Task: *File* → *New Project* or *Actions* → *Project Management* → *New Project*

Use Case: The user wants to create a new DIAsDEM project to semantically annotate text documents.

Prerequisites: A dedicated local *Project Directory* must have been created for storing project-related files and subdirectories. Template directories for *Project Directory* are provided in `${DIAsDEM_HOME}/data/templates`.

Result: A file *Project File Name* that contains project-related metadata is created in *Project Directory*. Additionally, the project properties *Project Name*, *Project Notes*, *Absolute File Name of Project File*, *Absolute File Name of Project Directory*, and *Absolute File Name of Parameter Directory* are set.

Remarks: The new project can be opened by selecting *File* → *Open Project* and thereafter choosing *Project File Name*. Opened projects can be closed by selecting *File* → *Close Project*.

New Project: Parameters

Project Name: Name of the new DIAsDEM project that might include blank spaces; default value: <`DefaultProjectName`>

Project File Name: Valid local file name of project file to be created in *Project Directory*; file extension: `.dpr`

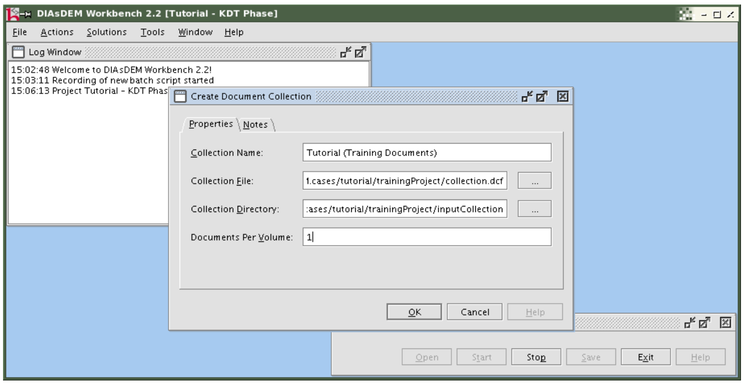
Project Directory: Existing local directory that should contain *Project File Name* and all project-related files; corresponds to `${PROJECT_HOME}` in this case study

Parameter Directory: Existing local directory whose subdirectories contain default parameter files of default DIAsDEM tasks; corresponds to the directory `${PARAMETER_HOME}` in this case study

3.2.3 Creating a Document Collection

Text documents are considered a collection of related documents if they belong to the same application domain and should be semantically tagged. DIAsDEM Workbench is only capable of processing one collection at a time. Each document collection is identified by a specific collection file with extension `.dcf`. This *Collection File* contains metadata about the corresponding archive and references to all DIAsDEM documents constituting the collection. Before texts can be imported into a collection for subsequent processing, a new collection must be created. Therefore, select *Actions* → *Prepare Data Set* → *Create Document Collection* and input the following parameters:

Parameter	Value
Collection Name	`Tutorial (Training Documents)`
Collection File	`${PROJECT_HOME}/collection.dcf`
Collection Directory	`${PROJECT_HOME}/inputCollection`
Documents Per Volume	`1`



```
15:02:48 Welcome to DIAsDEM Workbench 2.2!
15:03:11 Recording of new batch script started
15:06:13 Project Tutorial - KDT Phas
```

Collection Name: Tutorial (Training Documents)

Collection File: 1.cases/tutorial/trainingProject/collection.dcf

Collection Directory: :ases/tutorial/trainingProject/inputCollection

Documents Per Volume: 1

Figure 3.5: *Create Document Collection* Dialog

Click on *OK* to create a new collection file. Thereafter, the directory `${PROJECT_HOME}` contains the files `collection.dcf` and `collection.dcf.files`. In the remainder of this case study, `collection.dcf` is referred to as *Collection File*. It uniquely identifies the corresponding archive of Commercial Register entries and contains relevant metadata. Each *Collection File* is accompanied by an auxiliary file with extention `.dcf.files` (e.g., `collection.dcf.files`). This file only contains absolute file names of DIAsDEM document volume files comprising the collection. Both *Collection File* and its auxiliary file should neither be modified nor deleted manually.

The default implementation of DIAsDEM Workbench is file-based. Each DIAsDEM collection consists of multiple so-called DIAsDEM volumes that in turn include one or more DIAsDEM documents. In this case study, one document is stored per volume file to improve clarity. Each DIAsDEM document contains the initially imported text, both the original and the processed text units (e.g., sentences), as well as possibly rollback text units, and structured metadata. In the default implementation, DIAsDEM volumes are XML documents that conform to the XML document type definition listed in Subsection 4.1 on page 90.

Create Document Collection: Summary

Task: *Actions → Prepare Data Set → Create Document Collection*

Use Case: The user wants to create a new DIAsDEM document collection to semantically annotate text documents.

Prerequisites: A dedicated local *Collection Directory* must have been created for storing DIAsDEM documents in DIAsDEM volumes.

Result: A new DIAsDEM collection is created in *Collection Directory.* It is labeled *Collection Name* and can be referenced by *Collection File.* Additionally, the project properties *Default Collection File* and *Default Collection Directory* are set.

Remarks: Most subsequent processing modules of DIAsDEM Workbench require a specific *Collection File* as an input parameter. Text documents can hereafter be imported into the new collection by selecting *Actions → Prepare Data Set → Import Plain Text Files.*

Create Document Collection: Parameters

Collection Name: Name of the new document collection that might include blank spaces

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

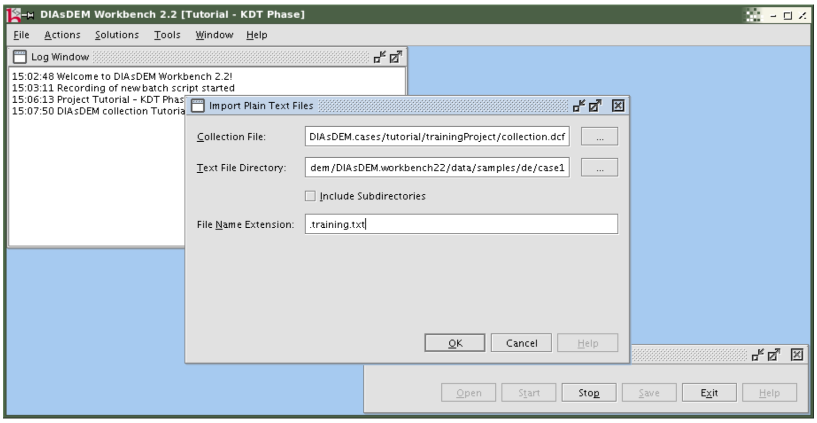
Collection Directory: Existing local directory that subsequently contains DIAsDEM volume files; default value: project property *Default Collection Directory*

Documents Per Volume: Number of DIAsDEM documents stored in a single DIAsDEM volume file; default value: 10

3.2.4 Importing Plain Text Files

Commercial Register entries provided for case 1 are plain text files. They are imported into the new, up to now empty document collection by selecting *Actions* → *Prepare Data Set* → *Import Plain Text Files*. Please provide the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Text File Directory	`${SAMPLES_HOME}/de/case1`
	`Disabled: Include Subdirectories`
File Name Extension	`.training.txt`



15:02:48 Welcome to DIAsDEM Workbench 2.2!
15:03:11 Recording of new batch script started
15:06:13 Project Tutorial - KDT Phas
15:07:50 DIAsDEM collection Tutoria

Collection File: DIAsDEM.cases/tutorial/trainingProject/collection.dcf

Text File Directory: dem/DIAsDEM.workbench22/data/samples/de/case1

☐ Include Subdirectories

File Name Extension: .training.txt

Figure 3.6: *Import Plain Text Files* Dialog

Click the *OK* button to start importing altogether 985 text files with *File Name Extension* from *Text File Directory*. These text files are imported into the new DIAsDEM collection referenced by *Collection File*. Check the content of the subdirectory `${PROJECT_HOME}/inputCollection`:

```
DIAsDEM.cases/tutorial/trainingProject> ls -l inputCollection | more
-rw-r--r-- 1 kwinkler users 1575 2007-08-12 15:49 DefaultDIAsDEMvolume.dtd
...
-rw-r--r-- 1 kwinkler users 1690 2007-08-12 15:56 volume100000.xml
-rw-r--r-- 1 kwinkler users 2037 2007-08-12 15:56 volume100001.xml
...
-rw-r--r-- 1 kwinkler users 1509 2007-08-12 15:56 volume100984.xml
```

`DefaultDIAsDEMvolume.dtd` is the XML document type definition of volume files. For example, the DIAsDEM document that corresponds to the Commercial Register entry listed in Table 3.1 (i.e., `${SAMPLES_HOME}/de/case1/file10780.training.txt`) is stored in volume file `${PROJECT_HOME}/inputCollection/volume100878.xml`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DefaultDIAsDEMvolume SYSTEM "DefaultDIAsDEMvolume.dtd">
<DefaultDIAsDEMvolume NumberOfDocuments="1">
  <DefaultDIAsDEMdocument NumberOfTextUnitsLayers="0">
    <MetaData>
      <Name>DiasdemDocumentID</Name>
      <Content>/home/kwinkler/.../inputCollection/volume100878.xml:0</Content>
    </MetaData>
    <MetaData>
      <Name>SourceFile</Name>
      <Content>/home/kwinkler/.../de/case1/file10780.training.txt</Content>
    </MetaData>
    <OriginalText>Der Handel mit Waren aller Art sowie Import und Export. Der
    Dienstleistungsbereich bezieht sich auf ... Bundesanzeiger.</OriginalText>
  </DefaultDIAsDEMdocument>
</DefaultDIAsDEMvolume>
```

In the default implementation of our workbench, the metadata attribute `DiasdemDocumentID` is created by concatenating the absolute file name of the volume file, a colon, and the index of the respective document (0, 1, . . .) within its volume. Note, the original text will not be modified by any subsequently performed tasks.

Import Plain Text Files: Summary

Task: *Actions → Prepare Data Set → Import Plain 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 or its subdirectories.

Prerequisites: Each text file contains exactly one text document.

Result: Files in *Text File Directory* whose file names end with *File Name Extension* are imported into *Collection File* and stored as DIAsDEM volumes conforming to XML DTD `DefaultDIAsDEMvolume.dtd` described in Subsection 4.1 on page 90. Additionally, the project properties *Default Collection File* and *Default Text File Directory* are set.

Remarks: Texts can be imported from one directory after the other. Alternatively, the user might implement a specialized DIAsDEM Workbench plug-in to import text documents into a collection. In this case, additional metadata can be included in DIAsDEM documents as well.

Import Plain Text Files: Parameters

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

Text File Directory: Existing local directory that contains text files to be imported; if *Include Subdirectories* is enabled, text files stored in subdirectories of *Text File Directory* are also imported; default value: project property *Default Text File Directory*

File Name Extension: Text files are only imported if they have the specified file name extension; default value: `.txt`

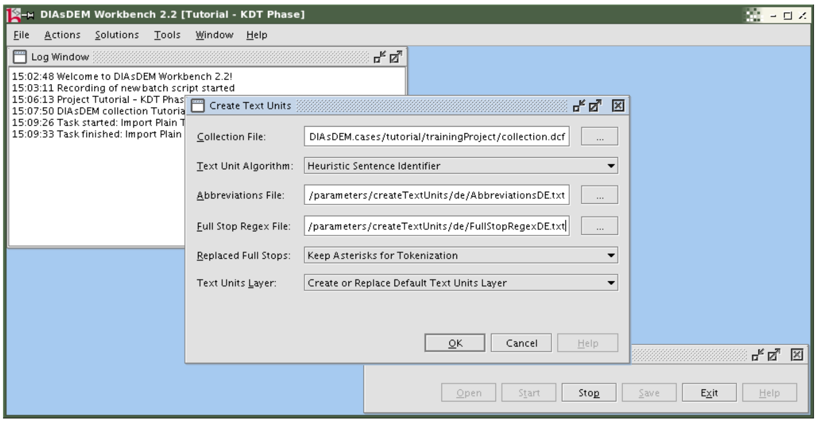
3.2.5 Creating Text Units

After importing texts into a collection, the data pre-processing phase starts with identifying and separating text units. In this case study, sentences of Commercial Register entries correspond to text units. Hence, only sentences are semantically annotated by DIAsDEM Workbench in the course of this case study. To proceed, select *Actions → Prepare Data Set → 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`
Text Units Layer	`Create or Replace Default Text Units Layer`

Click the *OK* button to identify and separate altogether 9,254 sentences in DIAsDEM documents. The Heuristic Sentence Identifier first replaces full stops in abbreviations (e.g., “z.B.”) listed in *Abbreviations File* with asterisks. However, only abbreviations are replaced that either occur at the beginning of the text or that follow one of certain special characters (i.e., blank space and `(),;:/-'"`). Thereafter, regular expressions contained in *Full Stop Regex File* are matched against the original text. These regular expressions match full stops that are no sentence boundaries (e.g., “01.01.2002”) and replace all matches with asterisks as well. These textual parameter files can be edited to include additional domain knowledge.

Using, for example, the built-in, prototypical *Tools → Miscellaneous → XML Document Viewer*, have a look at the content of DIAsDEM volume file `${PROJECT_HOME}/inputCollection/file10878.txt.xml`. Its DIAsDEM document has been extended by the new section `<TextUnitsLayer>`, which consists of two subsections. The elements `<OriginalTextUnit>` of the subsection `<OriginalTextUnits>` mark up single, original



```
15:02:48 Welcome to DIAsDEM Workbench 2.2!
15:03:11 Recording of new batch script started
15:06:13 Project Tutorial - KDT Phas
15:07:50 DIAsDEM collection Tutoria
15:09:26 Task started: Import Plain T
15:09:33 Task finished: Import Plain
```

Collection File: DIAsDEM.cases/tutorial/trainingProject/collection.dcf

Text Unit Algorithm: Heuristic Sentence Identifier

Abbreviations File: /parameters/createTextUnits/de/AbbreviationsDE.txt

Full Stop Regex File: /parameters/createTextUnits/de/FullStopRegexDE.txt

Replaced Full Stops: Keep Asterisks for Tokenization

Text Units Layer: Create or Replace Default Text Units Layer

Figure 3.7: *Create Text Units* Dialog

sentences. Elements `<ProcessedTextUnit>` of subsection `<ProcessedTextUnits>` mark up single, processed sentences. All sentences of the former subsection are retained unmodified whereas the content of the latter subsection is modified by tasks performed hereafter.

In general, the DIAsDEM framework supports multiple structural views on texts by introducing the notion of a text units layer. For example, one might define three text units layers to semantically annotate each document at the text level, at the paragraph level, and at sentence level. However, DIAsDEM Workbench 2.2 does not fully support nested semantic tagging of text documents. Instead, tasks always process text units associated with the default layer and output accordingly tagged XML documents only. In this case study, the default text units layer 0 provides a structural view on sentences of each document. The index of the default text units layer is determined by the setting of project property *Index of Default Active Text Units Layer in DIAsDEM Documents*. It can be modified in the *Project Properties* tab of the *Tools* → *Options* dialog.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DefaultDIAsDEMvolume SYSTEM "DefaultDIAsDEMvolume.dtd">
<DefaultDIAsDEMvolume NumberOfDocuments="1">
  <DefaultDIAsDEMdocument NumberOfTextUnitsLayers="0"> ...
    <TextUnitsLayer TextUnitsLayerID="0" TextUnitsDescription="Algorithm:
    HEURISTIC_SENTENCE_IDENTIFIER">
      <OriginalTextUnits>
        <OriginalTextUnit TextUnitID="0" BeginIndex="0" EndIndex="55">Der
        Handel mit Waren aller Art sowie Import und Export.</OriginalTextUnit>
        ... <OriginalTextUnit TextUnitID="2" BeginIndex="138" EndIndex="162">
        Stammkapital: 50.000 DM.</OriginalTextUnit> ... <OriginalTextUnit
        TextUnitID="9" BeginIndex="867" EndIndex="963"> Marion Marcella Adolph
        geb. Priester, 22.03.1957, Offenbach, ist zur Geschäftsführerin
        bestellt.</OriginalTextUnit> ...
      </OriginalTextUnits>
      <ProcessedTextUnits>
        <ProcessedTextUnit TextUnitID="0">Der Handel mit Waren aller Art
        sowie Import und Export.</ProcessedTextUnit> ... <ProcessedTextUnit
        TextUnitID="2">Stammkapital: 50*000 DM.</ProcessedTextUnit> ...
        <ProcessedTextUnit TextUnitID="9">Marion Marcella Adolph geb* Priester,
        22*03*1957, Offenbach, ist zur Geschäftsführerin bestellt.
        </ProcessedTextUnit> ...
      </ProcessedTextUnits>
    </TextUnitsLayer>
  </DefaultDIAsDEMdocument>
</DefaultDIAsDEMvolume>
```

Asterisks that occur within the section `<ProcessedTextUnits>` either replace full stops in an abbreviation listed in *Abbreviations File* (e.g., `geb.`) or full stops matched by a regular expression in *Full Stop Regex File*. For example, the date literal `22.03.1957` is matched by the expression `([0-9]{1,2})\.([\ ]*[0-9]{1,2})\.([\ ]*[0-9]{2,4})`. Therefore, 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

Task: *Actions → Prepare Data Set → Create Text Units*

Use Case: The user must pre-process imported texts as part of the DIAsDEM KDT process for semantic tagging of domain-specific texts archives. Creating text units is the mandatory pre-processing step 1 of 4.

Prerequisites: Texts must have been imported into the DIAsDEM collection.

Result: The specified text units layer of each document is created or replaced. Elements of the sections `<OriginalTextUnits>` and `<ProcessedTextUnits>` mark up either single sentences (*Heuristic Sentence Identifier*) or the entire text (*Text as a Single Text Unit*). If *Create or Replace Default Text Units Layer* is enabled, an existing default layer is completely replaced. Additionally, the project properties *Default Collection File*, *Default Abbreviations File*, and *Default Full Stop Regex File* are set and updated, respectively.

Remarks: Creating text units is a prerequisite for the remaining mandatory pre-processing steps 2 (i.e., tokenization) through 4 (i.e., lemmatization). Take the following side effect into consideration: All asterisks that occur in imported texts are eventually replaced by full stops.

Create Text Units: Parameters

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

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

Abbreviations File: Valid local file name of existing file that contains known abbreviations in the format described in Subsection 4.3.1 on page 91; file extension: `.txt`; default value: project property *Default Abbreviations File*

Full Stop Regex File: Valid local file name of existing file that contains regular expressions in the format described in Subsection 4.3.1 on page 91; file extension: `.txt`; default value: project property *Default Full Stop Regex File*

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

Text Units Layer: If the recommended option *Create or Replace Default Text Units Layer* is enabled, the default layer is created and replaced, respectively. Otherwise, an additional text units layer is added to each document. Note, the default text units layer index is determined by the project property *Index of Default Active Text Units Layer in DIAsDEM Documents.*

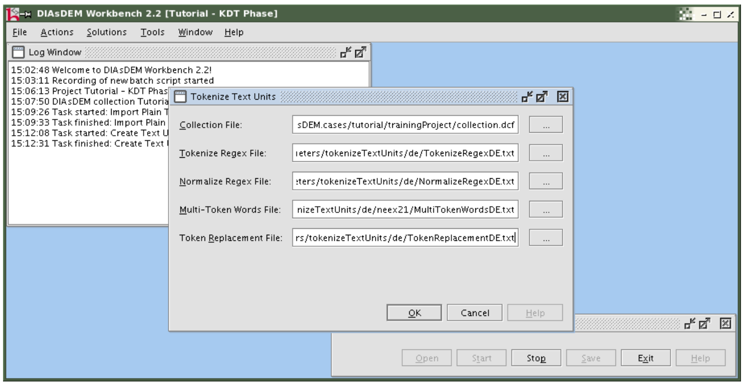
3.2.6 Tokenizing Text Units

After creating text units, tokenizing them constitutes the second pre-processing step. During tokenization, text units are decomposed into individual words and tokens, respectively. In addition, text units are normalized to map, for example, date literals appearing in many formats (e.g., “1 Jan 2003” and “1.1.2003”) onto a canonical representation (e.g., “01.01.2003”). Moreover, multi-token terms that contain blank spaces (e.g., “for example”) are identified to subsequently process them as single tokens. Select *Actions* → *Prepare Data Set* → *Tokenize 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`
	`/neex21/MultiTokenWordsDE.txt`

Token Replacement File	`${PARAMETER_HOME}/createTextUnits/de/TokenReplacementDE.txt`



```
15:02:48 Welcome to DIAsDEM Workbench 2.2!
15:03:11 Recording of new batch script started
15:06:13 Project Tutorial - KDT Phas
15:07:50 DIAsDEM collection Tutoria
15:09:26 Task started: Import Plain T
15:09:33 Task finished: Import Plain
15:12:08 Task started: Create Text U
15:12:31 Task finished: Create Text
```

Collection File: sDEM.cases/tutorial/trainingProject/collection.dcf

Tokenize Regex File: ıeters/tokenizeTextUnits/de/TokenizeRegexDE.txt

Normalize Regex File: :ters/tokenizeTextUnits/de/NormalizeRegexDE.txt

Multi-Token Words File: nizeTextUnits/de/neex21/MultiTokenWordsDE.txt

Token Replacement File: rs/tokenizeTextUnits/de/TokenReplacementDE.txt

Figure 3.8: *Tokenize Text Units* Dialog

Click on *OK* to tokenize and normalize text units, as well as to identify and replace multi-token terms. All processing steps can be fully parameterized by editing regular expressions or multi-token terms in the respective parameter files. The heuristic normalization algorithm does not separate asterisks from their surrounding characters because asterisks correspond to previously replaced full stops. 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. Furthermore, blank spaces in multi-token terms listed in *Multi-Token Words File* (e.g., “for example”) are replaced by underscores (e.g., “for_example”) to create a single token. Moreover, the search and replace operations specified in *Token Replacement File* are executed. For example, composite nouns (e.g., “Gewinnanstieg”) could be split (e.g., “Gewinn Anstieg”), or English clitics (e.g., “wont” and “’ ll”) can be expanded (e.g., “will not” and “will”). Finally, all asterisks that occur in the section `<ProcessedTextUnits>` are replaced by full stops.

Check the content of file `${PROJECT_HOME}/inputCollection/volume100878.xml`: Firstly, the section `<ProcessedTextUnits>` of this DIAsDEM document has been updated. After tokenization, its elements `<ProcessedTextUnit>` mark up single, tokenized and normalized sentences. Secondly, a section `<RollbackTextUnits>` has been added to the default text units layer of each document. By executing *Actions → Miscellaneous → Rollback Processed Text Units*, the content of `<RollbackTextUnits>` can be copied into `<ProcessedTextUnits>` to undo the effects of one preceding task. Note, DIAsDEM Workbench 2.2 supports three different rollback policy options: No backup at all (0), rollback of the immediately preceding task (1, default setting), and rollback of any preceding task (2). The active rollback policy is determined by the project property *Rollback Option (0, 1, 2) for ProcessedTextUnits in DIAsDEM Documents.*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DefaultDIAsDEMvolume SYSTEM "DefaultDIAsDEMvolume.dtd">
<DefaultDIAsDEMvolume NumberOfDocuments="1">
  <DefaultDIAsDEMdocument NumberOfTextUnitsLayers="0"> ...
    <TextUnitsLayer TextUnitsLayerID="0" TextUnitsDescription="Algorithm:
    HEURISTIC_SENTENCE_IDENTIFIER"> ...
      <ProcessedTextUnits>
        <ProcessedTextUnit TextUnitID="0">Der Handel mit Waren aller Art sowie
        Import und Export .</ProcessedTextUnit> ... <ProcessedTextUnit
        TextUnitID="2">Stammkapital : 50000 DEM .</ProcessedTextUnit>
        <ProcessedTextUnit TextUnitID="3">Gesellschaft_mit_beschränkter_Haftung
        .</ProcessedTextUnit> ... <ProcessedTextUnit TextUnitID="9">Marion
        Marcella Adolph geb. Priester , 22.03.1957 , Offenbach , ist zur
        Geschäftsführerin bestellt .</ProcessedTextUnit> ...
      </ProcessedTextUnits>
      <RollbackTextUnits RollbackID="0">
        <ProcessedTextUnit TextUnitID="0">Der Handel mit Waren aller Art
        sowie Import und Export.</ProcessedTextUnit> ... <ProcessedTextUnit
        TextUnitID="2">Stammkapital: 50*000 DM.</ProcessedTextUnit> ...
        <ProcessedTextUnit TextUnitID="9">Marion Marcella Adolph geb* Priester,
        22*03*1957, Offenbach, ist zur Geschäftsführerin bestellt.
        </ProcessedTextUnit> ...
      </RollbackTextUnits>
    </TextUnitsLayer>
  </DefaultDIAsDEMdocument>
</DefaultDIAsDEMvolume>
```

The task works as follows: Firstly, regular expressions listed in *Tokenization Regex File* are matched against each processed 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, regular expressions listed in *Normalization Regex File* are matched against processed text units. Analogously, matching character subsequences are substituted by the corresponding replacement string. Thirdly, multi-token terms included in *Multi-Token Words File* are looked up in text units. Identified multi-token terms are reduced to single tokens by replacing their inner blank spaces with underscores.

Fourthly, the token search and replace operations specified in *Token Replacement File* are executed.

Tokenize Text Units: Summary

Task: *Actions → Prepare Data Set → Tokenize Text Units*

Use Case: The user must pre-process all imported texts as part of the DIAsDEM KDT process for semantic tagging of domain-specific texts archives. Tokenizing text units is mandatory pre-processing step 2 of 4.

Prerequisites: The default text units layer of each DIAsDEM document must contain the section `<ProcessedTextUnits>`. Text units should have been created in the DIAsDEM collection.

Result: Elements of the section `<ProcessedTextUnits>` mark up tokenized and normalized text units. Inner blank spaces in multi-token terms have been replaced with underscores, and all asterisks have been replaced with full stops. Additionally, the project properties *Default Collection File*, *Default Normalize Regex File*, *Default Tokenize Regex File*, *Default Multi-Token Words File*, and *Default Token Replacement File* are set and updated, respectively.

Tokenize Text Units: Parameters

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

Tokenize Regex File: Valid local file name of existing file that contains regular expressions in the format described in Subsection 4.3.2 on page 92; file extension: `.txt`; default value: project property *Default Tokenize Regex File*;

Normalize Regex File: Valid local file name of existing file that contains regular expressions in the format described in Subsection 4.3.2 on page 92; file extension: `.txt`; default value: project property *Default Normalize Regex File*

Multi-Token Words File: Valid local file name of existing file that contains multi-token terms in the format described in Subsection 4.3.2 on page 92; file extension: `.txt`; default value: project property *Default Multi-Token Words File*

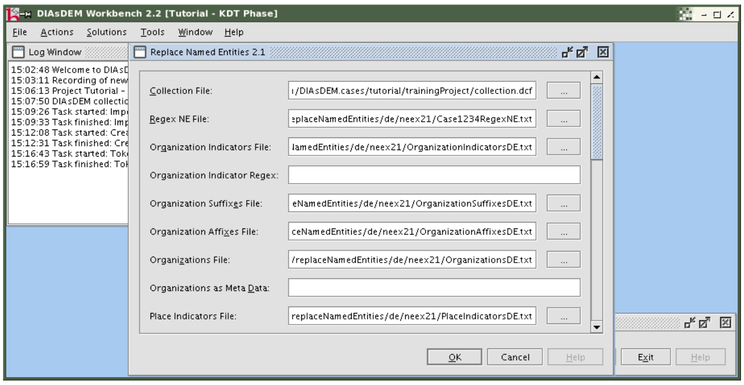
Token Replacement File: Valid local file name of existing file that contains tokens to be searched and replaced in the format described in Subsection 4.3.2 on page 92; file extension: `.txt`; default value: project property *Default Token Replacement File*

3.2.7 Replacing Named Entities with NEEX 2.1

After creating and tokenizing text units, identifying named entities and replacing them with placeholders constitutes the third pre-processing step. 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”, and “Leipzig” instantiates the named entity type “place”. Based on lists, regular expressions, and extraction rules, the Named Entity Extractor of DIAsDEM Workbench 2.2 is capable of identifying instances of many named entities types, among them “person”, “company”, “company_relocation”, “number”, “date”, “time”, “amount_of_money”, “paragraph”, “email”, “url”, “organization_id”, “document_id”, “court”, “postal_code”, “street”, “isin”, and “wkn”. Continue by selecting *Actions → Prepare Data Set → Replace Named Entities 2.1* and typing in the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Regex NE File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/Case1234RegexNE.txt`
Organization Indicators File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/OrganizationIndicatorsDE.txt`
Organization Indicator Regex	
Organization Suffixes File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/OrganizationSuffixesDE.txt`
Organization Affixes File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/OrganizationAffixesDE.txt`
Organizations File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/OrganizationsDE.txt`
Organizations as Meta Data	
Place Indicators File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/PlaceIndicatorsDE.txt`
Places File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/PlacesDE.txt`
Place Affixes File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/PlaceAffixesDE.txt`
Person Name Indicators File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/PersonNameIndicatorsDE.txt`
Titles File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/TitlesDE.txt`
Forenames File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/ForenamesDE.txt`
Middle Initials File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/MiddleInitialsDE.txt`
Surnames File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/SurnamesDE.txt`

Surname Suffixes File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/SurnameSuffixesDE.txt`
Name Affixes File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/NameAffixesDE.txt`
Professions File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/ProfessionsDE.txt`
Street Exceptions File	
Street Suffixes File	
Street Prefix Token Regex	
Street Affix Token Regex	
Street Exclusion Token Regex	
Min. Tokens in Street	
Composite NE File	`${PARAMETER_HOME}/replaceNamedEntities/de/neex21/Case1234CompositeNE.txt`
Debugging Files Directory	`${PROJECT_HOME}/outputNeex21Files`
Advanced Options	`Disabled: Extract Basic Named Entities of Type ‘Street’`


DIAsDEM Workbench 2.2 [Tutorial - KDT Phase]
File
Actions
Solutions
Tools
Window
Help
Log Window
Replace Named Entities 2.1
...
OK
Cancel
Help
Exit
Help

```
15:02:48 Welcome to DIAsD
15:03:11 Recording of new
15:06:13 Project Tutorial -
15:07:50 DIAsDEM collectic
15:09:26 Task started: Imp
15:09:33 Task finished: Imp
15:12:08 Task started: Cre
15:12:31 Task finished: Cre
15:16:43 Task started: Tok
15:16:59 Task finished: Tok
```

Collection File: ı/DIAsDEM.cases/tutorial/trainingProject/collection.dcf

Regex NE File: ɛplaceNamedEntities/de/neex21/Case1234RegexNE.txt

Organization Indicators File: lamedEntities/de/neex21/OrganizationIndicatorsDE.txt

Organization Indicator Regex:

Organization Suffixes File: eNamedEntities/de/neex21/OrganizationSuffixesDE.txt

Organization Affixes File: ceNamedEntities/de/neex21/OrganizationAffixesDE.txt

Organizations File: /replaceNamedEntities/de/neex21/OrganizationsDE.txt

Organizations as Meta Data:

Place Indicators File: replaceNamedEntities/de/neex21/PlaceIndicatorsDE.txt

Figure 3.9: *Replace Named Entities 2.1* Dialog

Click the *OK* button to identify and replace named entities in processed text units. Thereafter, open the file `${PROJECT_HOME}/inputCollection/volume100878.xml`. The default text units layer of its DIAsDEM document has been extended by the new section `<NamedEntities>` whose elements `<NamedEntity>` mark up extracted named entities. Elements of the section `<ProcessedTextUnits>` now mark up tokenized and normalized sentences, which optionally contain named entity placeholders. Each placeholder tag `<NeRef>` references its associated named entity in section `<NamedEntity>` via the attribute `NeID`. Note, the section `<RollbackTextUnits>` has also been updated. Hence, you might undo the effects of the performed named entity extraction task, if necessary.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DefaultDIAsDEMvolume SYSTEM "DefaultDIAsDEMvolume.dtd">
```

```
<DefaultDIAsDEMvolume NumberOfDocuments="1">
  <DefaultDIAsDEMdocument NumberOfTextUnitsLayers="0"> ...
    <TextUnitsLayer TextUnitsLayerID="0" TextUnitsDescription="Algorithm:
    HEURISTIC_SENTENCE_IDENTIFIER"> ...
      <ProcessedTextUnits>
        <ProcessedTextUnit TextUnitID="0">Der Handel mit Waren aller Art sowie
        Import und Export .</ProcessedTextUnit> ... <ProcessedTextUnit
        TextUnitID="2">Stammkapital : <NeRef NeID="0" /> .</ProcessedTextUnit>
        <ProcessedTextUnit TextUnitID="3">Gesellschaft_mit_beschränkter_Haftung
        .</ProcessedTextUnit> ... <ProcessedTextUnit TextUnitID="9"><NeRef
        NeID="16" />, ist zur Geschäftsführerin bestellt .</ProcessedTextUnit> ...
      </ProcessedTextUnits> ...
      <NamedEntities>
        <NamedEntity NeID="0" NeType="amount_of_money">50000 DEM</NamedEntity> ...
        <NamedEntity NeID="12" NeType="date">22.03.1957</NamedEntity>
        <NamedEntity NeID="13" NeType="place">Offenbach</NamedEntity>
        <NamedEntity NeID="14" NeType="person_name">Marion Marcella Adolph
        </NamedEntity>
        <NamedEntity NeID="15" NeType="person_name">Priester</NamedEntity>
        <NamedEntity NeID="16" NeType="person">16|null|person|null|Marion
        Marcella Adolph|null|null|Priester|22.03.1957|null|null|null|Offenbach
        |null|null</NamedEntity>
      </NamedEntities>
    </TextUnitsLayer>
  </DefaultDIAsDEMdocument>
</DefaultDIAsDEMvolume>
```

NEEX 2.1 can be fully parameterized by editing the corresponding parameter files. The heuristic Named Entity Extractor of DIAsDEM Workbench 2.2 works as follows:

1. Firstly, regular expressions listed in the parameter file *Regex NE File* are matched against intermediate text units to identify instances of the following named entities types, which are referred to as basic named entities: "number", "date", "time", "amount_of_money", "paragraph", "email", "url", "organization_id", "document_id", "court", "postal_code", "reference_number", "percentage", "newspaper", "wkn" (i.e., German securities identification number), "isin" (i.e., international securities identification number), "stock_exchange", "number_of_shares", and "amount_of_money_per_share".

2. Secondly, instances of the basic named entity type "organization" are identified by employing the parameter files *Organization Indicators File*, *Organization Suffixes File*, *Organization Affixes File*, and *Organizations File.* The latter file contains a list of complete, tokenized names of important organizations that are extracted in any case. To heuristically extract less important organization names, the algorithm initially searches for known organization name suffix tokens (e.g., "Ltd.") and then looks backwards for valid organization indicator tokens, such as "takeover

of", to instantiate a named entity of type "organization". Subsequently, identified organization names are extended if they are followed by organization name affix tokens (e.g., "Worldwide") listed in the respective parameter file.

3. Thirdly, tokens or sequences thereof instantiating the basic named entity type "place" are extracted using the parameter files *Place Indicators File*, *Places File*, and *Place Affixes File*. The algorithm first looks up all tokens in the dictionary of known places comprising, for example, countries like "Germany" and cities like "Berlin". Subsequently, place candidate tokens are extended if they are immediately followed by another known place or a place affix token, such as "Airport". However, even place candidates are only instantiated as named entities of type "place" if they are directly preceded by a known place indicator token, such as "in" or "to".

4. Thereafter, the parameter files *Person Name Indicators File*, *Titles File*, *Forenames File*, *Middle Initials File*, *Surnames File*, *Surname Suffixes File*, and *Name Affixes File* are used to discover instances of the basic named entity "person_name". Each instance corresponds to a contiguous sequence of tokens, each of which instantiates one of the following basic named entity types: academic "title" (e.g., "Dr."), "forename", "middle_initial", "surname", or "name_affix" (e.g., "Sen."). Composite forenames and surnames comprising a hyphen are also identified. Person name candidates are extended if they are immediately followed by a known name affix token or a capitalized token ending with a known surname suffix listed in *Surname Suffixes File*. However, single-token person name candidates are only instantiated if they are directly preceded by a person name indicator token contained in the respective parameter file. If the optional parameter *Professions File* is specified, instances of basic named entity type "profession" are identified in text units that contain at least one instance of named entity type person_name.

5. Constructor rules, which are specified in a workbench-specific syntax in the parameter file *Composite NE File*, are finally applied to intermediate text units that contain identified basic named entities to discover instances of the following composite named entities: "person", "company", "company_relocation", "date_period", "amount_of_money_range", "percentage_range", "equity_stake", "unit_of_company", and "key_figure". Each composite named entity consists of basic named entities that occur in the context of tokens defined by constructor rules. For instance, composite named entities of type "person" can be constructed from the basic named entities "person_name", "date", and "place". If a composite named entity is identified, both ordinary tokens and basic named entity placeholders matched by the rule are substituted by the corresponding composite named entity placeholder. Adding the exemplary rule "<<person_name>> geb. <<person_name>> , <<date>> , <<place>>" to match tokens and instances of basic named entities along with

an appropriate constructor statement to *Composite NE File* maps, for instance, the German token sequence "Marion Marcella Adolph geb. Priester , 22.03.1957 , Offenbach" onto an instance of the composite named entity type "person".

In general, each token might simultaneously instantiate various basic named entities. For instance, the token `"Hagen"` could be a German forename or a German city. Thus, various heuristics are used in conjunction with the parameter files *Person Name Indicators File* and *Place Indicators File* to decide whether a particular token probably instantiates a "place" or is more likely a partial "person_name".

Replace Named Entities 2.1: Summary

Task: *Actions → Prepare Data Set → Replace Named Entities 2.1*

Use Case: The user must pre-process all imported texts as part of the DIAsDEM KDT process for semantic tagging of domain-specific texts archives. Identifying and replacing named entities is the mandatory pre-processing step 3 of 4.

Prerequisites: The default text units layer of each DIAsDEM document must contain the section `<ProcessedTextUnits>`. Elements `<ProcessedTextUnit>` must not contain previously inserted named entity references `<NeRef>`. Text units should have been created and tokenized in the DIAsDEM collection.

Result: Elements of section `<ProcessedTextUnits>` mark up text units containing placeholders for extracted named entities. The named entities are stored in elements `<NamedEntity>` of section `<NamedEntities>`. Additionally, the project properties that represent default values of input parameters are set and updated on request, respectively.

Replace Named Entities 2.1: Parameters

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

Regex NE File: Valid local file name of existing file that contains regular expressions for identifying basic named entities (e.g., of type "amount_of_money") in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; default value: project property *NEEX 2.1: Default Regex NE File*

Organization Indicators File: Valid local file name of existing file that contains terms that frequently precede organization names (e.g., "acquired") in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; default value: project property *NEEX 2.1: Default Organization Indicators File*

Organization Indicator Regex: Optional input parameter comprising a syntactically valid regular expression (e.g., `\d+\)`) that matches organization indicator tokens, such as enumerations like "1)" and "2)"; default value: project property *NEEX 2.1: Default Organization Indicator Regex*

Organization Suffixes File: Valid local file name of existing file that contains a list of organizational abbreviations (e.g., "Corp." or "AG") in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; default value: project property *NEEX 2.1: Default Organization Suffixes File*

Organization Affixes File: Valid local file name of existing file that contains a list of terms that frequently follow organization suffixes as part of organization names (e.g., "Import and Export") in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; optional parameter; default value: project property *NEEX 2.1: Default Organization Affixes File*

Organizations File: Valid local file name of existing file that contains a list of complete, tokenized organization names (e.g., "Foo and Partners Ltd.") in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; optional parameter; default value: project property *NEEX 2.1: Default Organizations File*

Organizations as Meta Data: Valid name of metadata attribute in DIAsDEM documents whose values store exactly one complete, tokenized organization name (e.g., "TokenizedNameOfPublishingCompany"); optional parameter; default value: project property *NEEX 2.1: Default Organizations Meta Data Attribute in DIAsDEM Documents*

Place Indicators File: Valid local file name of existing file that contains terms that frequently precede places (e.g., "in" or "to") in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; optional parameter; default value: project property *NEEX 2.1: Default Place Indicators File*

Places File: Valid local file name of existing file that contains a list of places (i.e., cities) in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; default value: project property *NEEX 2.1: Default Places File*

Place Affixes File: Valid local file name of existing file that contains a list of terms that frequently follow places (e.g., districts and names of rivers) as part of the place name in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; optional parameter; default value: project property *NEEX 2.1: Default Place Affixes File*

Person Name Indicators File: Valid local file name of existing file that contains terms that frequently precede person names (e.g., “Mr.” or “with”) in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; optional parameter; default value: project property *NEEX 2.1: Default Person Name Indicators File*

Titles File: Valid local file name of existing file that contains a list of academic and professional titles (e.g., “Prof.” or “Prof. Dr.”) in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; default value: project property *NEEX 2.1: Default Titles File*

Forenames File: Valid local file name of existing file that contains a list of forenames in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; default value: project property *NEEX 2.1: Default Forenames File*

Middle Initials File: Valid local file name of existing file that contains a list of middle initials (e.g., “von”, “de la” or “A.”) in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; default value: project property *NEEX 2.1: Default Middle Initials File*

Surnames File: Valid local file name of existing file that contains a list of surnames in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; default value: project property *NEEX 2.1: Default Surnames File*

Surname Suffixes File: Valid local file name of existing file that contains a list of frequent surname suffixes (e.g., “wicz” or “ova”) in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; default value: project property *NEEX 2.1: Default Surname Suffixes File*

Name Affixes File: Valid local file name of existing file that contains a list of terms that frequently follow person names (e.g., “Ph.D.” or “jr.”) as part of the name in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; optional parameter; default value: project property *NEEX 2.1: Default Name Affixes File*

Professions File: Valid local file name of existing file that contains a list of professions (e.g., “CEO” or “President”) that should be associated with person names in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; optional parameter; default value: project property *NEEX 2.1: Default Professions File*

Composite NE File: Valid local file name of existing file that contains NEEX-specific rules for instantiating composite named entities of type “person”, “companies”, and “company_relocation” in the format described in Subsection 4.3.3 on page 94; file extension: `.txt`; default value: project property *NEEX 2.1: Default Composite NE File*;

Debugging Files Directory: Valid local file name of existing directory for storing debugging files created by NEEX 2.1; optional parameter; default value: project property *NEEX 2.1: Default Directory of Debugging HTML Files*;

Advanced Options: If the option *Extract Basic Named Entities of Type 'Street'* is enabled, instances of basic named entity type "street" are identified. The option *Determine Canonical Forms of Named Entities* cannot be enabled because NEEX 2.1 does not support this operation.

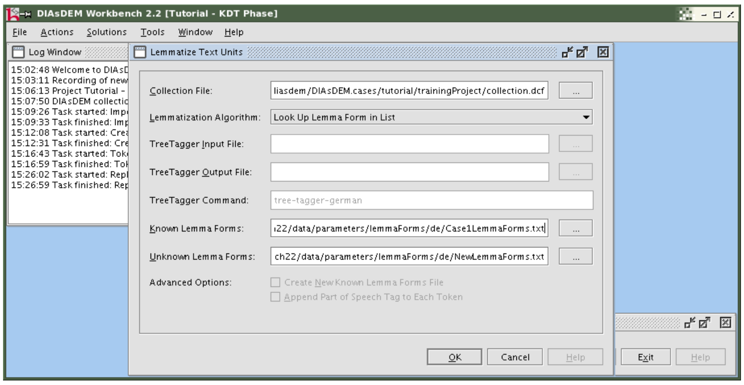
3.2.8 Lemmatizing Text Units

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

DIAsDEM Workbench supports two distinct 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. TreeTagger is a multilingual part-of-speech tagger developed by Helmut Schmid [Sch94]. As of August 2007, TreeTagger for Linux and Solaris can be used for evaluation and research purposes free of charge. Using TreeTagger is the preferred method of lemmatization. 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 a main disadvantage: Lemma forms can only be determined for terms whose grammatical root forms are listed in the file of known lemma forms. Additionally, the syntactical context of term occurrences is not taken into consideration. Select *Actions → Prepare Data Set → Lemmatize Text Units* 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/Case1LemmaForms.txt`

Unknown Lemma Forms	`${PARAMETER_HOME}/lemmaForms/de/NewLemmaForms.txt`
Advanced Options	`Disabled: Create New Known Lemma Forms File`
	`Disabled: Append Part of Speech Tag to Each Token`



15:02:48 Welcome to DIAsD
15:03:11 Recording of new
15:06:13 Project Tutorial -
15:07:50 DIAsDEM collectio
15:09:26 Task started: Imp
15:09:33 Task finished: Imp
15:12:08 Task started: Cre
15:12:31 Task finished: Cre
15:16:43 Task started: Tok
15:16:59 Task finished: Tok
15:26:02 Task started: Repl
15:26:59 Task finished: Rep

Collection File: liasdem/DIAsDEM.cases/tutorial/trainingProject/collection.dcf
Lemmatization Algorithm: Look Up Lemma Form in List
TreeTagger Input File:
TreeTagger Output File:
TreeTagger Command: tree-tagger-german
Known Lemma Forms: ı22/data/parameters/lemmaForms/de/Case1LemmaForms.txt
Unknown Lemma Forms: ch22/data/parameters/lemmaForms/de/NewLemmaForms.txt
Advanced Options: ☐ Create New Known Lemma Forms File
☐ Append Part of Speech Tag to Each Token

Figure 3.10: *Lemmatize Text Units* Dialog

Click the *OK* button to start lemmatizing text units. Thereafter, check the content of the file `${PROJECT_HOME}/inputCollection/volume100878.xml`: The section `<ProcessedTextUnits>` of this DIAsDEM document has been updated. After lemmatization, its elements `<ProcessedTextUnit>` mark up lemma forms and named entity placeholders of identified text units. Furthermore, the section `<RollbackTextUnits>` has been updated to enable a rollback of this task.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DefaultDIAsDEMvolume SYSTEM "DefaultDIAsDEMvolume.dtd">
<DefaultDIAsDEMvolume NumberOfDocuments="1">
  <DefaultDIAsDEMdocument NumberOfTextUnitsLayers="0"> ...
    <TextUnitsLayer TextUnitsLayerID="0" TextUnitsDescription="Algorithm:
    HEURISTIC_SENTENCE_IDENTIFIER"> ...
      <ProcessedTextUnits>
        <ProcessedTextUnit TextUnitID="0">d Handel mit Ware alle Art sowie
        Import und Export .</ProcessedTextUnit> ... <ProcessedTextUnit
        TextUnitID="2">Stammkapital : <NeRef NeID="0" /> .</ProcessedTextUnit>
        <ProcessedTextUnit TextUnitID="3">Gesellschaft_mit_beschränkter_Haftung
        .</ProcessedTextUnit> ... <ProcessedTextUnit TextUnitID="9"><NeRef
        NeID="16" />, sein zur Geschäftsführerin bestellen .</ProcessedTextUnit> ...
        <ProcessedTextUnit TextUnitID="11">nicht eintragen : d Bekanntmachung d
        Gesellschaft erfolgen im Bundesanzeiger .</ProcessedTextUnit>
      </ProcessedTextUnits> ...
    </TextUnitsLayer>
  </DefaultDIAsDEMdocument>
</DefaultDIAsDEMvolume>
```

Note, the list of known lemma forms had been created using TreeTagger. In the file shown above, the inflected verb form “ist” now occurring in `<RollbackTextUnit>` has been mapped onto its infinitive form “sein” in the section `<ProcessedTextUnit>`. If TreeTagger is unable to determine the grammatical root form for a term, its lemma form simply equals the original term.

During the iterative clustering phase, text unit vectors are clustered based on content similarity to discover semantic XML tags. Text unit vectors are created by mapping elements of the section `<ProcessedTextUnits>` onto vectors. Thereby, vector dimensions correspond to so-called text units descriptors that are defined in a domain-specific and case-sensitive thesaurus. Consequently, thesauri should contain case-sensitive lemma forms of descriptor and non-descriptor thesaurus terms because they truly occur in the section `<ProcessedTextUnits>` only.

Lemmatize Text Units: Summary

Task: *Actions → Prepare Data Set → Lemmatize Text Units*

Use Case: The user must pre-process all imported texts as part of the DIAsDEM KDT process for semantic tagging of domain-specific texts archives. Creating lemma forms is the mandatory pre-processing step 4 of 4.

Prerequisites: The default text units layer of each DIAsDEM document must contain the section `<ProcessedTextUnits>`. Text units should have been created and tokenized in the DIAsDEM collection. Named entities should have been replaced with placeholders in all text units. If *Use TreeTagger to Determine Lemma Form* is enabled, the absolute file name of the respective TreeTagger start script (e.g., `/.../tree-tagger-german`) must be entered in the *External Programs* tab of the *Tools → Options* dialog.

Result: Elements of section `<ProcessedTextUnits>` mark up text units containing lemma forms and named entity placeholders. Additionally, the project properties *Default Collection File*, *Default TreeTagger Input File*, *Default TreeTagger Output File*, *Default Known Lemma Forms File*, and *Default Unknown Lemma Forms File* are set and updated, respectively.

Lemmatize Text Units: Parameters

Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: project property *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 absolute file name of the respective TreeTagger start script (e.g., `/.../tree-tagger-german`) must be entered in the *External Programs* tab of the *Tools* → *Options* dialog. If *Look Up Lemma Form in List* is enabled, a list of a priory known grammatical root forms (i.e., *Known Lemma Forms*) is utilized.

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 is replaced; this temporary file is created by DIAsDEM Workbench and includes text to be POS-tagged by TreeTagger; file extension: `.txt`; default value: project property *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 is replaced; this temporary file is created by TreeTagger and includes the results of POS-tagging; file extension: `.txt`; default value: project property *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 Subsection 4.3.5 on page 102; file extension: `.txt`; default value: project property *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 existing file that is created or extended by DIAsDEM Workbench; includes terms occurring 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 Subsection 4.3.5 on page 102; file extension: `.txt`; default value: project property *Default Unknown Lemma Forms File*

Advanced Options: 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.* If *Append Part of Speech Tag to Each Token* is enabled, each token is appended by its POS tag. When semantically marking up text ducuments, it is recommended to disable this option.

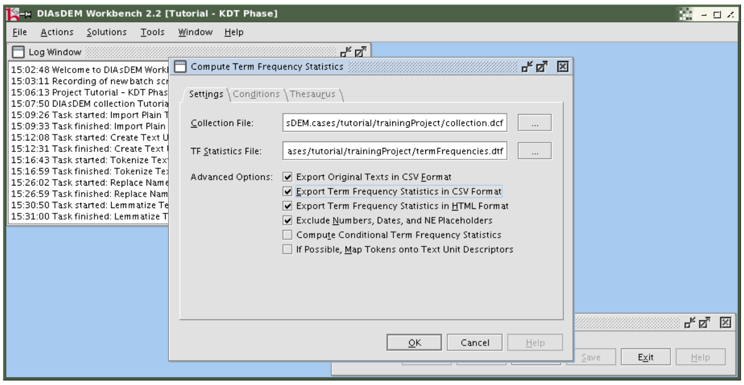
3.3 Iterative Clustering in the KDT Phase

3.3.1 Computing Term Frequency 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. Computing term frequency (TF) statistics for a collection is the first step in establishing or updating a thesaurus for subsequent use in clustering. Term frequency statistics give an insight into the specific word frequency distribution prevalent in a certain document collection. Based on term 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 term frequency statistics is described in this tutorial for the sake of completeness. Therefore, select *Actions* → *Understand Domain* → *Compute Term Frequency Statistics* and provide the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
TF Statistics File	`${PROJECT_HOME}/termFrequencies.dtf`

Advanced Options	`Enabled: Export Original Texts in CSV Format`
	`Enabled: Export Term Frequency Statistics in CSV Format`
	`Enabled: Export Term Frequency Statistics in HTML Format`
	`Enabled: Exclude Numbers, Dates and NE Placeholders`
	`Disabled: Compute Conditional Term Frequency Statistics`
	`Disabled: If Possible, Map Tokens onto Text Unit Descriptors`



```
15:02:48 Welcome to DIAsDEM Workl
15:03:11 Recording of new batch scr
15:06:13 Project Tutorial - KDT Phas
15:07:50 DIAsDEM collection Tutoria
15:09:26 Task started: Import Plain T
15:09:33 Task finished: Import Plain
15:12:08 Task started: Create Text U
15:12:31 Task finished: Create Text
15:16:43 Task started: Tokenize Tex
15:16:59 Task finished: Tokenize Te
15:26:02 Task started: Replace Name
15:26:59 Task finished: Replace Nam
15:30:50 Task started: Lemmatize Te
15:31:00 Task finished: Lemmatize T
```

Collection File: sDEM.cases/tutorial/trainingProject/collection.dcf ...

TF Statistics File: ases/tutorial/trainingProject/termFrequencies.dtf ...

Advanced Options:
☑ Export Original Texts in CSV Format
☑ Export Term Frequency Statistics in CSV Format
☑ Export Term Frequency Statistics in HTML Format
☑ Exclude Numbers, Dates, and NE Placeholders
☐ Compute Conditional Term Frequency Statistics
☐ If Possible, Map Tokens onto Text Unit Descriptors

Figure 3.11: *Compute Term Frequency Statistics* Dialog

Click on *OK* to compute term frequency statistics. Thereafter, check the content of the directory `${PROJECT_HOME}`, which now contains the DIAsDEM-specific term frequency statistics file termFrequencies.dtf. It can be opened using *Tools → Term Frequency Statistics Viewer*. Moreover, this directory contains term frequency statistics in CSV (termFrequencies.csv) and HTML (termFrequencies.html) format, respectively. Due to the settings of advanced options, all original texts of this collection have also been exported into the CSV files termFrequencies.orig.csv and termFrequencies.proc.csv. The former contains file names of input documents and the original texts as stored in section <OriginalText>. In contrast, the latter file contains processed texts (i.e., the content of the section <ProcessedTextUnits>) along with their input file names. Both export files can be input to further text analysis and text mining activities employing third-party software. According to the parameter settings, numbers, date literals, named entity placeholders, and other tokens that do not contain at least one letter are excluded from the term frequency statistics. The disabled options to compute conditional term frequency statistics and to utilize a thesaurs are beyond the scope of this case study.

Compute Term Frequency Statistics: Summary

Task: *Actions → Understand Domain → Compute Term Frequency Statistics*

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

Prerequisites: The default text units layer of each DIAsDEM document must contain the section <ProcessedTextUnits>. Text units should have been created and tokenized in the DIAsDEM collection and named entities should have been replaced with placeholders in all text units.

Result: *TF Statistics File* contains the absolute frequencies of single- and multi-token terms that occur in the section <ProcessedTextUnits> of DIAsDEM documents in collection *Collection File*. Additionally, the project properties *Default Collection File* and *Default Word Statistics File* are set and updated, respectively.

Remarks: The section <ProcessedTextUnits> should contain lemmatized text units in case of computing TF statistics for thesaurus establishment or update because only lemmatized text units should be mapped onto vectors for subsequent clustering. Thesauri to be employed by DIAsDEM Workbench should include lemma forms of both descriptors and non-descriptors only.

Compute Term Frequency Statistics: Parameters

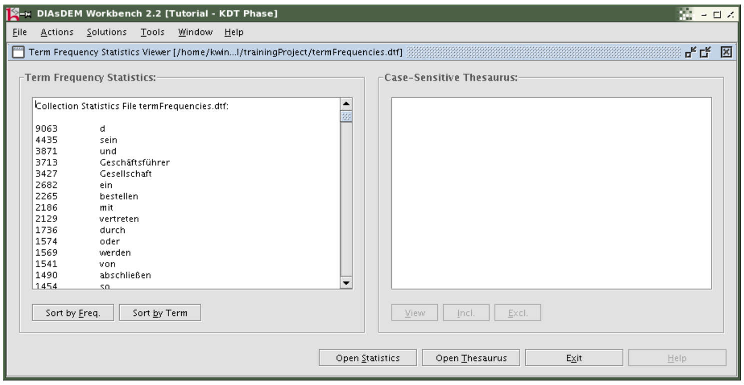
Collection File: Valid local file name of existing collection file; file extension: `.dcf`; default value: project property *Default Collection File*

TF Statistics File: Valid local file name of new or existing file that is created or replaced by DIAsDEM Workbench; file extension: `.dtf`; default value: project property *Default Word Statistics File*

Advanced Options: If *Export Original Texts in CSV Format* is enabled, two CSV files are created in `${PROJECT_HOME}` that contain the file name and the textual content of each document. If *Export Term Frequency Statistics in CSV Format* is enabled, a CSV file is created in `${PROJECT_HOME}` that contains all terms and the respective absolute frequencies. If *Export Term Frequency Statistics in HTML Format* is enabled, an HTML file is created in `${PROJECT_HOME}` that lists terms and their absolute frequencies. Tokens that do not contain at least one letter are excluded from TF statistics if *Exclude Numbers, Dates and NE Placeholders* is enabled.

3.3.2 Viewing Term Frequency Statistics

Select *Tools* → *Term Frequency Statistics Viewer* to open the new term frequency statistics file. Click on *Open Statistics* and choose the file `${PROJECT_HOME}/termFrequencies.dtf`. After entering the minimum frequency of terms to be displayed (e.g., 5), terms are shown in the left pane, as illustrated in Figure 3.12. Term frequency statistics can either be sorted by decreasing frequency or by ascending term. To sort the list of terms in the left pane, click the buttons *Sort by Freq.* and *Sort by Term*, respectively.

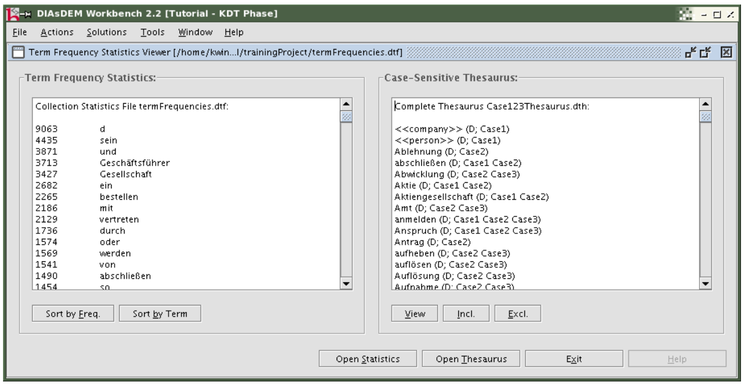

DIAsDEM Workbench 2.2 [Tutorial - KDT Phase]
File
Actions
Solutions
Tools
Window
Help
Term Frequency Statistics Viewer [/home/kwin...l/trainingProject/termFrequencies.dtf]
Term Frequency Statistics:
Case-Sensitive Thesaurus:
Sort by Freq.
Sort by Term
View
Incl.
Excl.
Open Statistics
Open Thesaurus
Exit
Help

```
Collection Statistics File termFrequencies.dtf:

9063        d
4435        sein
3871        und
3713        Geschäftsführer
3427        Gesellschaft
2682        ein
2265        bestellen
2186        mit
2129        vertreten
1736        durch
1574        oder
1569        werden
1541        von
1490        abschließen
1454        so
```

Figure 3.12: *Term Frequency Statistics Viewer* of DIAsDEM Workbench 2.2

Terms appearing in the left pane can be compared with an existing DIAsDEM-specific thesaurus file. To proceed, click the *Open Thesaurus* button and select the thesaurus file `${PARAMETER_HOME}/thesauri/de/Case123Thesaurus.dth`. As illustrated in Figure 3.13, the entire thesaurus is initially displayed in the right pane. Each line corresponds to one thesaurus term that can either be a descriptor or a non-descriptor that references its associated descriptor term.


DIAsDEM Workbench 2.2 [Tutorial - KDT Phase]
File
Actions
Solutions
Tools
Window
Help
Term Frequency Statistics Viewer [/home/kwin...l/trainingProject/termFrequencies.dtf]
Term Frequency Statistics:
Case-Sensitive Thesaurus:
Sort by Freq.
Sort by Term
View
Incl.
Excl.
Open Statistics
Open Thesaurus
Exit
Help

```
Collection Statistics File termFrequencies.dtf:

9063          d
4435          sein
3871          und
3713          Geschäftsführer
3427          Gesellschaft
2682          ein
2265          bestellen
2186          mit
2129          vertreten
1736          durch
1574          oder
1569          werden
1541          von
1490          abschließen
1454          so
```

```
Complete Thesaurus Case123Thesaurus.dth:

<<company>> (D; Case1)
<<person>> (D; Case1)
Ablehnung (D; Case2)
abschließen (D; Case1 Case2)
Abwicklung (D; Case2 Case3)
Aktie (D; Case1 Case2)
Aktiengesellschaft (D; Case1 Case2)
Amt (D; Case2 Case3)
anmelden (D; Case1 Case2 Case3)
Anspruch (D; Case1 Case2 Case3)
Antrag (D; Case2)
aufheben (D; Case2 Case3)
auflösen (D; Case2 Case3)
Auflösung (D; Case2 Case3)
Aufnahme (D; Case2 Case3)
```

Figure 3.13: *Term Frequency Statistics Viewer* of DIAsDEM Workbench 2.2

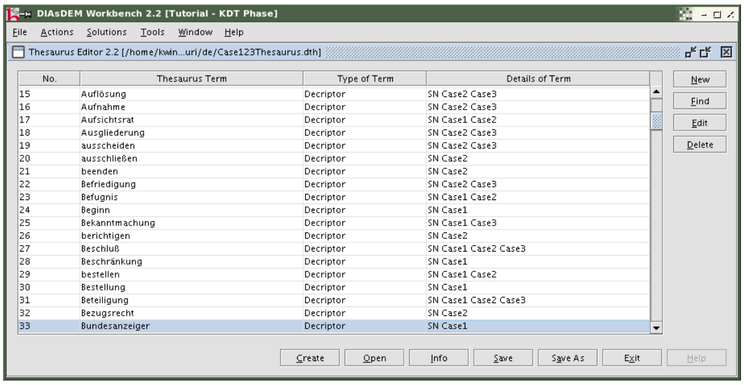
Thesaurus entry `Ablehnung (D; Case2)` corresponds to the descriptor ("D") term "Ablehnung", which is a valid text unit descriptor in case study 2 only. Note again, valid text unit descriptors correspond to dimensions of text unit vectors to be clustered subsequently. According to thesaurus entry `<<person>> (D; Case1)`, 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 respective descriptor counter is incremented. Finally, thesaurus entry `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" is thus incremented. However, the descriptor `Beginn (D; Case1)` is a valid descriptor in case study 1 only.

Click the *Incl.* button in the right pane to filter term frequency statistics of terms that are also descriptor or non-descriptor thesaurus terms. In this case, the collection-specific term frequency is also shown for each thesaurus term in the right pane. Click the *Excl.* button to filter term frequency statistics of terms that are not contained in the thesaurus. Frequently occurring and semantically important terms, which are not listed in the thesaurus, are candidates for thesaurus updates. In contrast, infrequently occurring terms might be removed from the thesaurus to reduce the dimensionality of text units vectors. However, concepts such as "Tätigkeit" should not be deleted if they are more general descriptor terms for many, less frequently occurring non-descriptors. Note, specific named entity placeholders (e.g., `<<0>>`) should not be included in any thesaurus because they might replace instances of various named entity types, such as “person” or “date”.

After analyzing term frequency statistics, you might consider to add the frequently occurring 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 due to its infrequent occurrence in the collection. Thesaurus updates are explained in the next subsection. Therefore, do not close *Term Frequency Statistics Viewer* yet.

3.3.3 Editing Domain-Specific Thesauri

DIAsDEM Workbench includes two German thesauri in `${PARAMETER_HOME}/thesauri/de`. They contain application-specific vocabularies for case studies related to Commercial Register entries and corporate news, respectively. Thesauri, which should be employed in different application domains, can be created by *Actions → Understand Domain → Establish Initial Thesaurus*, as described in Subsection 3.6.2. However, the remainder of this section focuses on the process of updating an existing, DIAsDEM-specific thesaurus by adding, editing, and removing terms. Select *Tools → Thesaurus Editor 2.2*, click on *Open*, and select the thesaurus file `Case123Thesaurus.dth` in the directory `${PARAMETER_HOME}/thesauri/de`.

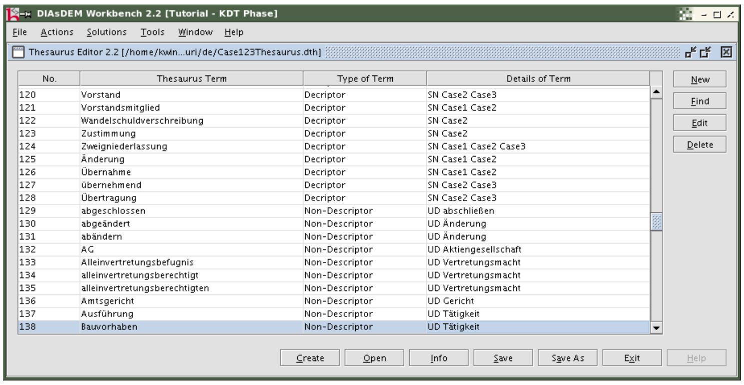

DIAsDEM Workbench 2.2 [Tutorial - KDT Phase]
File
Actions
Solutions
Tools
Window
Help
Thesaurus Editor 2.2 [/home/kwin...uri/de/Case123Thesaurus.dth]
New
Find
Edit
Delete
Create
Open
Info
Save
Save As
Exit
Help

No.	Thesaurus Term	Type of Term	Details of Term
15	Auflösung	Decriptor	SN Case2 Case3
16	Aufnahme	Decriptor	SN Case2 Case3
17	Aufsichtsrat	Decriptor	SN Case1 Case2
18	Ausgliederung	Decriptor	SN Case2 Case3
19	ausscheiden	Decriptor	SN Case2 Case3
20	ausschließen	Decriptor	SN Case2
21	beenden	Decriptor	SN Case2
22	Befriedigung	Decriptor	SN Case2 Case3
23	Befugnis	Decriptor	SN Case1 Case2
24	Beginn	Decriptor	SN Case1
25	Bekanntmachung	Decriptor	SN Case1 Case3
26	berichtigen	Decriptor	SN Case2
27	Beschluß	Decriptor	SN Case1 Case2 Case3
28	Beschränkung	Decriptor	SN Case1
29	bestellen	Decriptor	SN Case1 Case2
30	Bestellung	Decriptor	SN Case1
31	Beteiligung	Decriptor	SN Case1 Case2 Case3
32	Bezugsrecht	Decriptor	SN Case2
33	Bundesanzeiger	Decriptor	SN Case1

Figure 3.14: *Thesaurus Editor 2.2* of DIAsDEM Workbench 2.2

To add the first new term, click on *New* and enter “Bundesanzeiger” in the appearing dialog. You might use the system clipboard to transfer textual content, for example, from *Term Frequency Statistics Viewer* to *Thesaurus Editor 2.2*. Using the mouse, select the term “Bundesanzeiger” in *Term Frequency Statistics Viewer*. The highlighted text can be copied into the clipboard by the keyboard shortcut *CTRL-C*. Afterwards, the clipboard content 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., to 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 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, set the attribute value *Type of Term* to *Descriptor*. The content of the field *Scope Notes* can be used to limit the number of valid descriptors during the process of vectorizing text units. 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 3.14 depicts *Thesaurus Editor 2.2* after committing the insertion of the new term “Bundesanzeiger”.


DIAsDEM Workbench 2.2 [Tutorial - KDT Phase]
File
Actions
Solutions
Tools
Window
Help
Thesaurus Editor 2.2 [/home/kwin...uri/de/Case123Thesaurus.dth]
New
Find
Edit
Delete
Create
Open
Info
Save
Save As
Exit
Help

No.	Thesaurus Term	Type of Term	Details of Term
120	Vorstand	Decriptor	SN Case2 Case3
121	Vorstandsmitglied	Decriptor	SN Case1 Case2
122	Wandelschuldverschreibung	Decriptor	SN Case2
123	Zustimmung	Decriptor	SN Case2
124	Zweigniederlassung	Decriptor	SN Case1 Case2 Case3
125	Änderung	Decriptor	SN Case1 Case2
126	Übernahme	Decriptor	SN Case1 Case2
127	übernehmend	Decriptor	SN Case2 Case3
128	Übertragung	Decriptor	SN Case2 Case3
129	abgeschlossen	Non-Descriptor	UD abschließen
130	abgeändert	Non-Descriptor	UD Änderung
131	abändern	Non-Descriptor	UD Änderung
132	AG	Non-Descriptor	UD Aktiengesellschaft
133	Alleinvertretungsbefugnis	Non-Descriptor	UD Vertretungsmacht
134	alleinvertretungsberechtigt	Non-Descriptor	UD Vertretungsmacht
135	alleinvertretungsberechtigten	Non-Descriptor	UD Vertretungsmacht
136	Amtsgericht	Non-Descriptor	UD Gericht
137	Ausführung	Non-Descriptor	UD Tätigkeit
138	Bauvorhaben	Non-Descriptor	UD Tätigkeit

Figure 3.15: *Thesaurus Editor* of DIAsDEM Workbench 2.2

Insert the second new term “Bauvorhaben” into the thesaurus and set its *Type of Term* to *Non-Descriptor*. For each non-descriptor, an associated descriptor must be specified in the field *Use Descriptor*. Hence, type in the descriptor term “Tätigkeit” in this field. Note, the *Use Descriptor* field must contain an existing text unit descriptor or another non-descriptor that in turn provides a reference to the relevant text unit descriptor in the same thesaurus. As before, input “Case1” in the field *Scope Notes* as well. Figure 3.15 illustrates *Thesaurus Editor 2.2* after committing the insertion of the term “Bauvorhaben”.

Existing terms, such as “Bundesanzeiger” and “Bauvorhaben”, can be modified by selecting the table row of the term to be modified and subsequently clicking the *Edit* button. Alternatively, select the table row of the term to be modified, right-click the row, and select *Edit Selected Term* from the context menu. Analogously, terms can be removed from the thesaurus. 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.

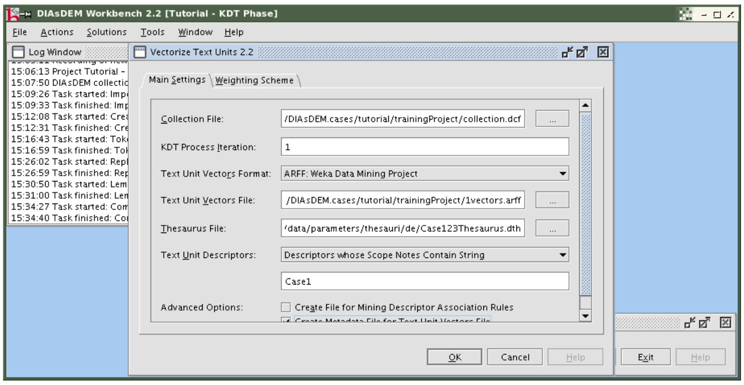
Click the *Info* button and look at the brief thesaurus summary that lists the number of terms, descriptors, and non-descriptors in the opened thesaurus. The case-specific thesaurus should now include 127 descriptor and 104 non-descriptor terms. Keep in mind that the number of descriptors should be kept as low as possible because DIAsDEM Workbench does not employ uncontrolled techniques for dimensionality reduction (e.g., singular value decomposition). As a rule of thumb, the number of descriptors should not exceed 250 terms. Click the *Save* button to commit the previous thesaurus updates. After saving, inspect the content of the directory `${PARAMETER_HOME}/thesauri/de`. 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 both *Thesaurus Editor 2.2* and *Term Frequency Statistics Viewer*.

3.3.4 Vectorizing Text Units in Iteration 1

Concerning the clustering of text unit vectors, DIAsDEM Workbench implements both a plug-in and a plug-out concept, which enables the usage of various clustering algorithms. Users can employ three built-in Weka [WF05] clustering algorithms (i.e., k-means, Cobweb and EM), five clustering algorithms implemented within the Weka framework by the author (i.e., simple and bisecting k-means, Kohonen's Batch Map algorithm, Jarvis/ Patrick SNN clustering, as well as Ertöz/Steinbach/Kumar SNN clustering), or utilize algorithms supplied by external data mining applications. To ensure this flexibility, DIAsDEM Workbench is capable of exporting text unit vector files in four different formats. As the k-means clustering algorithm provided by the Java-based data mining library Weka is employed is this case study, vectors are exported in the Weka-specific ARFF format only. However, all four formats are briefly described in Subsection 4.4.1 on page 104. To export text unit vectors for the first clustering iteration, select *Actions → Prepare Data Set → Vectorize Text Units 2.2* and enter the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
KDT Process Iteration	`1`
Text Unit Vectors Format	`ARFF: Weka Data Mining Project`
Text Unit Vectors File	`${PROJECT_HOME}/1vectors.arff`
Thesaurus File	`${PARAMETER_HOME}/thesauri/de/Case123Thesaurus.dth`
Text Unit Descriptors	`Descriptors whose Scope Notes Contain String Case1`

Descriptor Frequency	`Boolean Descriptor Frequency`
Collection Frequency	`Inverse Collection Frequency: Create New File`
Collection Frequencies File	`${PROJECT_HOME}/1weights.dcfq`
Length Normalization	`No Length Normalization`
Advanced Options	`Disabled: Create File for Mining Descriptor Association Rules`
	`Enabled: Create Metadata File for Text Unit Vectors File`



```
15:06:13 Project Tutorial -
15:07:50 DIAsDEM collectic
15:09:26 Task started: Imp
15:09:33 Task finished: Imp
15:12:08 Task started: Cre
15:12:31 Task finished: Cre
15:16:43 Task started: Tok
15:16:59 Task finished: Tok
15:26:02 Task started: Repl
15:26:59 Task finished: Rep
15:30:50 Task started: Lem
15:31:00 Task finished: Len
15:34:27 Task started: Com
15:34:40 Task finished: Co
```

Collection File: /DIAsDEM.cases/tutorial/trainingProject/collection.dcf

KDT Process Iteration: 1

Text Unit Vectors Format: ARFF: Weka Data Mining Project

Text Unit Vectors File: /DIAsDEM.cases/tutorial/trainingProject/1vectors.arff

Thesaurus File: 'data/parameters/thesauri/de/Case123Thesaurus.dth

Text Unit Descriptors: Descriptors whose Scope Notes Contain String

Case1

Advanced Options: □ Create File for Mining Descriptor Association Rules

☑ Create Metadata File for Text Unit Vectors File

Figure 3.16: *Vectorize Text Units 2.2* Dialog

Click the *OK* button to export text unit vectors according to these parameter settings. In the first clustering iteration, processed text units in section `<ProcessedTextUnits>` are mapped onto their vector representations. Let D be the set of descriptors. The dimensionality of text unit vectors corresponds to $|D| = 73$ descriptors in thesaurus file `Case123Thesaurus.dth` that contain the string "Case1" in their respective scope notes.

Mapping 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 the 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 occurring in almost all text units. Length normalization is not performed as text units do not vary in length. To sum up, 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 metadata file `${PROJECT_HOME}/1vectors.arff.meta` that lists the descriptor frequency and the inverse document frequency for each descriptor:

```
...
D1_Aktie = Aktie; Descriptor Frequency = 37; Descriptor Weight = 5.5
D2_Gesellschafter = Gesellschafter; Descriptor Frequency = 172; Descriptor Weight = 3.9
...
D73_Anspruch = Anspruch; Descriptor Frequency = 9; Descriptor Weight = 6.9
```

Note that descriptor term “Aktie” and the associated non-descriptors, such as “Namensaktie”, occur 37 times in the collection of Commercial Register entries. The term weight of “Aktie”, which 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 IDF weighting schema, “Aktie” has a greater discriminative power than “Gesellschafter” due to its relatively infrequent occurrence in the collection. Note, this metadata file has a purely informative character. In contrast, the file `${PROJECT_HOME}/1weights.ddw` contains the same descriptor weights for usage in the first clustering iteration of the application phase.

The text unit vector file `${PROJECT_HOME}/vectors1.arff` is input to the first clustering iteration, which is described in the next section. This file contains 9,254 vectors to be clustered in the Weka-specific ARFF format [WF05]. ARFF-files include metadata about the relation and its attributes (i.e., their names and domains), as well as the actual data below the line `@data`. For example, the second vector depicted below corresponds to the second text unit of file `${PROJECT_HOME}/inputCollection/volume100668.xml`: “Persönlich haftende Gesellschafterin: AGE Glas Vertrieb GmbH, Sitz: Garbsen.” The descriptor “Gesellschafter” occurs in this text unit in the form of its associated non-descriptor “Gesellschafterin”. Hence the second vector component represents a term weight greater than zero. The first component of the second vector equals zero because neither the descriptor term “Aktie” nor a related non-descriptor occurs in this sentence.

```
@relation 'DIAsDEM'
@attribute DocumentType string
@attribute Document string
@attribute TextUnit string
@attribute D1_Aktie real
@attribute D2_Gesellschafter real
...
@attribute D73_Anspruch real
@data
...
null,/home/.../volume100668.xml:0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...
null,/home/.../volume100668.xml:0,1,0,3.98531,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...
...
```

Vectorize Text Units 2.2: Summary

Task: *Actions → Prepare Data Set → Vectorize Text Units 2.2*

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

Prerequisites: The default text units layer of each DIAsDEM document must contain the section `<ProcessedTextUnits>`. Text units should have been created, tokenized, and lemmatized in the DIAsDEM collection, and named entities should have been replaced with placeholders in all text units.

Remarks: In the KDT phase of the DIAsDEM framework, an iteration-specific collection frequencies file is created for usage in the application phase.

Vectorize Text Units 2.2: Parameters

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

KDT Process Iteration: If `1` is input to indicate the first iteration, text unit vectors are created for all text units in section `<ProcessedTextUnits>`. If a number greater than `1` is input to indicate subsequent iterations, text unit vectors are created for text units in section `<ProcessedTextUnits>` that have not been semantically named in a previous clustering iteration (i.e., `ClusterLabel="-"`). These vectors have been assigned to qualitatively unacceptable clusters in all preceding iterations, as explained in Section 1. Default value: project property *Default Iteration (1, 2, ...)*

Text Unit Vectors Format: Choice of vector file format as described in Subsection 4.4.1 on page 104 between comma separated values (CSV file), fixed width values (TXT file), the Weka-specific ARFF file format, and the sparse ARFF file format; default value: project property *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 *Text Unit Vectors Format*: `.csv`, `.txt`, or `.arff`; default value: project property *Default Text Unit Vectors File*

Thesaurus File: Valid local file name of existing DIAsDEM-specific thesaurus file as described in Subsection 4.4.1 on page 104; file extension: `.dth`; default value: project property *Default Thesaurus File*

Text Unit Descriptors: If *All Descriptors in Thesaurus* is enabled, all descriptor terms in *Thesaurus File* are vector dimensions. If *Descriptors whose Scope Note Contain String* is enabled, only descriptor terms in *Thesaurus File* are valid whose scope notes contain the complete 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 complete string entered below.

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

Collection Frequency: In the KDT phase, either the option *No Collection Frequency: Create New File* or *Inverse Collection Frequency: Create New File* has to be enabled. In the former case, the term frequency of valid descriptor d in text unit u is not weighted by a collection frequency component, and thus all descriptor weights equal the descriptor frequency. If the latter option is enabled, the descriptor frequency of valid descriptor d in text unit u is multiplied by the inverse document frequency of d in the entire collection to compute the descriptor weight. In contrast, the option *Apply Existing Collection Frequencies File* must be enabled in the application phase. In this phase, an existing file comprising iteration-specific collection frequencies file must be specified as well.

Collection Frequencies File: Valid name of local file comprising collection frequencies that is created or replaced in the KDT phase and retrieved in the application phase, respectively; file extension: `.dcfq`; default value: project property *Default Descriptor Weights File*

Length Normalization: If *No Length Normalization* is selected, the final descriptor weights are not normalized to take care of variation in text unit length. Otherwise, a Cosine length normalization is performed during text unit vectorization.

Advanced Options: If *Create File for Mining Descriptor Association Rules* is enabled, an additional file named analogously to *Text Unit Vectors File* is created, which is suffixed `.assoc`. It can be used for discovering association rules between descriptor terms in text units. If *Create Metadata File for Text Unit Vectors File* is enabled, an additional file is created that is named analogously to *Text Unit Vectors File* with suffix `.meta`. This metadata file contains mappings of abbreviated attribute names onto their respective unabbreviated descriptors along with their descriptor frequencies and descriptor 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 content for subsequent semantic labeling. This plug-out concept has been successfully tested in case studies employing commercial data mining applications [WS01c, WS02c]. Additionally, DIAsDEM Workbench implements a plug-in concept that, for example, wraps the Java-based data mining library Weka.

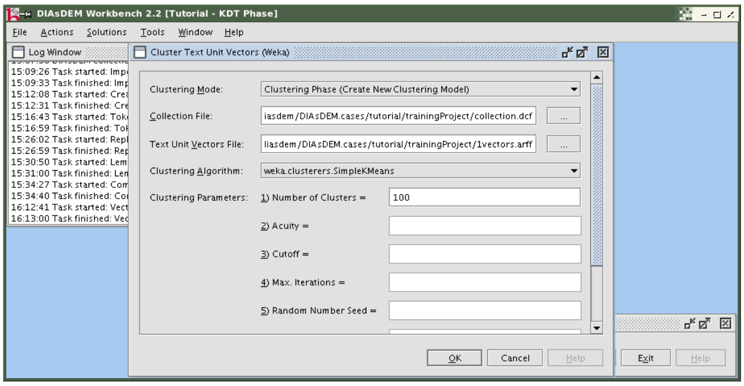
Along with various data pre-processing and machine learning algorithms, Weka includes three clustering algorithms (i.e., k-means, Cobweb, and EM), which have been integrated into DIAsDEM Workbench. Discussing these clustering algorithms is beyond the scope of this tutorial. See [WF05] for an excellent description of these algorithms, their parameters, and their implementation in the open source Weka library. For moderate amounts of data, all three Weka algorithms are capable of clustering text unit vectors without memory- or runtime-related problems. For large amounts of data, the task *Actions → Discover Patterns → Cluster Text Unit Vectors (hypKNOWsys)* outperforms the Weka algoritms. In this case study, however, the Weka k-means clustering algorithm is employed only. To proceed, select *Actions → Discover Patterns → Cluster Text Unit Vectors (Weka)* and enter the following parameters:

Parameter	Value
Clustering Mode	`Clustering Phase (Create New Clustering Model)`
Collection File	`${PROJECT_HOME}/collection.dcf`
Text Unit Vectors File	`${PROJECT_HOME}/1vectors.arff`
Clustering Algorithm	`weka.clusterers.SimpleKMeans`

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	`${PARAMETER_HOME}/1results.csv`
Text Unit Clusterer File	`${PARAMETER_HOME}/1clusterer.wskm`

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 remain empty. *Number of Clusters* is the only parameter of this algorithm whereas *Acuity* and *Cutoff* are two parameters of the Cobweb clustering algorithm. The EM algorithm can be parameterized by *Max. Iterations*, *Random Number Seed*, and *Min. Std. Deviation.* All algorithms require text unit vector files that conform to the Weka-specific ARFF-format. Again, see [WF05] for a detailed discussion of these parameters. The progress of clustering, which requires a few minutes, cannot be displayed due to the missing support of progress measurement in Weka.


DIAsDEM Workbench 2.2 [Tutorial - KDT Phase]
File
Actions
Solutions
Tools
Window
Help
Log Window
Cluster Text Unit Vectors (Weka)
...
OK
Cancel
Help
Exit
Help

```
15:09:26 Task started: Imp
15:09:33 Task finished: Imp
15:12:08 Task started: Cre
15:12:31 Task finished: Cre
15:16:43 Task started: Tok
15:16:59 Task finished: Tok
15:26:02 Task started: Repl
15:26:59 Task finished: Rep
15:30:50 Task started: Lem
15:31:00 Task finished: Len
15:34:27 Task started: Com
15:34:40 Task finished: Co
16:12:41 Task started: Vect
16:13:00 Task finished: Vec
```

Clustering Mode: Clustering Phase (Create New Clustering Model)

Collection File: iasdem/DIAsDEM.cases/tutorial/trainingProject/collection.dcf

Text Unit Vectors File: liasdem/DIAsDEM.cases/tutorial/trainingProject/1vectors.arff

Clustering Algorithm: weka.clusterers.SimpleKMeans

Clustering Parameters: 1) Number of Clusters = 100

2) Acuity =

3) Cutoff =

4) Max. Iterations =

5) Random Number Seed =

Figure 3.17: *Cluster Text Unit Vectors (Weka)* Dialog

DIAsDEM Workbench post-processes the proprietary output file generated by Weka clusterers (e.g., `${PROJECT_HOME}/1results.csv.temp`). Clustering results are converted into CSV files, which can easily be processed by the tasks *Actions* → *Postprocess Patterns* → *Monitor Cluster Quality 2.2* and *Actions* → *Postprocess Patterns* → *Tag Text Units*, respectively. After clustering has finished, inspect the *Clustering Results File* `${PROJECT_HOME}/1results.csv`. Each line contains a DIAsDEM document ID, the respective text unit ID as the second attribute, and the associated cluster ID as the third attribute. Consider, for example, the first and only DIAsDEM document in file `${PROJECT_HOME}/inputCollection/volume100668.xml`: The first text unit is assigned to cluster 2 whereas the second one is assigned to cluster 48. Note, cluster 2 also contains the first text unit of the DIAsDEM document in file `volume100669.xml`.

```
...
/home/.../volume100668.xml:0,0,2
/home/.../volume100668.xml:0,1,48
/home/.../volume100668.xml:0,2,31
/home/.../volume100668.xml:0,3,26
/home/.../volume100668.xml:0,4,52
/home/.../volume100669.xml:0,0,2
...
```

As described in Subsection 3.3.6, the content of text units clusters can be visualized by *Actions* → *Postprocess Patterns* → *Monitor Cluster Quality 2.2*. The file `${PROJECT_HOME}/1clusterer.wskm` is a serialized instance of the Java class `weka.clusterers.SimpleKMeans`. This so-called text unit clusterer can be employed to cluster text unit vectors during the application phase of the DIAsDEM framework. In contrast to the clustering or KDT phase exemplified by this section, *Text Unit Clusterer File* is an input file during the application phase. However, running DIAsDEM Workbench in application mode can be simulated by applying `${PROJECT_HOME}/1clusterer.wskm` to the same text unit vectors in `${PROJECT_HOME}/1vectors.arff` using the following parameters:

Parameter	Value
Clustering Mode	`Application Phase (Apply Existing Clustering Model)`
Collection File	`${PROJECT_HOME}/collection.dcf`
Text Unit Vectors File	`${PROJECT_HOME}/1vectors.arff`
Clustering Algorithm	`weka.clusterers.SimpleKMeans`

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

Clustering Results File	`${PARAMETER_HOME}/1results.csv`
Text Unit Clusterer File	`${PARAMETER_HOME}/1clusterer.wskm`

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

The task *Actions → Discover Patterns → Cluster Text Unit Vectors (Weka)* encapsulates three Weka clustering algorithms (i.e., k-means, Cobweb, and EM). Due to their poor performance on large data sets in discovery mode, the missing support of common cluster validity indices, and the necessity to process data sets inside the main memory in application mode, however, the author decided to implement five additional algorithms: an enhanced simple k-means algorithm, the bisecting k-means algorithm [SKK00], the Batch Map algorithm [Koh01, pp. 139–140], the Jarvis/Patrick SNN clustering [JP73], as well as the Ertöz/Steinbach/Kumar SNN clustering [ESK04]. These algorithms are available via the task *Actions → Discover Patterns → Cluster Text Unit Vectors (hypKNOWsys)*.

Cluster Text Unit Vectors (Weka): Summary

Task:	*Actions → Discover Patterns → Cluster Text Unit Vectors (Weka)*
Use Case:	The user wants to cluster text unit vectors as part of the DIAsDEM KDT 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. This file format is exported by *Actions → Prepare Data Set → Vectorize Text Units 2.2* and is described in Subsection 4.4.1 on page 104.

Result: In clustering mode, text unit vectors are clustered by the chosen algorithm. The resulting text unit 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 project properties *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 might also be used if the results can be exported or converted into a file format supported by DIAsDEM Workbench.

Cluster Text Unit Vectors (Weka): Parameters

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 content of *Text Unit Vectors File.*

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

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

Clustering Algorithm: One of three algorithms supported by the Java-based Weka library [WF05] must be selected: *weka.clusterers.SimpleKMeans*, *weka.clusterers.Cobweb*, or *weka.clusterers.EM.*

Clustering Parameters: If *Clustering Phase (Create New Clustering Model)* is enabled, the selected algorithm can be parameterized as follows [WF05]: *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: project property *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 depends on clustering algorithm: `.wskm`, `.wcw`, or `.wem`; default value: project property *Default Text Unit Clusterer File*

3.3.6 Monitoring Cluster Quality in Iteration 1

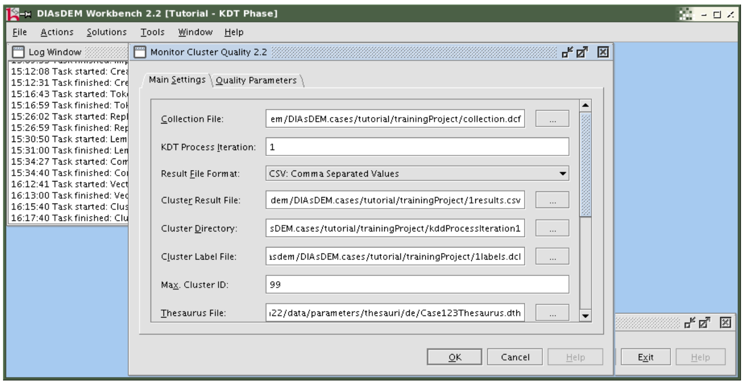
As explained in the introduction, the set of text unit clusters discovered during clustering has to be analyzed to separate qualitatively acceptable clusters from unacceptable ones. Recall that members of the former are semi-automatically assigned a semantic label whereas all text unit vectors assigned 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 cluster quality criteria are described in [GSW01]. The cluster quality monitor of DIAsDEM Workbench computes descriptive statistics for clusters, visualizes the content of clusters in HTML files, and creates a cluster label file. The latter contains default semantic labels for qualitatively acceptable clusters only. Default cluster labels are composed of text unit descriptors that prevail in the respective clusters. Select *Actions → Postprocess Patterns → Monitor Cluster Quality 2.2* and submit the following parameters to start monitoring cluster quality:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
KDT Process Iteration	`1`
Result File Format	`CSV: Comma Separated Values`
Cluster Result File	`${PROJECT_HOME}/1results.csv`
Cluster Directory	`${PROJECT_HOME}/kddProcessIteration1`
Cluster Label File	`${PROJECT_HOME}/1labels.dcl`
Max. Cluster ID	`99`
Thesaurus File	`${PARAMETER_HOME}/thesauri/de/Case123Thesaurus.dth`

Text Unit Descriptors	`Descriptors whose Scope Notes Contain String`
	`Case1`

Advanced Options	`Disabled: Ignore First Line of Cluster Result File`
	`Enabled: Ignore Empty Clusters in Cluster Index HTML File`
	`Enabled: Rank Clusters by Quality in Cluster Index HTML File`
	`Enabled: Launch Web Browser with Cluster Index HTML File`
	`Enabled: Launch Cluster Label Editor with Cluster Label File`
	`Enabled: Dump DIAsDEM Documents for Visualization`

Cluster Quality Criteria	`1) Dominant Descriptor Threshold = 0.8`
	`2) Rare Descriptor Threshold = 0.01`
	`3) Max. Descriptor Coverage = 0.75`
	`4) Min. Descriptor Dominance = 0.25`
	`5) Min. Cluster Size = 50`
	`6) Frequent Non-Descriptor Threshold = 0.2`
	`7) Max. Number of Output Text Units = 1000`



```
15:12:08 Task started: Crea
15:12:31 Task finished: Cre
15:16:43 Task started: Toke
15:16:59 Task finished: Tok
15:26:02 Task started: Repl
15:26:59 Task finished: Rep
15:30:50 Task started: Lem
15:31:00 Task finished: Len
15:34:27 Task started: Com
15:34:40 Task finished: Coi
16:12:41 Task started: Vect
16:13:00 Task finished: Vec
16:15:40 Task started: Clus
16:17:40 Task finished: Clu
```

Collection File: em/DIAsDEM.cases/tutorial/trainingProject/collection.dcf

KDT Process Iteration: 1

Result File Format: CSV: Comma Separated Values

Cluster Result File: dem/DIAsDEM.cases/tutorial/trainingProject/1results.csv

Cluster Directory: sDEM.cases/tutorial/trainingProject/kddProcessIteration1

Cluster Label File: ısdem/DIAsDEM.cases/tutorial/trainingProject/1labels.dcl

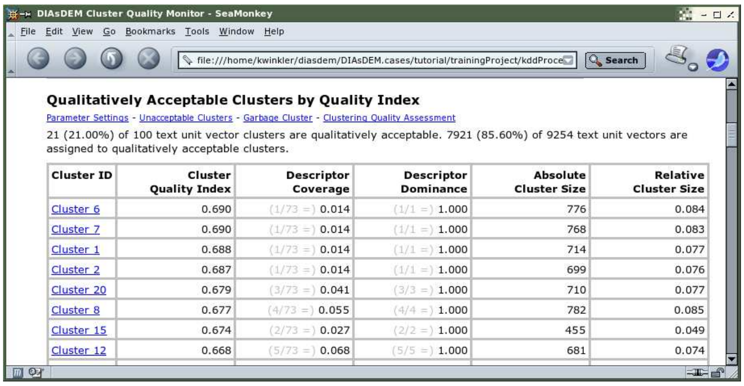
Max. Cluster ID: 99

Thesaurus File: ı22/data/parameters/thesauri/de/Case123Thesaurus.dth

Figure 3.18: *Monitor Cluster Quality 2.2* Dialog

The settings of *Thesaurus File* and *Text Unit Descriptors* must exactly correspond to the parameters entered in *Actions → Prepare Data Set → Vectorize Text Units 2.2* in the current clustering iteration. *Max. Cluster ID* must equal the greatest integer that serves 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. However, *Max. Cluster ID* is 99 according to the output of *Actions → Discover Patterns → Cluster Text Unit Vectors (Weka)*. Please refer to [GSW01] for description of *Cluster Quality Criteria*. Note that [GSW01] used a different terminology and refers to *Max. Descriptor Coverage* as *Max. Distinct Ratio*, to *Min. Descriptor Dominance* as *Min. Frequent Ratio*, and to *Min. Cluster Size* as *Min. Cardinality*. In addition, DIAsDEM Workbench 2.2 employs four additional threshold parameters that affect either cluster quality computation (i.e., *Dominant Descriptor Threshold* and *Rare Descriptor Threshold*) or cluster visualization output (i.e., *Frequent Non-Descriptor Threshold* and *Max. Number of Output Text Units*). Contact the author to obtain a publication that describes the cluster quality criteria in detail. In principle, however, decreasing *Min. Cluster Size*, decreasing *Min. Descriptor Dominance*, or increasing *Max. Descriptor Coverage* 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}/kddProcessIteration1/index.html`. As illustrated


DIAsDEM Cluster Quality Monitor - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
file:///home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject/kddProce
Search

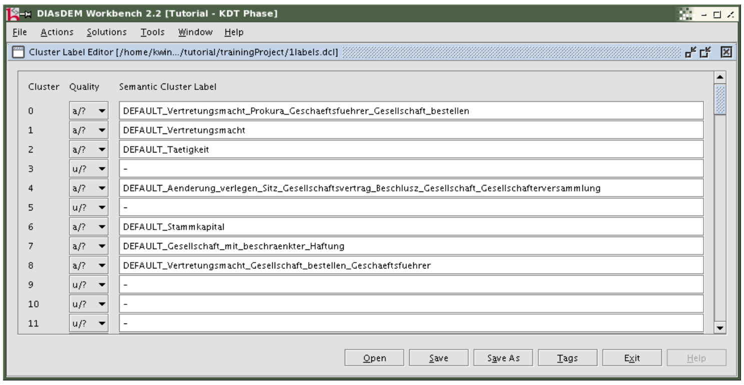
Qualitatively Acceptable Clusters by Quality Index

Parameter Settings - Unacceptable Clusters - Garbage Cluster - Clustering Quality Assessment

21 (21.00%) of 100 text unit vector clusters are qualitatively acceptable. 7921 (85.60%) of 9254 text unit vectors are assigned to qualitatively acceptable clusters.

Cluster ID	Cluster Quality Index	Descriptor Coverage	Descriptor Dominance	Absolute Cluster Size	Relative Cluster Size
Cluster 6	0.690	(1/73 =) 0.014	(1/1 =) 1.000	776	0.084
Cluster 7	0.690	(1/73 =) 0.014	(1/1 =) 1.000	768	0.083
Cluster 1	0.688	(1/73 =) 0.014	(1/1 =) 1.000	714	0.077
Cluster 2	0.687	(1/73 =) 0.014	(1/1 =) 1.000	699	0.076
Cluster 20	0.679	(3/73 =) 0.041	(3/3 =) 1.000	710	0.077
Cluster 8	0.677	(4/73 =) 0.055	(4/4 =) 1.000	782	0.085
Cluster 15	0.674	(2/73 =) 0.027	(2/2 =) 1.000	455	0.049
Cluster 12	0.668	(5/73 =) 0.068	(5/5 =) 1.000	681	0.074

Figure 3.19: Cluster Index File Created by *Monitor Cluster Quality 2.2*


DIAsDEM Workbench 2.2 [Tutorial - KDT Phase]
File Actions Solutions Tools Window Help
Cluster Label Editor [/home/kwin.../tutorial/trainingProject/1labels.dcl]
Open
Save
Save As
Tags
Exit
Help

Cluster	Quality	Semantic Cluster Label
0	a/?	DEFAULT_Vertretungsmacht_Prokura_Geschaeftsfuehrer_Gesellschaft_bestellen
1	a/?	DEFAULT_Vertretungsmacht
2	a/?	DEFAULT_Taetigkeit
3	u/?	-
4	a/?	DEFAULT_Aenderung_verlegen_Sitz_Gesellschaftsvertrag_Beschlusz_Gesellschaft_Gesellschafterversammlung
5	u/?	-
6	a/?	DEFAULT_Stammkapital
7	a/?	DEFAULT_Gesellschaft_mit_beschraenkter_Haftung
8	a/?	DEFAULT_Vertretungsmacht_Gesellschaft_bestellen_Geschaeftsfuehrer
9	u/?	-
10	u/?	-
11	u/?	-

Figure 3.20: *Cluster Label Editor* of DIAsDEM Workbench 2.2

in Figure 3.19, this file references all non-empty cluster files in the same directory. If the browser cannot be launched, check the current settings in the *External Programs* tab of the *Tools* → *Options* dialog. Figure 3.20 depicts *Cluster Label Editor*, which is also launched within DIAsDEM Workbench. This editor allows you to modify the automatically created cluster label file `${PROJECT_HOME}/1labels.dcl` by altering and deleting default cluster labels, as well as by semantically naming clusters without a default label. Using *Cluster Label Editor* to customize the file `${PROJECT_HOME}/1labels.dcl` is described in Subsection 3.3.7. Nevertheless, close both *Cluster Label Editor* and the browser displaying `${PROJECT_HOME}/kddProcessIteration1/index.html`.

Monitor Cluster Quality 2.2: Summary

Task: *Actions → Postprocess Patterns → Monitor Cluster Quality 2.2*

Use Case: The user wants to separate qualitatively acceptable text unit vector clusters from unacceptable ones after clustering as part of the DIAsDEM KDT 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. Both formats are described in Subsection 4.4.3 on page 107.

Result: All discovered text unit vector clusters are visualized as HTML files in *Clustering Directory*. *Cluster Label File* contains default semantic labels for qualitatively acceptable clusters according to the specified *Cluster Quality Criteria*. Additionally, the project properties that represent default values of input parameters are set and updated, respectively. respectively.

Remarks: This task is only executed in the KDT phase of the DIAsDEM framework, as explained in Subsection 1. In this phase, monitoring cluster quality and thereby creating *Cluster Label File* is a prerequisite for subsequently tagging text units using *Actions → Postprocess Patterns → Tag Text Units*. *Thesaurus File* and *Text Unit Descriptors* must exactly correspond to the parameters entered in *Actions → Prepare Data Set → Vectorize Text Units 2.2* in the same clustering iteration.

Monitor Cluster Quality 2.2: Parameters

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

KDT Process Iteration: Number of current KDT process iteration; default value: project property *Default Iteration (1, 2, ...)*

Result File Format: Choice of cluster result file format, as described in Subsection 4.4.3 on page 107, between comma separated values (CSV file) and fixed width values (TXT file); default value: project property *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: project property *Default Cluster Result File*

Cluster Directory: Valid local file name of existing directory or directory to be created by DIAsDEM Workbench; *Cluster Directory* should be empty; default value: project property *Default Cluster Visualization Directory*

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

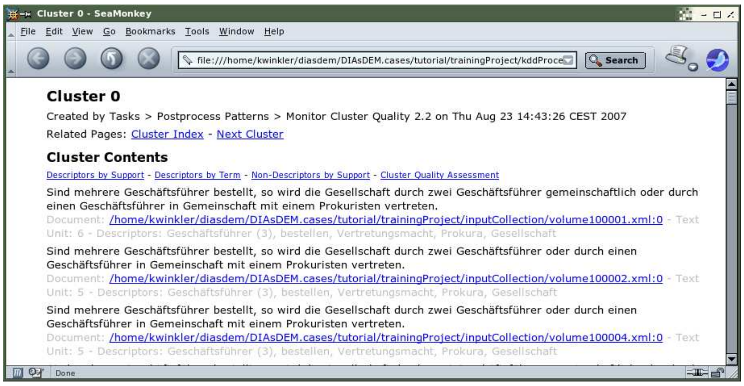
Max. Cluster ID: integer greater than zero; corresponds to the greatest cluster identifier assigned by the clustering algorithm in the current iteration; default value: project property *Default Max. Cluster ID*

Thesaurus File: Valid local file name of existing DIAsDEM-specific thesaurus file, as described in Subsection 4.4.1 on page 104; file extension: `.dth`; default value: project property *Default Thesaurus File*

Text Unit Descriptors: If *All Descriptors in Thesaurus* is enabled, all descriptor terms in *Thesaurus File* are vector dimensions. If *Descriptors whose Scope Note Contain String* is enabled, only descriptor terms in *Thesaurus File* are valid whose scope notes contain the complete 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 string entered below.

Advanced Options: If the first line of *Cluster Result File* is a list of 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 *Rank Clusters by Quality in Cluster Index HTML File* is enabled, clusters are ranked by decreasing quality index in the index file of *Clustering Directory.* If *Launch Web Browser with Cluster Index HTML File* is enabled, the browser specified in *Tools → Options* 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.* If *Dump DIAsDEM Documents for Visualization* is enabled, all documents comprising the collection are exported as XML files in subdirectories of *Cluster Directory.* In this case, HTML files that visualize the content of clusters comprise a direct link to each text unit to its original DIAsDEM document to allow a quick analysis of the text unit context.

Cluster Quality Criteria: Please refer to [GSW01] for description of *Cluster Quality Criteria.* Note that this paper uses a different terminology and refers to *Max. Descriptor Coverage* as *Max. Distinct Ratio*, to *Min. Descriptor Dominance* as *Min. Frequent Ratio*, and to *Min. Cluster Size* as *Min. Cardinality.* In addition, DIAsDEM Workbench 2.2 employs four additional threshold parameters that affect either cluster quality computation


Cluster 0 - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
file:///home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject/kddProce
Search
Done

Cluster 0

Created by Tasks > Postprocess Patterns > Monitor Cluster Quality 2.2 on Thu Aug 23 14:43:26 CEST 2007

Related Pages: Cluster Index - Next Cluster

Cluster Contents

Descriptors by Support - Descriptors by Term - Non-Descriptors by Support - Cluster Quality Assessment

Sind mehrere Geschäftsführer bestellt, so wird die Gesellschaft durch zwei Geschäftsführer gemeinschaftlich oder durch einen Geschäftsführer in Gemeinschaft mit einem Prokuristen vertreten.
Document: /home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject/inputCollection/volume100001.xml:0 - Text Unit: 6 - Descriptors: Geschäftsführer (3), bestellen, Vertretungsmacht, Prokura, Gesellschaft

Sind mehrere Geschäftsführer bestellt, so wird die Gesellschaft durch zwei Geschäftsführer oder durch einen Geschäftsführer in Gemeinschaft mit einem Prokuristen vertreten.
Document: /home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject/inputCollection/volume100002.xml:0 - Text Unit: 5 - Descriptors: Geschäftsführer (3), bestellen, Vertretungsmacht, Prokura, Gesellschaft

Sind mehrere Geschäftsführer bestellt, so wird die Gesellschaft durch zwei Geschäftsführer oder durch einen Geschäftsführer in Gemeinschaft mit einem Prokuristen vertreten.
Document: /home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject/inputCollection/volume100004.xml:0 - Text Unit: 5 - Descriptors: Geschäftsführer (3), bestellen, Vertretungsmacht, Prokura, Gesellschaft

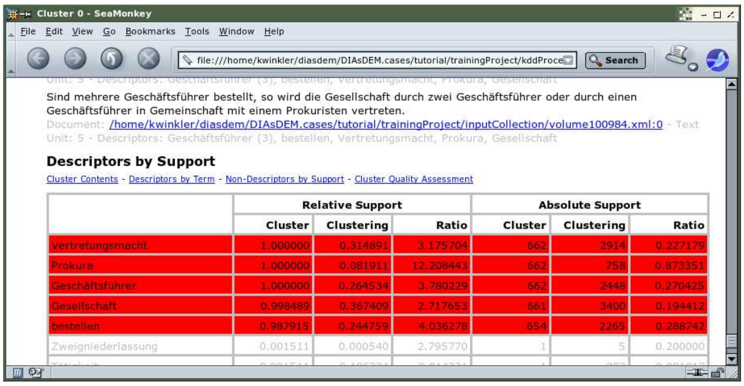
Figure 3.21: Top of HTML File Visualizing the Content of Cluster 3

(i.e., *Dominant Descriptor Threshold* and *Rare Descriptor Threshold*) or cluster visualization output (i.e., *Frequent Non-Descriptor Threshold* and *Max. Number of Output Text Units*).

3.3.7 Editing the Cluster Label File 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 labels finally serve as elements of the XML DTD to be derived. Text units whose vectors are assigned to semantically labeled clusters will be annotated by an XML tag whose name corresponds to the respective cluster label. The remaining text unit vectors are input to the clusterer in the next iteration.

Firstly, qualitatively acceptable clusters should be checked, which have been automatically assigned a default cluster label. In this case study, default German cluster labels have to be replaced by English labels manually. Furthermore, 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 the same cluster. In these cases, default labels should be deleted in *Cluster Label File* to enforce a re-clustering of the corresponding text unit vectors in the next KDT process iteration. Secondly, qualitatively unacceptable clusters should be inspected as well because the applied cluster quality criteria cannot capture all cases of semantic similarity. For instance, a cluster might contain text units that belong to a common semantic concept although there are no statistically prevailing text unit descriptors.


Cluster 0 - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
file:///home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject/kddProce
Search

Unit: 5 - Descriptors: Geschäftsführer (3), bestellen, Vertretungsmacht, Prokura, Gesellschaft

Sind mehrere Geschäftsführer bestellt, so wird die Gesellschaft durch zwei Geschäftsführer oder durch einen Geschäftsführer in Gemeinschaft mit einem Prokuristen vertreten.
Document: /home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject/inputCollection/volume100984.xml:0 - Text Unit: 5 - Descriptors: Geschäftsführer (3), bestellen, Vertretungsmacht, Prokura, Gesellschaft

Descriptors by Support

Cluster Contents - Descriptors by Term - Non-Descriptors by Support - Cluster Quality Assessment

	Relative Support			Absolute Support		
	Cluster	Clustering	Ratio	Cluster	Clustering	Ratio
Vertretungsmacht	1.000000	0.314891	3.175704	662	2914	0.227179
Prokura	1.000000	0.081911	12.208443	662	758	0.873351
Geschäftsführer	1.000000	0.264534	3.780229	662	2448	0.270425
Gesellschaft	0.998489	0.367409	2.717653	661	3400	0.194412
bestellen	0.987915	0.244759	4.036278	654	2265	0.288742
Zweigniederlassung	0.001511	0.000540	2.795770	1	5	0.200000

Figure 3.22: Descriptor Frequencies at the Bottom of Cluster 3 HTML File

Select *Tools* → *Cluster Label Editor* and open the file `${PROJECT_HOME}/1labels.dcl` by clicking on *Open* and choosing this cluster label file. As depicted in Figure 3.20, this file contains default cluster labels assigned during the first iteration. Additionally, open the index HTML file `${PROJECT_HOME}/kddProcessIteration1/index.html`. For each cluster, there exists an HTML file in the same directory that visualizes the cluster content and that provides descriptive statistics of frequently occurring text unit descriptors.

Altogether, 21 qualitatively acceptable clusters and 79 non-empty unacceptable ones have been automatically discovered by the preceding task. For example, Figures 3.21 and 3.22 illustrate the HTML file visualizing the qualitatively acceptable cluster 0, which has been assigned the label "DEFAULT_Vertretungsmacht_Prokura_Geschaeftsfuehrer_Gesellschaft_bestellen". This default label has been created by concatenating text unit descriptors that prevail in cluster 0 and that are highlighted in Figure 3.22. Change the label of cluster 0 into its English equivalent "IfAppointmentOfManyManagingDirectors_JointPowerToRepresent" in *Cluster Label Editor*, modify the cluster quality status from "a/?" (i.e., regarded as acceptable by DIAsDEM Workbench and no expert assessment yet) to "a/a" (i.e., regarded as acceptable by both DIAsDEM Workbench and expert), and click the *Save* button. Consider cluster 90, which is listed in the section "Qualitatively Unacceptable Clusters" of the file `${PROJECT_HOME}/kddProcessIteration1/index.html`. The German concept "Tätigkeit" occurs along with the contextually related concepts "Unternehmen" and "Beteiligung" in all members of cluster 90. Therefore, this cluster can be manually labeled with its English equivalent "PurposeOfCompany". Furthermore, change the cluster quality status of cluster 90 from "u/?" (i.e., regarded as unacceptable by DIAsDEM Workbench and no expert assessment yet) to "u/a" (i.e., regarded as acceptable by expert despite other assessment of DIAsDEM Workbench). Inspect the remaining clusters and modify their semantic labels in *Cluster Label File* according to Table 3.2. Finally, close the Web browser, save *Cluster Label*

File by clicking on *Save*, and close *Cluster Label Editor* by clicking the *Exit* button.

Cluster	Quality	Semantic Cluster Label
0	a/a	IfAppointmentOfManyManagingDirectors_JointPowerToRepresent
1	a/a	SolePowerToRepresentCanBeGranted
2	a/a	PurposeOfCompany
4	a/a	ResolutionByShareholders_ChangeOfPlaceOfDomicile
6	a/a	ShareCapital
7	a/a	LimitedLiabilityCompany
8	a/a	IfAppointmentOfOneManagingDirector_SolePowerToRepresent
12	a/a	PublicationMediaOfCommercialRegisterEntries
13	a/a	SolePowerToRepresent_PowerToContractWithOneself
14	a/a	ConclusionAndModificationOfPartnershipAgreement
15	a/a	ConclusionOfPartnershipAgreement
20	a/a	AppointmentOfManagingDirector
21	a/a	PurposeOfCompany
22	a/u	-
25	a/a	SolePowerToRepresent_PowerToContractWithOneself
26	a/a	CommencementOfPartnership
31	a/a	LimitedPartnership
37	a/u	-
52	a/a	NumberOfLimitedPartners
53	a/a	AppointmentOfManagingDirector
56	a/a	PowerToContractWithOneself
90	u/a	PurposeOfCompany

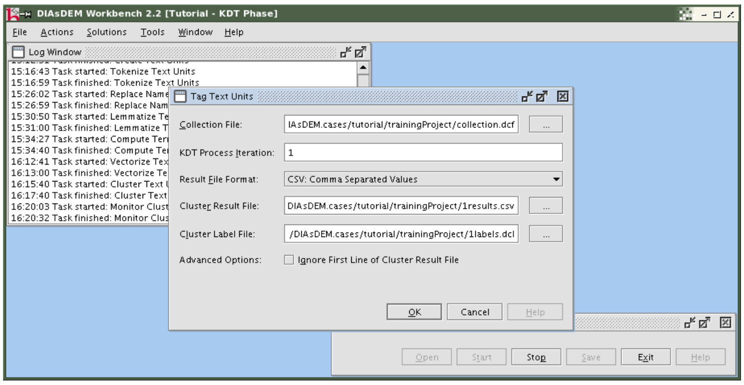
Table 3.2: Summary of Semantic Cluster Labels in the First Iteration

Cluster Label Editor of DIAsDEM Workbench 2.2 supports so-called tag proposal files to minimize human typing efforts. Tag proposal files can either be text files or DIAsDEM-specific files containing a previously derived, concept-based document type definition. For example, copy the cluster labels listed in Table 3.2 into an empty text file such that each line exactly contains one label. After clicking the button *Tags* and selecting this new tag proposal file, *Cluster Label Editor* creates a drop-down menu beside each cluster label field. All drop-down menus comprise the same list of potentially useful cluster labels for the expert to choose from. Furthermore, tag proposal files can be imported into *Cluster Label Editor* one after the other.

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 DIAsDEM documents associated with *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. In addition, members of qualitatively acceptable and thus labeled clusters have to be annotated with the respective semantic label 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 output XML documents) after the final clustering iteration. Hence, select *Actions* → *Postprocess Patterns* → *Tag Text Units* and type in the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
KDT Process Iteration	`1`
Result File Format	`CSV: Comma Separated Values`
Cluster Result File	`${PROJECT_HOME}/1results.csv`
Cluster Label File	`${PROJECT_HOME}/1labels.dcl`
Advanced Options	`Disabled: Ignore First Line of Cluster Result File`



```
15:16:43 Task started: Tokenize Text Units
15:16:59 Task finished: Tokenize Text Units
15:26:02 Task started: Replace Name
15:26:59 Task finished: Replace Nam
15:30:50 Task started: Lemmatize Te
15:31:00 Task finished: Lemmatize T
15:34:27 Task started: Compute Ter
15:34:40 Task finished: Compute Te
16:12:41 Task started: Vectorize Tex
16:13:00 Task finished: Vectorize Te
16:15:40 Task started: Cluster Text U
16:17:40 Task finished: Cluster Text
16:20:03 Task started: Monitor Clust
16:20:32 Task finished: Monitor Clus
```

Collection File: IAsDEM.cases/tutorial/trainingProject/collection.dcf
KDT Process Iteration: 1
Result File Format: CSV: Comma Separated Values
Cluster Result File: DIAsDEM.cases/tutorial/trainingProject/1results.csv
Cluster Label File: /DIAsDEM.cases/tutorial/trainingProject/1labels.dcl
Advanced Options: □ Ignore First Line of Cluster Result File

Figure 3.23: *Tag Text Units* Dialog

The parameters *KDT Process Iteration*, *Result File Format*, and *Cluster Result File* have been discussed in Subsection 3.3.6 in the context of monitoring cluster quality. *Cluster Label File* corresponds to the file that has been created by *Actions* → *Postprocess Patterns* → *Monitor Cluster Quality 2.2*. Click on *OK* to tag all text units accordingly. Thereafter, open the file `${PROJECT_HOME}/inputCollection/volume100878.xml`. The elements of section `<ProcessedTextUnits>` mark up the same content as before. However, they have been extended by the attributes `Iteration`, `ClusterID`, and `ClusterLabel` to keep track of cluster assignments. Values of `ClusterLabel` equal either "-" for unlabeled clusters or correspond to the semantic label of the respective cluster.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DefaultDIAsDEMvolume SYSTEM "DefaultDIAsDEMvolume.dtd">
<DefaultDIAsDEMvolume NumberOfDocuments="1">
  <DefaultDIAsDEMdocument NumberOfTextUnitsLayers="0"> ...
    <TextUnitsLayer TextUnitsLayerID="0" TextUnitsDescription="Algorithm:
    HEURISTIC_SENTENCE_IDENTIFIER"> ...
      <ProcessedTextUnits> ...
        <ProcessedTextUnit TextUnitID="2" Iteration="1" ClusterID="6"
        ClusterLabel="ShareCapital">Stammkapital : <NeRef NeID="0" /> ...
        <ProcessedTextUnit TextUnitID="8" Iteration="1" ClusterID="1"
        ClusterLabel="SolePowerToRepresentCanBeGranted-">
        Einzelvertretungsbefugnis können erteilen werden .
        </ProcessedTextUnit><ProcessedTextUnit TextUnitID="9" Iteration="1"
        ClusterID="20" ClusterLabel="AppointmentOfManagingDirector"><NeRef
        NeID="16" />, sein zur Geschäftsführerin bestellen .</ProcessedTextUnit>
       ... <ProcessedTextUnit TextUnitID="11" Iteration="1" ClusterID="12"
       ClusterLabel="PublicationMediaOfCommercialRegisterEntries">nicht
       eintragen : d Bekanntmachung d Gesellschaft erfolgen im Bundesanzeiger
       .</ProcessedTextUnit>
      </ProcessedTextUnits> ...
    </TextUnitsLayer>
  </DefaultDIAsDEMdocument>
</DefaultDIAsDEMvolume>
```

Consider the first text unit shown in the file excerpt above, which corresponds to the original sentence “Stammkapital: 50.000 DM.” Its text unit vector has been assigned to cluster 6, which in turn has been labeled “ShareCapital”. Thus, this sentence is subsequently tagged as “`<ShareCapital>` Stammkapital: 50.000 DM. `</ShareCapital>`”. Furthermore, 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 20, which has been semantically labeled “AppointmentOfManagingDirector”. This exemplary text document does not comprise any unlabeled text units after the first clustering iteration. Altogether, 1,462 text unit vectors that correspond to unlabeled text units are input to the second clustering iteration. Note, all text unit vectors representing annotated text units are not re-clustered in the next iteration. Once a semantic label has been attached to a text unit, it can only be changed by the auxiliary task *Actions → Miscellaneous → Replace Labels of Text Units.*

Executing the second clustering iteration is concisely described in the next subsection. Subsequently, Subsections 3.4.1 and 3.4.2 introduce the tasks for deriving a concept-based XML document type definition and finally creating semantically tagged XML documents.

Tag Text Units: Summary

Task: *Actions → Postprocess Patterns → Tag Text Units*

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

Prerequisites: The default text units layer of each DIAsDEM document must contain the section `<ProcessedTextUnits>`. Moreover, clustering results in *Cluster Result File* must conform either to the DIAsDEM-specific CSV or to the DIAsDEM-specific TXT file format. Both file formats are described in Subsection 4.4.3 on page 107.

Result: In the first clustering iteration, the attributes `Iteration`, `ClusterID`, and `ClusterLabel` of all processed text units are created or reset. In subsequent iterations, the section `<ProcessedTextUnits>` is updated. In both cases, text units referred to in *Cluster Result File* are annotated with the iteration number, their current cluster ID, and the corresponding label according to *Cluster Label File*. Additionally, the project properties *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 either starting the next clustering iteration or for finally executing the tasks *Actions* → *Postprocess Patterns* → *Derive Conceptual DTD 2.2* and thereafter *Actions* → *Postprocess Patterns* → *Tag Documents 2.2*.

Tag Text Units: Parameters

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

KDT Process Iteration: Number of current KDT process iteration; default value: project property *Default Iteration (1, 2, ...)*

Result File Format: Choice of cluster result file format, as described in Subsection 4.4.3 on page 107, between comma separated values (CSV file) and fixed width values (TXT file); default value: project property *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: project property *Default Cluster Result File*

Cluster Label File: Valid local file name of existing file created by DIAsDEM Workbench in *Actions* → *Postprocess Patterns* → *Monitor Cluster Quality 2.2* and possibly modified by *Tools* → *Cluster Label Editor*; file extension: `.dcl`; default value: project property *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 KDD Process Iteration 2

Text units are re-clustered in iteration 2 if their vectors have not been assigned to a qualitatively acceptable cluster in the first iteration. Consequently, text unit vectors corresponding to unlabeled sentences need to be exported and clustered again. 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 KDT 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, which constitute the input data set to iteration 2, are exported. Please select *Actions* → *Prepare Data Set* → *Vectorize Text Units 2.2*, type in the following parameters, and click on *OK*.

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
KDT Process Iteration	`2`
Text Unit Vectors Format	`ARFF: Weka Data Mining Project`
Text Unit Vectors File	`${PROJECT_HOME}/2vectors.arff`
Thesaurus File	`${PARAMETER_HOME}/thesauri/de/Case123Thesaurus.dth`
Text Unit Descriptors	`Descriptors whose Scope Notes Contain String Case1`
Descriptor Frequency	`Boolean Descriptor Frequency`
Collection Frequency	`Inverse Collection Frequency: Create New File`
Collection Frequencies File	`${PROJECT_HOME}/2weights.dcfq`
Length Normalization	`No Length Normalization`

Advanced Options	`Disabled: Create File for Mining Descriptor Association Rules`
	`Enabled: Create Metadata File for Text Unit Vectors File`

Compared to the first iteration, *Text Unit Vectors File* contains only 16% of all text unit vectors. Note additionally, collection-based term weights, such as inverse document frequencies, 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 metadata files `${PROJECT_HOME}/1vectors.arff.meta` and `2vectors.arff.meta`, respectively. To cluster exported text unit vectors, select *Actions* → *Discover Patterns* → *Cluster Text Unit Vectors (Weka)*, enter the following parameters, and click on the *OK* button. In contrast to iteration 2, the maximum number of clusters to be discovered by the k-means algorithm is $k = 50$.

Parameter	Value
Clustering Mode	`Clustering Phase (Create New Clustering Model)`
Collection File	`${PROJECT_HOME}/collection.dcf`
Text Unit Vectors File	`${PROJECT_HOME}/2vectors.arff`
Clustering Algorithm	`weka.clusterers.SimpleKMeans`
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	`${PARAMETER_HOME}/2results.csv`
Text Unit Clusterer File	`${PARAMETER_HOME}/2clusterer.wskm`

Analogously to iteration 1, clustering text unit vectors is followed by monitoring the cluster quality. Hence, select *Actions → Postprocess Patterns → Monitor Cluster Quality 2.2*, submit the following parameters, and click on *OK*. Compared with the first iteration, the minimum cluster size is decreased from 50 to 25.

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
KDT Process Iteration	`2`
Result File Format	`CSV: Comma Separated Values`
Cluster Result File	`${PROJECT_HOME}/2results.csv`
Cluster Directory	`${PROJECT_HOME}/kddProcessIteration2`
Cluster Label File	`${PROJECT_HOME}/2labels.dcl`
Max. Cluster ID	`49`
Thesaurus File	`${PARAMETER_HOME}/thesauri/de/Case123Thesaurus.dth`
Text Unit Descriptors	`Descriptors whose Scope Notes Contain String`
	`Case1`
Advanced Options	`Disabled: Ignore First Line of Cluster Result File`
	`Enabled: Ignore Empty Clusters in Cluster Index HTML File`
	`Enabled: Rank Clusters by Quality in Cluster Index HTML File`
	`Enabled: Launch Web Browser with Cluster Index HTML File`
	`Enabled: Launch Cluster Label Editor with Cluster Label File`
	`Enabled: Dump DIAsDEM Documents for Visualization`
Cluster Quality Criteria	`1) Dominant Descriptor Threshold = 0.8`
	`2) Rare Descriptor Threshold = 0.01`
	`3) Max. Descriptor Coverage = 0.75`
	`4) Min. Descriptor Dominance = 0.25`
	`5) Min. Cluster Size = 25`
	`6) Frequent Non-Descriptor Threshold = 0.2`
	`7) Max. Number of Output Text Units = 1000`

After monitoring cluster quality, the Web browser pops up and displays the HTML file `${PROJECT_HOME}/kddProcessIteration2/index.html`, which references all non-empty cluster files in the same directory. Additionally, *Cluster Label Editor* is launched within DIAsDEM Workbench. DIAsDEM Workbench has automatically discovered nine qualitatively acceptable clusters only. Please have a look at these clusters and modify the file `${PROJECT_HOME}/2labels.dcl` in *Cluster Label Editor* according to Table 3.3.

Cluster ID	Quality	Semantic Cluster Label
6	a/u	-
8	a/a	NameOfMerchant
9	a/a	ResolutionByShareholders_ChangeOfPlaceOfDomicile
15	a/a	ConfermentOfProkura
25	a/a	PurposeOfCompany
26	a/a	PublicationMediaOfCommercialRegisterEntries
34	a/a	FullyLiablePartner
39	a/u	-
42	a/a	ChangeOfFirmName

Table 3.3: Summary of Semantic Cluster Labels in the Second Iteration

After editing `${PROJECT_HOME}/2labels.dcl`, select *Actions → Postprocess Patterns → Tag Text Units*, type in the following parameters, and click the *OK* button to annotate text units accordingly:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
KDT Process Iteration	`2`
Result File Format	`CSV: Comma Separated Values`
Cluster Result File	`${PROJECT_HOME}/2results.csv`
Cluster Label File	`${PROJECT_HOME}/2labels.dcl`
Advanced Options	`Disabled: Ignore First Line of Cluster Result File`

Open the file `${PROJECT_HOME}/inputCollection/volume100878.xml`. Note that annotations and cluster IDs assigned in the first iteration remain untouched. For example, the original sentence “Stammkapital: 50.000 DM.” is still assigned to cluster 6 of the first iteration and is still labeled “ShareCapital”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DefaultDIAsDEMvolume SYSTEM "DefaultDIAsDEMvolume.dtd">
<DefaultDIAsDEMvolume NumberOfDocuments="1">
  <DefaultDIAsDEMdocument NumberOfTextUnitsLayers="0"> ...
    <TextUnitsLayer TextUnitsLayerID="0" TextUnitsDescription="Algorithm:
    HEURISTIC_SENTENCE_IDENTIFIER"> ...
      <ProcessedTextUnits> ...
        <ProcessedTextUnit TextUnitID="2" Iteration="1" ClusterID="6"
        ClusterLabel="ShareCapital">Stammkapital : <NeRef NeID="0" /> ...
        <ProcessedTextUnit TextUnitID="8" Iteration="1" ClusterID="1"
        ClusterLabel="SolePowerToRepresentCanBeGranted-">
        Einzelvertretungsbefugnis können erteilen werden .
        </ProcessedTextUnit><ProcessedTextUnit TextUnitID="9" Iteration="1"
        ClusterID="20" ClusterLabel="AppointmentOfManagingDirector"><NeRef
        NeID="16" />, sein zur Geschäftsführerin bestellen .</ProcessedTextUnit>
       ... <ProcessedTextUnit TextUnitID="11" Iteration="1" ClusterID="12"
       ClusterLabel="PublicationMediaOfCommercialRegisterEntries">nicht
       eintragen : d Bekanntmachung d Gesellschaft erfolgen im Bundesanzeiger
       .</ProcessedTextUnit>
      </ProcessedTextUnits> ...
    </TextUnitsLayer>
  </DefaultDIAsDEMdocument>
</DefaultDIAsDEMvolume>
```

In this case study, only two clustering iterations are performed to exemplify the interactive and iterative DIAsDEM knowledge discovery process. When applying the DIAsDEM KDT process to real document archives, the iterative clustering should be continued until further qualitatively acceptable clusters cannot be discovered. Nevertheless, 7,792 (84.2%) of altogether 9,254 text units have been assigned a semantic label in the training phase of this case study.

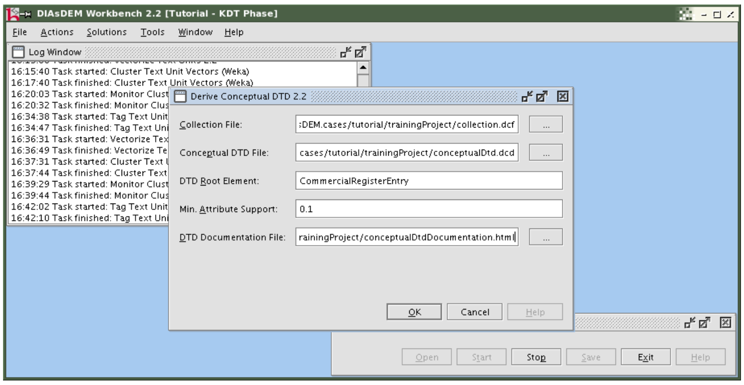
3.4 XML Tagging of Texts in the KDT Phase

After finishing the second, in our case the final clustering iteration, text documents have to be converted into an archive of semantically tagged XML documents to attain the objectives of the DIAsDEM framework. Hence, a collection-specific and concept-based XML document type definition has to be derived in the postprocessing phase of the DIAsDEM knowledge discovery process. Thereafter, semantically tagged XML documents, which conform to this XML DTD, can be constructed from the collection of intermediate DIAsDEM documents.

3.4.1 Establishing a Concept-Based XML DTD

A concept-based XML document type definition concisely describes frequently occurring, 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. To continue, select the task *Actions → Postprocess Patterns → Derive Conceptual DTD 2.2* and input the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
Conceptual DTD File	`${PROJECT_HOME}/conceptualDtd.dcd`
DTD Root Element	`CommercialRegisterEntry`
Min. Attribute Support	`0.1`
DTD Documentation File	`${PROJECT_HOME}/conceptualDtdDocumentation.html`



```
16:15:40 Task started: Cluster Text Unit Vectors (Weka)
16:17:40 Task finished: Cluster Text Unit Vectors (Weka)
16:20:03 Task started: Monitor Clust
16:20:32 Task finished: Monitor Clus
16:34:38 Task started: Tag Text Unit
16:34:47 Task finished: Tag Text Uni
16:36:31 Task started: Vectorize Tex
16:36:49 Task finished: Vectorize Te
16:37:31 Task started: Cluster Text U
16:37:44 Task finished: Cluster Text
16:39:29 Task started: Monitor Clust
16:39:44 Task finished: Monitor Clus
16:42:02 Task started: Tag Text Unit
16:42:10 Task finished: Tag Text Uni
```

Collection File: :DEM.cases/tutorial/trainingProject/collection.dcf

Conceptual DTD File: cases/tutorial/trainingProject/conceptualDtd.dcd

DTD Root Element: CommercialRegisterEntry

Min. Attribute Support: 0.1

DTD Documentation File: rainingProject/conceptualDtdDocumentation.html

Figure 3.24: *Derive Conceptual DTD 2.2* Dialog

Note, *Conceptual DTD File* is a DIAsDEM-specific file that contains metadata about the XML DTD for internal usage only. According to *DTD Root Element*, the term `CommercialRegisterEntry` is the root element of the XML document type definition. Hence, `CommercialRegisterEntry` is the root tag of output XML documents, which are created afterwards. Due to *Min. Attribute Support*, named entity type e (e.g., "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. Click the *OK* button to derive the DIAsDEM-specific *Conceptual DTD File*, which is a required input parameter for tasks that create final, semantically annotated output documents, such as *Actions → Postprocess Patterns → Tag Documents 2.2*.

`${PROJECT_HOME}` now contains five new files, but only `conceptualDtd.dcd` is referred to as *Conceptual DTD File*. The auxiliary files `conceptualDtd.dcd.elements`, `conceptualDtd.dcd.attributes`, and `conceptualDtd.dcd.xml` are internally referenced by *Conceptual DTD File* and must thus always reside in the same directory. The generated DTD documentation template `conceptualDtdDocumentation.html` contains an enumeration of valid attributes and five exemplary text units for each DTD element. DIAsDEM Workbench 2.2 derives a conceptual XML document type definition that enumerates occurring DTD elements (i.e., XML tags) and attributes associated with XML tags. Using any text editor, open the DTD file `conceptualDtd.dcd.xml`, which is located in `${PROJECT_HOME}`.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!ELEMENT CommercialRegisterEntry (MetaData*, TaggedDocument)>

<!ELEMENT MetaData (Name, Content)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Content (#PCDATA)>

<!ELEMENT TaggedDocument ( #PCDATA | AppointmentOfManagingDirector
 | ChangeOfFirmName | CommencementOfPartnership | ... | ConfermentOfProkura
 | FullyLiablePartner | ... | LimitedLiabilityCompany | LimitedPartnership
 | NameOfMerchant | NumberOfLimitedPartners | PowerToContractWithOneself
 | ... | ShareCapital | ... | SolePowerToRepresent_PowerToContractWithOneself
)* >

<!ELEMENT AppointmentOfManagingDirector (#PCDATA)>
<!ELEMENT ChangeOfFirmName (#PCDATA)>
<!ELEMENT CommencementOfPartnership (#PCDATA)> ...
<!ELEMENT SolePowerToRepresent_PowerToContractWithOneself (#PCDATA)>

<!ATTLIST AppointmentOfManagingDirector Date CDATA #IMPLIED>
<!ATTLIST AppointmentOfManagingDirector Person CDATA #IMPLIED>
<!ATTLIST AppointmentOfManagingDirector Place CDATA #IMPLIED>
<!ATTLIST ChangeOfFirmName Company CDATA #IMPLIED> ...
<!ATTLIST ShareCapital AmountOfMoney CDATA #IMPLIED>
```

Valid output documents (i.e., annotated Commercial Register entries) consists of two main sections: Metadata stored in DIAsDEM documents is copied into the optional section `<MetaData>` to facilitate further data processing. The mandatory section `<TaggedDocument>` includes semantically annotated text. Elements of the latter section are defined as a listing of unordered DTD elements such that semantic tags can occur anywhere in the text. Furthermore, attributes of XML tags are defined as well. For example, the XML tag `AppointmentOfManagingDirector` has three optional attributes `Date`, `Person`, and `Place`. Due to the setting of *Min. Attribute Support*, instances of named entity types “date”, “person”, and “place” occur in at least 10% of all text units annotated with `AppointmentOfManagingDirector`. Note, semantically tagged XML documents created by *Actions → Postprocess Patterns → Tag Documents 2.2* are valid XML documents with respect to this DTD. Attributes of XML tags cannot be semantically named in this release of DIAsDEM Workbench.

Derive Conceptual DTD 2.2: Summary

Task: *Actions → Postprocess Patterns → Derive Conceptual DTD 2.2*

Use Case: The user wants to derive a concept-based XML DTD from semantically annotated DIAsDEM documents as part of the DIAsDEM KDT process for semantic tagging of domain-specific texts archives.

Prerequisites: The default text units layer of each DIAsDEM document must contain the section `<ProcessedTextUnits>`. All elements `<ProcessedTextUnit>` must have the attributes `Iteration`, `ClusterID`, and `ClusterLabel`.

Results: A collection-specific XML document type definition is derived, which enumerates valid XML tags and their attributes. Additionally, the project properties that correspond to the input parameters are set and updated, respectively.

Remarks: After deriving the collection-specific, concept-based DTD, semantically annotated XML documents can be output by *Actions → Postprocess Patterns → Tag Documents 2.2*.

Derive Conceptual DTD 2.2: Summary

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

Conceptual DTD File: Valid local file name of file to be created or replaced by DIAsDEM Workbench; file extension: `.dcd`

DTD Root Element: ISO-8859-1 encoded string without blank spaces; default value: project property *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: project property *Default Min. Attribute Support*

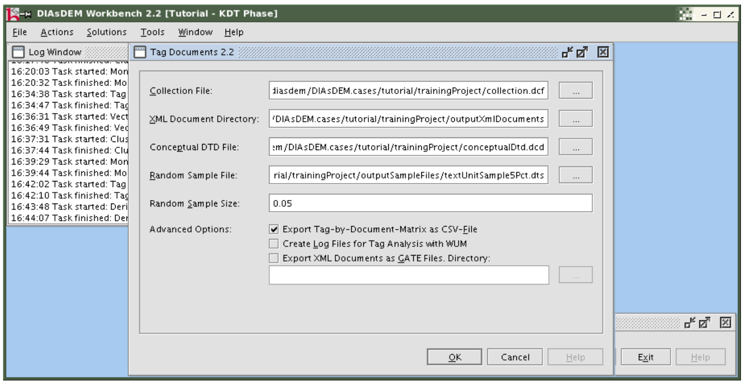
DTD Documentation File: Valid local file name of file to be created or replaced by DIAsDEM Workbench; file extension: `.html`

3.4.2 Tagging Documents

After deriving an archive-specific document type definition, output 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 in the original text. Besides the derived XML document type definition and the generated text unit clusterer for subsequent batch processing, semantically tagged XML documents constitute the main output of the DIAsDEM KDT process. To proceed, select *Actions → Postprocess Patterns → Tag Documents 2.2* and enter the following parameters:

Parameter	Value
Collection File	`${PROJECT_HOME}/collection.dcf`
XML Document Directory	`${PROJECT_HOME}/outputXmlDocuments`
Conceptual DTD File	`${PROJECT_HOME}/conceptualDtd.dcd`
Random Sample File	`${PROJECT_HOME}/outputSampleFiles/textUnitSample5Pct.dts`
Random Sample Size	`0.05`

Advanced Options	`Enabled: Create Tag-by-Document-Matrix as CSV-File`
	`Disabled: Create Log Files for Tag Analysis with WUM`
	`Disabled: Export XML Documents as GATE Files. Directory:`



```
16:20:03 Task started: Mon
16:20:32 Task finished: Mo
16:34:38 Task started: Tag
16:34:47 Task finished: Tag
16:36:31 Task started: Vect
16:36:49 Task finished: Vec
16:37:31 Task started: Clus
16:37:44 Task finished: Clu
16:39:29 Task started: Mon
16:39:44 Task finished: Mo
16:42:02 Task started: Tag
16:42:10 Task finished: Tag
16:43:48 Task started: Deri
16:44:07 Task finished: Der
```

Collection File: diasdem/DIAsDEM.cases/tutorial/trainingProject/collection.dcf

XML Document Directory: 'DIAsDEM.cases/tutorial/trainingProject/outputXmlDocuments

Conceptual DTD File: em/DIAsDEM.cases/tutorial/trainingProject/conceptualDtd.dcd

Random Sample File: rial/trainingProject/outputSampleFiles/textUnitSample5Pct.dts

Random Sample Size: 0.05

Advanced Options:
☑ Export Tag-by-Document-Matrix as CSV-File
☐ Create Log Files for Tag Analysis with WUM
☐ Export XML Documents as GATE Files. Directory:

Figure 3.25: *Tag Documents 2.2* Dialog

Click the *OK* button to start the semantic tagging of documents. Besides outputting semantically tagged XML documents in subdirectories of *XML Document Directory*, the task *Actions* → *Postprocess Patterns* → *Tag Documents 2.2* draws a 5% random text unit sample. This sample is saved in the DIAsDEM-specific *Random Sample File* for subsequent evaluation of the tagging quality. For each intermediate DIAsDEM document in the collection, a new XML file is output that contains semantically annotated content of the corresponding text. For example, the result XML document `${PROJECT_HOME}/outputXmlDocuments/part1/document879.xml` is depicted below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE CommercialRegisterEntry SYSTEM "CommercialRegisterEntry.dtd">
<CommercialRegisterEntry>
  <MetaData>
    <Name>DiasdemDocumentID</Name>
    <Content>/home/.../trainingProject/inputCollection/volume100878.xml:0</Content>
  </MetaData>
  <MetaData>
    <Name>SourceFile</Name>
    <Content>/home/.../data/samples/de/case1/file10780.training.txt</Content>
  </MetaData>
  <TaggedDocument>
    <PurposeOfCompany>Der Handel mit Waren aller Art sowie Import und Export.
    </PurposeOfCompany> ... <ShareCapital AmountOfMoney="50000 DEM">Stammkapital:
    50.000 DM.</ShareCapital><LimitedLiabilityCompany>Gesellschaft mit beschränkter
    Haftung.</LimitedLiabilityCompany> ... <SolePowerToRepresentCanBeGranted>
    Einzelvertretungsbefugnis kann erteilt werden.</SolePowerToRepresentCanBeGranted>
    <AppointmentOfManagingDirector Person="16; Marion Marcella Adolph; null; null;
    null; 22.03.1957; Priester; Offenbach; null; null">Marion Marcella Adolph geb.
    Priester, 22.03.1957, Offenbach, ist zur Geschäftsführerin bestellt.
    </AppointmentOfManagingDirector> ... <PublicationMediaOfCommercialRegisterEntries>
    Nicht eingetragen: Die Bekanntmachungen der Gesellschaft erfolgen im
    Bundesanzeiger.</PublicationMediaOfCommercialRegisterEntries>
  </TaggedDocument>
</CommercialRegisterEntry>
```

The maximum number of result files per subdirectory of *XML Document Directory* is determined by the project property *Maximum Files per Directory*. The supplementary *Random Sample File* contains a random sample of text units for subsequent evaluation of tagging quality using *Tools → Tagging Quality Evaluator 2.2*. In addition, a random sample of completely tagged documents can be created by the task *Actions → Postprocess Patterns → Draw Document Sample 2.2*. Open `${PROJECT_HOME}/outputXmlDocuments/TagByDocumentMatrix.csv`, which is partly shown below. Each line contains a relational representation of semantic XML tags that occur in a certain text document. For example, the XML tag `AppointmentOfManagingDirector` occurs in the XML file created from source document `/home/.../file10780.training.txt`, whereas the tag `ChangeOfFirmName` does not occur in this file. This CSV file can easily be imported in any relational database for further analysis.

```
DiasdemDocumentID,SourceFile,AppointmentOfManagingDirector,ChangeOfFirmName,...
"/home/.../volume100000.xml:0","/home/.../file10511.training.txt",1,0,0,1,0,...
"/home/.../volume100001.xml:0","/home/.../file10338.training.txt",1,0,0,1,0,...
...
"/home/.../volume100878.xml:0","/home/.../file10780.training.txt",1,0,0,1,0,...
...
```

Tag Documents 2.2: Summary

Task: *Actions → Postprocess Patterns → Tag Documents 2.2*

Use Case: The user wants to create result XML documents from semantically annotated DIAsDEM documents as part of the DIAsDEM KDT process for semantic tagging of domain-specific texts archives.

Prerequisites: The default text units layer of each DIAsDEM document must contain the section `<ProcessedTextUnits>`. All elements `<ProcessedTextUnit>` must have the attributes `Iteration`, `ClusterID`, and `ClusterLabel`. A collection-specific, concept-based XML DTD must have been derived.

Result: For each intermediate DIAsDEM document, a semantically annotated XML file is output. Additionally, the project properties that correspond to the input parameters are set and updated, respectively.

Remarks: After tagging result documents, only one task remains to be accomplished: The quality of semantic tags should be evaluated using the task *Tools → Tagging Quality Evaluator 2.2*.

Tag Documents 2.2: Parameters

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

Conceptual DTD File: Valid local file name of exiting file; file extension: `.dcd`

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

Random Sample Size: Floating point number in the interval $[0;1]$; proportion of text units to be randomly drawn for quality evaluation; default value: project property *Default Random Sample Size*

Advanced Options: If *Create Tag-by-Document-Matrix as CSV-File* is enabled, a file `TagByDocumentMatrix.csv` is created in *XML Document Directory* that contains a relational mapping of source file names onto discovered XML tags. If *Create Log Files for 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 *Export XML Documents as GATE Files* is enabled, semantically annotated GATE files are exported into the specified directory.

3.4.3 Evaluating the Tagging Quality

Finally, the quality of semi-automatically created semantic XML markup needs to be evaluated. Due to the absence of pre-tagged documents, a random sample of both tagged and untagged text units has been drawn by *Actions → Postprocess Patterns →*

Tag Documents 2.2 to allow a quality assessment. Now, a domain expert is asked to evaluate the markup of one semantically marked-up text unit after the other. Each text unit in the random sample is displayed exactly as contained in the corresponding semantically marked-up XML document. Based on his or her domain expertise, the expert evaluates the correctness of the (possibly non-existing) XML tag name with respect to the accompanying concept-based DTD, which can be viewed instantaneously, and the following four mutually exclusive and collectively exhaustive cases:

- *True Positive*: Text units enclosed in semantic XML tags are true positives if their tag names represent correct concepts that domain experts typically associate with the respective, marked-up text units.
- *False Positive*: If the semantic XML tag enclosing a text unit does not correspond to the concept that domain experts typically associate with the marked-up text unit, a false positive occurs.
- *True Negative*: A plain text unit not enclosed in a semantic XML tag is a true negative if the concept-based XML document type definition does not comprise a tag for the concept that domain experts typically associate with this text unit.
- *False Negative*: A false negative occurs when a plain text unit, which is not enclosed in a semantic XML tag, in fact represents a semantic concept that is listed in the concept-based XML document type definition.

If the automatically assigned XML tag name is incorrect, the correct one from the finite tag set in the concept-based DTD is input as well. For text units enclosed in semantic XML tags, the domain expert is further asked to assess the performance in extracting the relevant named entities. To that end, the true number of relevant named entities as well as the number of correct, partially correct, incorrect, and missing named entities is input for each XML attribute that is listed in the tag-specific DTD attribute definition. Analogous to the assessment of information extraction as performed in the seminal GATE project [MTU+01, CMB+02, pp. 114–115], we distinguish four mutually exclusive and collectively exhaustive types of identified named entities within XML tag attributes:

- *Completely Correct Named Entity*: If an attribute value comprises all components of a named entity without any extraction flaws, it is considered to be completely correct.
- *Partially Correct Named Entity*: If an attribute value does not contain a few tokens of the corresponding, typically complex named entity without conveying erroneous information (e.g., concerning its type), it is partially correct.
- *Incorrect Named Entity*: If an attribute value conveys erroneous information about a named entity, it represents an incorrect named entity. For example, a stand-alone basic named entity (e.g., "place") that is actually an integral part of a composite named entity (e.g., "company") is incorrectly extracted.

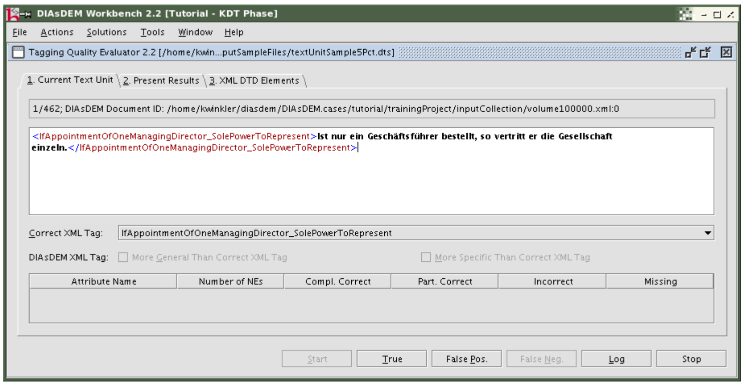
- *Missing Named Entity*: An existing named entity, whose type is listed in the tag-specific XML DTD attribute list, is considered to be a missing one if this entity is not referenced by an attribute.

In addition, all evaluation results are persistently stored in the file system, the entire process can be stopped and resumed at any time, all assessment decisions are documented in a protocol, and the markup quality measures defined in this section are finally computed for the entire random sample. Start the quality evaluation by selecting *Tools* → *Tagging Quality Evaluator 2.2*, clicking the *Start* button, and choosing the following five parameter files one after the other:

Parameter	Value
Existing Text Unit Sample File	`${PROJECT_HOME}/outputSamplesFile/textUnitSample5Pct.dts`
New or Existing File of Evaluated Text Units	`${PROJECT_HOME}/outputSamplesFile/evaluatedTextUnits.det`
Text Unit Sample File to be Created for Next Evaluation	`${PROJECT_HOME}/outputSamplesFile/textUnitSample5PctB.dts`
Existing Conceptual DTD File	`${PROJECT_HOME}/conceptualDtd.dcd`
New or Existing Log for Personal Notes	`${PROJECT_HOME}/outputSamplesFile/evaluationLog.txt`

The parameter files *Existing Text Unit Sample File* and *Existing Conceptual DTD File* have been created before. They are concisely described in Subsections 3.4.1 and 4.5.2, respectively. Evaluating tagging quality can be a lengthy task, even for rather small text unit 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 expert'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 session, this file *Text Unit Sample File to be Created for Next Evaluation* must be chosen as *Existing Text Unit Sample File*.

Figure 3.26 depicts the *Tagging Quality Evaluator 2.2* tool after opening the *Existing Text Unit Sample File* `textUnitSample5Pct.dts` in the first assessment session. In the upper text area, 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 can be visualized by activating the *XML DTD Elements* tab. Note that you probably have 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 false negatives (i.e., appropriate XML tags are actually part of DTD). When



1/462; DIAsDEM Document ID: /home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject/inputCollection/volume100000.xml:0

<IfAppointmentOfOneManagingDirector_SolePowerToRepresent>Ist nur ein Geschäftsführer bestellt, so vertritt er die Gesellschaft einzeln.</IfAppointmentOfOneManagingDirector_SolePowerToRepresent>

Correct XML Tag: IfAppointmentOfOneManagingDirector_SolePowerToRepresent

DIAsDEM XML Tag: □ More General Than Correct XML Tag □ More Specific Than Correct XML Tag

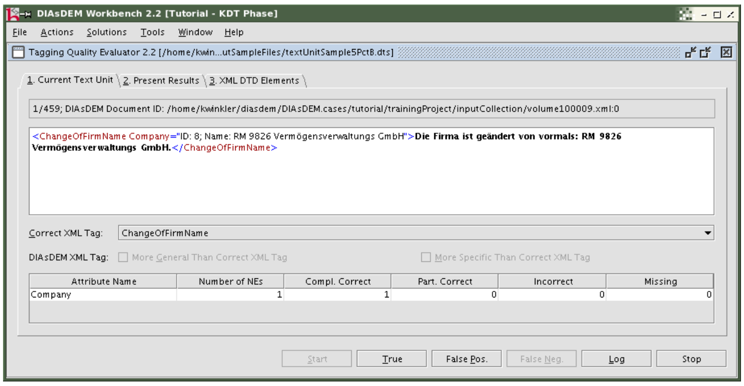
Attribute Name	Number of NEs	Compl. Correct	Part. Correct	Incorrect	Missing

Figure 3.26: *Tagging Quality Evaluator 2.2* Tool (1st Session)

assessing the quality of XML tag attributes and extracted named entities, respectively, according the concept-based XML DTD, the domain expert first counts the true number of named entities for each relevant named entity type. Subsequently, the number of completely correct, partially correct, incorrect, and missing named entities has to be determined for each relevant named entity type. Please assess three text units by evaluating the quality of XML attributes and subsequently clicking the appropriate buttons *True*, *False Pos.*, and *False Neg.*, respectively. Thereafter, click on *Stop* and open the file `${PROJECT_HOME}/outputSamplesFile/ evaluatedTextUnits.det`:

```
# TP,TN,FP,FN,Type,DiasdemXmlTag,CorrectXmlTag,DiasdemXmlTagIsMoreGeneral,
  DiasdemXmlTagIsMoreSpecific,NumberOfExistingAttributeNEs,
  NumberOfComplCorrectAttributeNEs,NumberOfPartCorrectAttributeNEs,
  NumberOfIncorrectAttributeNEs,NumberOfMissingAttributeNEs,TextUnit
1,0,0,0,"TP","IfAppointmentOfOneManagingDirector_SolePowerToRepresent",
  "IfAppointmentOfOneManagingDirector_SolePowerToRepresent",0,0,0,0,0,0,0,
  "/home/.../volume100000.xml:0 <IfAppointmentOfOneManagingDirector_
  SolePowerToRepresent>Ist nur ein Geschäftsführer bestellt, so vertritt er
  die Gesellschaft einzeln.</IfAppointmentOfOneManagingDirector_
  SolePowerToRepresent>"
1,0,0,0,"TP","IfAppointmentOfManyManagingDirectors_JointPowerToRepresent",
  "IfAppointmentOfManyManagingDirectors_JointPowerToRepresent",0,0,0,0,0,0,0,
  "/home/.../volume100002.xml:0 <IfAppointmentOfManyManagingDirectors_
  JointPowerToRepresent>Sind mehrere Geschäftsführer bestellt, so wird die
  Gesellschaft durch zwei Geschäftsführer oder durch einen Geschäftsführer in
  Gemeinschaft mit einem Prokuristen vertreten.</IfAppointmentOfManyManaging
  Directors_JointPowerToRepresent>"
```

The results of each assessment session are appended to `evaluatedTextUnits.det`. After completing the quality evaluation, this file can be renamed `evaluatedTextUnits.csv`



1/459; DIAsDEM Document ID: /home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject/inputCollection/volume100009.xml:0

```
<ChangeOfFirmName Company="ID: 8; Name: RM 9826 Vermögensverwaltungs GmbH">Die Firma ist geändert von vormals: RM 9826
Vermögensverwaltungs GmbH.</ChangeOfFirmName>
```

Correct XML Tag: ChangeOfFirmName

DIAsDEM XML Tag: □ More General Than Correct XML Tag □ More Specific Than Correct XML Tag

Attribute Name	Number of NEs	Compl. Correct	Part. Correct	Incorrect	Missing
Company	1	1	0	0	0

Figure 3.27: *Tagging Quality Evaluator 2.2* Tool (2nd Session)

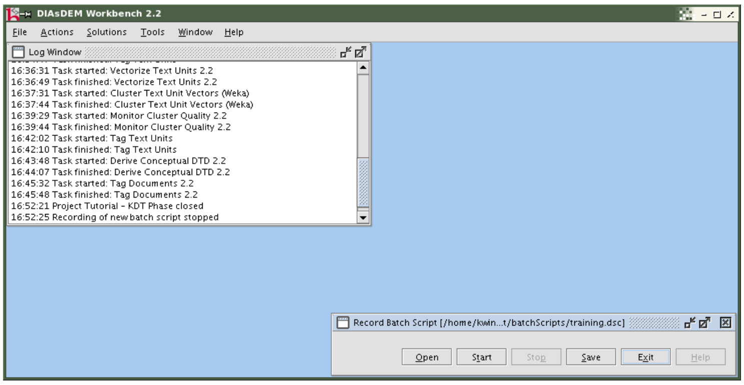
and imported into any spreadsheet application for detailed analysis. Note, the number of text units contained in the entire training collection is listed in *Existing Conceptual DTD File* as the property NUMBER_OF_TEXT_UNITS. Close the *Tagging Quality Evaluator 2.2* window to simulate the end of the current session. Thereafter, the second assessment session can be started by again selecting *Tools* → *Tagging Quality Evaluator 2.2*, clicking the *Start* button, and choosing the following five parameter files one after the other:

Parameter	Value
Existing Text Unit Sample File	`${PROJECT_HOME}/outputSamplesFile/textUnitSample5PctB.dts`
New or Existing File of Evaluated Text Units	`${PROJECT_HOME}/outputSamplesFile/evaluatedTextUnits.det`
Text Unit Sample File to be Created for Next Evaluation	`${PROJECT_HOME}/outputSamplesFile/textUnitSample5PctC.dts`
Existing Conceptual DTD File	`${PROJECT_HOME}/conceptualDtd.dpd`
New or Existing Log for Personal Notes	`${PROJECT_HOME}/outputSamplesFile/evaluationLog.txt`

Figure 3.27 illustrates the *Tagging Quality Evaluator 2.2* window at the beginning of the second assessment session. In the left pane, the results of previous sessions as contained in `${PROJECT_HOME}/outputSamplesFile/evaluatedTextUnits.det` are displayed as well. If you have time, you might assess the remaining 459 text units by clicking the appropriate buttons *True*, *False Pos.*, and *False Neg.*, respectively. At any time, activate the *Present Results* tab to have a look at the results of the markup quality assessment thus far.

3.4.4 Stopping the Batch Script Recorder

Before stopping the batch script recording, close the project by selecting *File* → *Close Project*. Thereafter, stop recording the tasks performed during the KDT phase by clicking the button *Stop* of *Batch Script Recorder*. Save the recorded batch script by clicking *Save* and choosing the file name `${PROJECT_HOME}/batchScripts/training.dsc`. Figure 3.28 depicts DIAsDEM Workbench after stopping the recording and saving the script. The knowledge discovery phase of the DIAsDEM framework is now finished. As explained in Section 3.5, the new batch script has to be edited before it can be executed to tag text archives in the application phase.



```
16:36:31 Task started: Vectorize Text Units 2.2
16:36:49 Task finished: Vectorize Text Units 2.2
16:37:31 Task started: Cluster Text Unit Vectors (Weka)
16:37:44 Task finished: Cluster Text Unit Vectors (Weka)
16:39:29 Task started: Monitor Cluster Quality 2.2
16:39:44 Task finished: Monitor Cluster Quality 2.2
16:42:02 Task started: Tag Text Units
16:42:10 Task finished: Tag Text Units
16:43:48 Task started: Derive Conceptual DTD 2.2
16:44:07 Task finished: Derive Conceptual DTD 2.2
16:45:32 Task started: Tag Documents 2.2
16:45:48 Task finished: Tag Documents 2.2
16:52:21 Project Tutorial - KDT Phase closed
16:52:25 Recording of new batch script stopped
```

Figure 3.28: DIAsDEM Workbench 2.2 after Saving the Batch Script

3.5 Summary of the Application Phase

Before this case is finished, 161 Commercial Register entries remain to be semantically tagged in the application phase of the DIAsDEM framework. The corresponding files (extension: `.application.txt`) are located in the directory `${SAMPLES_HOME}/de/case1`. The term application phase refers to the activity of iteratively applying previously created text unit clusterers to convert new text documents into semantically tagged XML documents. Output XML documents conform to the concept-based XML document type definition that was derived in the KDT phase. Note, texts to be tagged in the application phase must feature similar content as the training documents for obvious reasons. However, you might, for example, process a stratified sample of text documents in the KDT phase.

In contrast to the interactive training phase of the DIAsDEM framework, text documents are semantically annotated in an automated batch process, which does not require any human intervention at all. The application phase of the DIAsDEM framework differs from training text unit clusterers in the following steps:

- When vectorizing text units, an existing iteration-specific text unit descriptor weights file is applied to weight terms frequencies instead of creating a new one.
- When clustering text unit vectors, an existing iteration-specific text unit clusterer, which is often referred to as "score code", is applied to assign vectors to clusters.
- Monitoring the quality of text unit vector clusters is not necessary because there is not need to create and edit a new cluster label file in the application phase.
- When tagging text units, an existing iteration-specific cluster label file is applied to tag text units according to the cluster labels assigned in the KDT phase.
- Deriving a concept-based XML DTD is not necessary because output documents must conform to the concept-based XML DTD derived in the KDT phase.

The term application parameters encompasses certain parameter files that are created in the KDT phase for subsequent usage in the application phase of the DIAsDEM framework. For each KDT process iteration, three files constitute specific application parameters: *Collection Frequencies File* listing iteration-specific weights, *Text Unit Clusterer File* including an iteration-specific clustering model, and *Cluster Label File* containing mappings from cluster IDs onto iteration-specific cluster labels. Furthermore, *Conceptual DTD File* along with its three auxiliary files are application parameters as well. When applying text unit clusterers to a new collection, the remaining parameter settings and the sequence of tasks must exactly correspond to the training procedure.

3.5.1 Preparing the Application Phase

The application phase of this case study constitutes a new project, whose files should reside in a dedicated project directory. To avoid confusing training and application phase, the abbreviation `${APP_PROJECT_HOME}` corresponds to the directory `/home/kwinkler/diasdem/DIAsDEM.cases/tutorial/applicationProject` in the remainder of this case study. Create a new local directory `${APP_PROJECT_HOME}` on your machine and copy the entire directory template for application projects into this directory:

```
.cases/tutorial> pwd
/home/kwinkler/diasdem/DIAsDEM.cases/tutorial
.cases/tutorial> cp -R ../../DIAsDEM.workbench21/data/templates/applicationProject .
.cases/tutorial> ls
applicationProject trainingProject
```

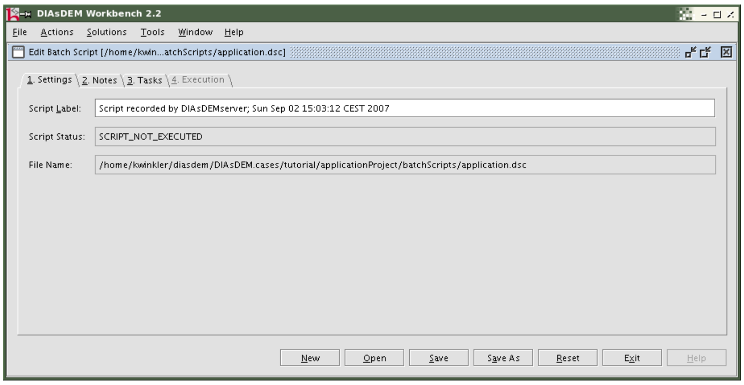
Thereafter, copy all application parameter files from the training project directory into the new directory: These files must include a *Collection Frequencies File* for each iteration, a *Text Unit Clusterer File* for each iteration, a *Cluster Label File* for each iteration, and the *Conceptual DTD File* along with its three auxiliary files. Furthermore, copy the previously recorded DIAsDEM batch script into the respective application subdirectory.

```
.cases/tutorial> pwd
/home/kwinkler/diasdem/DIAsDEM.cases/tutorial
.cases/tutorial> cp trainingProject/*.dcfq applicationProject/
.cases/tutorial> cp trainingProject/*.wskm applicationProject/
.cases/tutorial> cp trainingProject/*.dcl applicationProject/
.cases/tutorial> cp trainingProject/conceptualDtd* applicationProject/
.cases/tutorial> ls applicationProject/
1clusterer.wskm      batchScripts                      outputGateDocuments
1labels.dcl          conceptualDtd.dcd                 outputNeex21Files
1weights.dcfq        conceptualDtd.dcd.attributes      outputSampleFiles
2clusterer.wskm      conceptualDtd.dcd.elements        outputSqlScripts
2labels.dcl          conceptualDtd.dcd.xml             outputXmlDocuments
2weights.dcfq        conceptualDtdDocumentation.html README
applicationParameters inputCollection
DIAsDEM.cases/tutorial> cd trainingProject/batchScripts/
trainingProject/batchScripts> cp training.dsc application.dsc
trainingProject/batchScripts> ls
application.dsc README training.dsc
trainingProject/batchScripts> mv application.dsc ../../applicationProject/batchScripts/
```

3.5.2 Editing the Batch Script

DIAsDEM batch scripts are XML documents that conform to the XML document type definition listed in Subsection 4.2 on page 91. They can be modified using any text editor or preferrably using the dedicated editor (*Solutions → Batch Script Processing → Edit Batch Script*). Open the file ${APP_PROJECT_HOME}/batchScripts/application.dsc in your preferred text editor to quickly correct the project directory in this script. Replace all occurrences of the training directory ${PROJECT_HOME}, such as /home/kwinkler/diasdem/DIAsDEM.cases/tutorial/trainingProject, with the directory that now corresponds to ${APP_PROJECT_HOME}, such as /home/kwinkler/diasdem/DIAsDEM.cases/tutorial/applicationProject. Do not include the trailing slash in directory names to ensure the proper replacement of all 47 ${PROJECT_HOME} occurrences in the script.

Thereafter, open the DIAsDEM Batch Script Editor by selecting *Solutions → Batch Script Processing → Edit Batch Script*, clicking the *Open* button, and choosing the application script application.dsc located in the directory ${APP_PROJECT_HOME}/batchScripts. Figure 3.29 illustrates the initially shown *1. Settings* tab of the editor. After clicking on the *3. Tasks* tab, the sequence of 19 recorded tasks appears as depicted in Figure 3.30. Select task 11 ("Monitor Cluster Quality 2.2") by clicking once on the respective table row. Delete the respective task from the batch script by clicking the *Delete* button because monitoring cluster quality is not necessary in the application phase. Afterwards, task 14 ("Monitor Cluster Quality 2.2") must also be deleted. For the same reason, delete the task "Derive Conceptual DTD 2.2" as well.

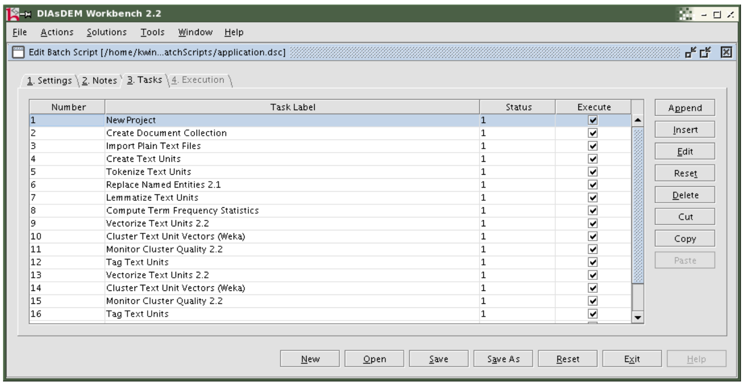


Script Label: Script recorded by DIAsDEMserver; Sun Sep 02 15:03:12 CEST 2007

Script Status: SCRIPT_NOT_EXECUTED

File Name: /home/kwinkler/diasdem/DIAsDEM.cases/tutorial/applicationProject/batchScripts/application.dsc

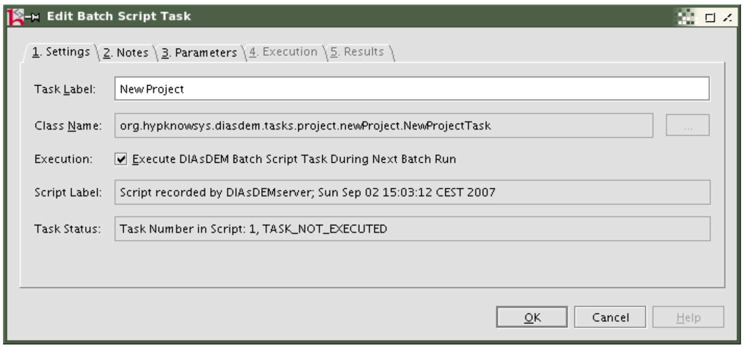
Figure 3.29: *1. Settings* Tab of *Edit Batch Script* Window



Number	Task Label	Status	Execute
1	New Project	1	☑
2	Create Document Collection	1	☑
3	Import Plain Text Files	1	☑
4	Create Text Units	1	☑
5	Tokenize Text Units	1	☑
6	Replace Named Entities 2.1	1	☑
7	Lemmatize Text Units	1	☑
8	Compute Term Frequency Statistics	1	☑
9	Vectorize Text Units 2.2	1	☑
10	Cluster Text Unit Vectors (Weka)	1	☑
11	Monitor Cluster Quality 2.2	1	☑
12	Tag Text Units	1	☑
13	Vectorize Text Units 2.2	1	☑
14	Cluster Text Unit Vectors (Weka)	1	☑
15	Monitor Cluster Quality 2.2	1	☑
16	Tag Text Units	1	☑

Figure 3.30: *3. Tasks* Tab of *Edit Batch Script* Window

The application script should now comprise the following sequence of 16 tasks: "New Project", "Create Document Collection", "Import Plain Text Files", "Create Text Units", "Tokenize Text Units", "Replace Named Entities 2.1", "Lemmatize Text Units", "Compute Term Frequency Statistics", "Vectorize Text Units 2.2", "Cluster Text Unit Vectors (Weka)", "Tag Text Units", "Vectorize Text Units 2.2", "Cluster Text Unit Vectors (Weka)", "Tag Text Units", "Tag Documents 2.2", and finally "Close Project". A few tasks have to be edited by hand to adjust their parameters settings. To proceed, select the row corresponding to task 1 ("New Project") and click on *Edit*. Figures 3.31 and 3.32 illustrate the appearing task dialog, which allows you to modify the parameter settings. For example, input the *Project Name* "Tutorial - Application Phase" and click on *OK* to commit the change.



Task Label: New Project

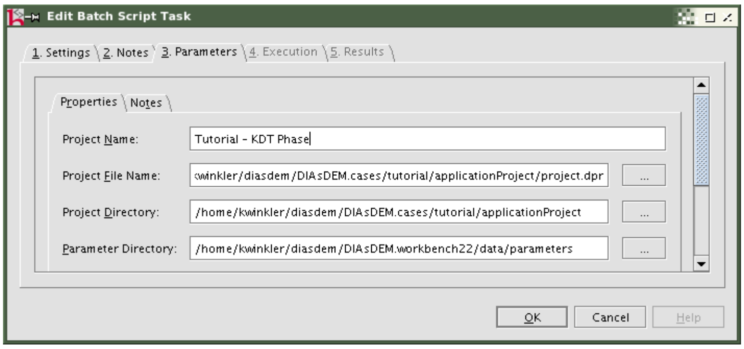
Class Name: org.hypknowsys.diasdem.tasks.project.newProject.NewProjectTask

Execution: ☑ Execute DIAsDEM Batch Script Task During Next Batch Run

Script Label: Script recorded by DIAsDEMserver; Sun Sep 02 15:03:12 CEST 2007

Task Status: Task Number in Script: 1, TASK_NOT_EXECUTED

Figure 3.31: *1. Settings* Tab of *Edit Batch Script Task* Dialog



Project Name: Tutorial - KDT Phase

Project File Name: <winkler/diasdem/DIAsDEM.cases/tutorial/applicationProject/project.dpr

Project Directory: /home/kwinkler/diasdem/DIAsDEM.cases/tutorial/applicationProject

Parameter Directory: /home/kwinkler/diasdem/DIAsDEM.workbench22/data/parameters

Figure 3.32: *3. Parameters* Tab of *Edit Batch Script Task* Dialog

In addition, the following five tasks have to edited as well: In the "Import Plain Text Files" task, modify the *File Name Extension* of files to be imported from ".training.txt" to ".application.txt". Ignore the appearing warning message stating that the specified *Collection File* does not exist. During script execution, this file is created in the preceding task "Create Document Collection". In the "Vectorize Text Units 2.2" task of both iterations, change *Collection Frequency* from "Inverse Collection Frequency: Create New File" to "Apply Existing Collection Frequencies File". In the "Cluster Text Unit Vectors (Weka)" task of both iterations, change *Clustering Mode* from "Clustering Phase (Create

New Clustering Model)" to "Application Phase (Apply Existing Clustering Model)". Finally, change the parameter *Conceptual DTD File* in the task "Tag Documents 2.2" from `${APP_PROJECT_HOME}/conceptualDtd.dpd` to `${PROJECT_HOME}/conceptualDtd.dpd`. After saving the script by clicking on *Save*, the modified batch script is now ready for execution, which is explained in the next section. Finally, click the *Exit* button to close the DIAsDEM batch script editor.

3.5.3 Executing the Batch Script

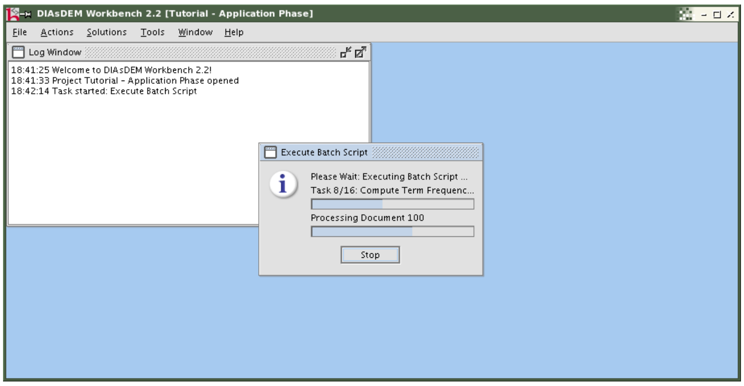
There are two options for executing batch scripts because DIAsDEM Workbench comprises both a command line and a GUI-based batch script processor. Firstly, the batch script `application.dsc` can be executed by the command line script processor as indicated in the installation notes:

```
/batchScripts> ../../../DIAsDEM.workbench22/bin/diasdembatch application.dsc verbose
*
* DIAsDEM Workbench 2.2 2.2.0.0 for Java 1.4.2 Released 27 May 2006
* Executing Batch Script: application.dsc ...
*
18:35:13 Task started: Execute Batch Script
18:35:13 Starting execution of task 1/16 (New Project)
18:35:13 Execution of task 1/16 (New Project) has terminated successfully
...
18:37:34 Execution of task 15/16 (Tag Documents 2.2) has terminated successfully
18:37:34 Starting execution of task 16/16 (Close Project)
18:37:34 Execution of task 16/16 (Close Project) has terminated successfully
18:37:34 Task successfully finished: Execute Batch Script
```

Secondly, the GUI-based script processor can be used to execute batch scripts. To work around a DIAsDEM Workbench 2.2 bug, open the new project by selecting *File → Open Project* and choosing the file `${APP_PROJECT_HOME}/project.dpr`. Subsequently, select the GUI-based script processor via *Solutions → Batch Script Processing → Execute Batch Script*. Open the batch script `application.dsc` located in the directory `${APP_PROJECT_HOME}/batchScripts` and click on *OK* to start the script execution. Figure 3.33 illustrates DIAsDEM Workbench while executing this script.

Semantically tagging 161 Commercial Register entries takes approx. two minutes in both cases. After script execution, altogether 161 semantically tagged XML documents are located in the directory `${APP_PROJECT_HOME}`. Using for example *Tools → Miscellaneous → XML Document Viewer*, open the output document `${APP_PROJECT_HOME}/outputXmlDocuments/part1/document1.xml`, which is partly depicted below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE CommercialRegisterEntry SYSTEM "CommercialRegisterEntry.dtd">
```



```
18:41:25 Welcome to DIAsDEM Workbench 2.2!
18:41:33 Project Tutorial – Application Phase opened
18:42:14 Task started: Execute Batch Script
```

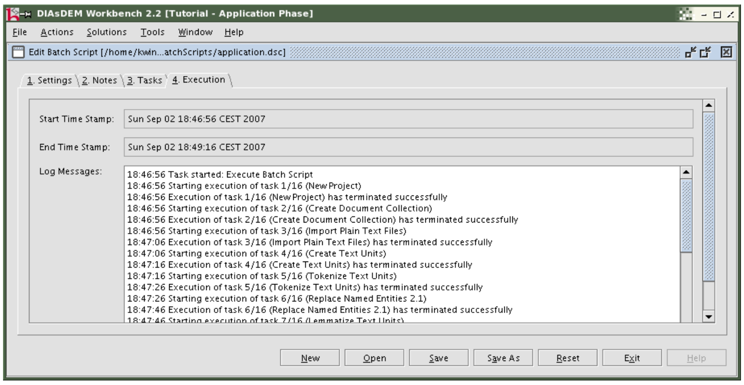
Please Wait: Executing Batch Script ...
Task 8/16: Compute Term Frequenc...
Processing Document 100

Figure 3.33: DIAsDEM Workbench 2.2 during GUI-Based Batch Script Execution

```
<CommercialRegisterEntry>
  <MetaData>
    <Name>DiasdemDocumentID</Name>
    <Content>/home/.../applicationProject/inputCollection/volume100000.xml:0</Content>
  </MetaData>
  <MetaData>
    <Name>SourceFile</Name>
    <Content>/home/.../data/samples/de/case1/file11075.application.txt</Content>
  </MetaData>
  <TaggedDocument>
    Dachdeckerarbeiten. <ShareCapital AmountOfMoney="25000 EUR">Stammkapital:
    25.000 EUR.</ShareCapital><LimitedLiabilityCompany>Gesellschaft mit
    beschränkter Haftung.</LimitedLiabilityCompany> ...
    <SolePowerToRepresentCanBeGranted>Einzelvertretungsbefugnis kann erteilt
    werden.</SolePowerToRepresentCanBeGranted><AppointmentOfManagingDirector
    Person="10; Mario Schmeling; null; null; null; 05.07.1967; null; Bamme; null;
    null [AND] 11; Thomas Weber; null; null; null; 16.06.1967; null; Rathenow;
    null; null">Mario Schmeling, 05.07.1967, Bamme; und Thomas Weber, 16.06.1967,
    Rathenow, sind zu Geschäftsführern bestellt.</AppointmentOfManagingDirector> ...
    <PublicationMediaOfCommercialRegisterEntries>Nicht eingetragen: Die
    Bekanntmachungen der Gesellschaft erfolgen im Bundesanzeiger.
    </PublicationMediaOfCommercialRegisterEntries>
  </TaggedDocument>
</CommercialRegisterEntry>
```

Furthermore, open the script `${APP_PROJECT_HOME}/batchScripts/application.dsc` in *Solutions* → *Batch Script Processing* → *Edit Batch Script*. As Figures 3.34 and 3.35 illustrate, the entire script and each performed task now contains additional information (e.g., log messages and time stamps) about the execution.

You have now reached the end of this introductory case study! We are looking forward to getting your feedback on the DIAsDEM framework and DIAsDEM Workbench.


DIAsDEM Workbench 2.2 [Tutorial - Application Phase]
File
Actions
Solutions
Tools
Window
Help
Edit Batch Script [/home/kwin...atchScripts/application.dsc]
1. Settings
2. Notes
3. Tasks
4. Execution
New
Open
Save
Save As
Reset
Exit
Help

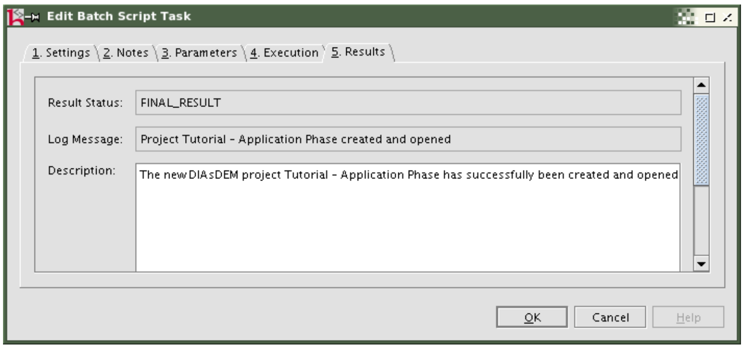
Start Time Stamp: Sun Sep 02 18:46:56 CEST 2007

End Time Stamp: Sun Sep 02 18:49:16 CEST 2007

Log Messages:

```
18:46:56 Task started: Execute Batch Script
18:46:56 Starting execution of task 1/16 (New Project)
18:46:56 Execution of task 1/16 (New Project) has terminated successfully
18:46:56 Starting execution of task 2/16 (Create Document Collection)
18:46:56 Execution of task 2/16 (Create Document Collection) has terminated successfully
18:46:56 Starting execution of task 3/16 (Import Plain Text Files)
18:47:06 Execution of task 3/16 (Import Plain Text Files) has terminated successfully
18:47:06 Starting execution of task 4/16 (Create Text Units)
18:47:16 Execution of task 4/16 (Create Text Units) has terminated successfully
18:47:16 Starting execution of task 5/16 (Tokenize Text Units)
18:47:26 Execution of task 5/16 (Tokenize Text Units) has terminated successfully
18:47:26 Starting execution of task 6/16 (Replace Named Entities 2.1)
18:47:46 Execution of task 6/16 (Replace Named Entities 2.1) has terminated successfully
18:47:46 Starting execution of task 7/16 (Lemmatize Text Units)
```

Figure 3.34: *4. Execution* Tab of *Edit Batch Script* Window



Result Status: FINAL_RESULT

Log Message: Project Tutorial - Application Phase created and opened

Description: The new DIAsDEM project Tutorial - Application Phase has successfully been created and opened

Figure 3.35: *5. Results* Tab of *Edit Batch Script* Dialog

3.6 Auxiliary Tasks

3.6.1 Removing Stopwords

DIAsDEM Workbench is capable of removing meaningless stopwords from processed text units. As explained in Section 1, the DIAsDEM framework proposes utilizing a controlled vocabulary (i.e., a domain-specific thesaurus) for dimension reduction. Thus, stopword removal can be skipped in this case study due to the existence of a domain-specific thesaurus. However, stopswords should be removed in case of establishing a new controlled vocabulary for a different domain. The text file `${PARAMETER_HOME}/removeStopwords/de/StopwordsDE.txt` contains a default German stopword list, which should be modified according to domain-specific needs.

Remove Stopwords: Summary

Task: *Actions → Prepare Data Set → Remove Stopwords*

Use Case: The user wants to remove meaningless stopwords from text units that are contained in the section `<ProcessedTextUnits>`.

Prerequisites: The default text units layer of each DIAsDEM document must contain the section `<ProcessedTextUnits>`. Text units should have been created and tokenized in the DIAsDEM collection.

Result: Elements `<ProcessedTextUnit>` do not contain terms listed in *Stopwords File*. Additionally, the project properties *Default Collection File* and *Default Stopwords File* are set and updated, respectively.

Remove Stopwords: Parameters

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

Stopwords File: Valid local file name of existing file that contains stopwords in the format described in Subsection 4.3.4 on page 101; file extension: `.txt`; default value: project property *Default Stopwords File*

3.6.2 Establishing an Initial Thesaurus

Before applying DIAsDEM Workbench to a new text collection, an application-specific thesaurus has to be created. This task establishes an initial thesaurus on the basis of term frequency statistics output by *Actions → Understand Domain → Compute Term Frequency Statistics*. Terms are inserted into the new thesaurus as descriptor terms if their collection-specific absolute frequency is greater than or equal to a minimum and less than or equal to a maximum threshold. However, the resulting initial thesaurus should be manually refined by removing, for example, 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.

Note, it is strongly recommended removing stopwords before establishing an initial thesaurus for a new application domain. If semantically less important stopwords are not removed, the resulting thesaurus contains them as well. After establishing an initial thesaurus, it should be refined by hand using *Tools → Thesaurus Editor 2.2*. To quickly remove less important terms, thesaurus files might also be edited using any common text editor: Simply delete all lines that correspond to terms to be removed.

Establish Initial Thesaurus: Summary

Task: *Actions → Understand Domain → Establish Initial Thesaurus*

Use Case: The user wants to establish an application-specific initial thesaurus that is based on collection-specific term frequencies. Subsequently, this thesaurus should be refined using *Tools → Thesaurus Editor 2.2.*

Prerequisites: Using *Actions → Understand Domain → Compute Term Frequency Statistics*, a collection-specific term frequency file must have been created. Text units should have been created, tokenized, and lemmatized in the DIAsDEM collection and named entities should have been replaced with placeholders in all text units.

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

Establish Initial Thesaurus: Parameters

TF Statistics File: Valid local file name of existing file; file extension: `.dws`; default value: project property *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. Term Frequency: Non-negative integer; minimum absolute term frequency of descriptors in *Initial Thesaurus File*

Max. Term Frequency: Non-negative integer; maximum absolute term frequency of descriptors in *Initial Thesaurus File*

4 Technical Specification

4.1 DIAsDEM Documents

The default implementation of DIAsDEM Workbench 2.2 stores DIAsDEM documents as part of so-called DIAsDEM volumes. The latter are XML files that conform to the following XML document type definition `DefaultDIAsDEMvolume.dtd`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!ELEMENT DefaultDIAsDEMvolume (DefaultDIAsDEMdocument*)>
<!ATTLIST DefaultDIAsDEMvolume NumberOfDocuments CDATA #IMPLIED>

<!ELEMENT DefaultDIAsDEMdocument (MetaData*, OriginalText, TextUnitsLayer*)>
<!ATTLIST DefaultDIAsDEMdocument NumberOfTextUnitsLayers CDATA #IMPLIED>

<!ELEMENT MetaData (Name, Content)>
<!ELEMENT Name (#PCDATA)> <!ELEMENT Content (#PCDATA)>
<!ELEMENT OriginalText (#PCDATA)>

<!ELEMENT TextUnitsLayer (MetaData*, OriginalTextUnits, ProcessedTextUnits?,
  RollbackTextUnits*, NamedEntities? )>
<!ATTLIST TextUnitsLayer TextUnitsLayerID CDATA #IMPLIED
  TextUnitsDescription CDATA #IMPLIED>

<!ELEMENT OriginalTextUnits (OriginalTextUnit+)>
<!ELEMENT OriginalTextUnit (#PCDATA)>
<!ATTLIST OriginalTextUnit TextUnitID CDATA #IMPLIED BeginIndex CDATA #IMPLIED
  EndIndex CDATA #IMPLIED>
<!ELEMENT ProcessedTextUnits (ProcessedTextUnit+)>
<!ELEMENT ProcessedTextUnit (#PCDATA | NeRef)*>
<!ATTLIST ProcessedTextUnit TextUnitID CDATA #IMPLIED Iteration CDATA #IMPLIED
  ClusterID CDATA #IMPLIED ClusterLabel CDATA #IMPLIED>
<!ELEMENT RollbackTextUnits (ProcessedTextUnit+)>
<!ELEMENT NamedEntities (NamedEntity+)>
<!ELEMENT NamedEntity (#PCDATA)>
<!ATTLIST NamedEntity NeID CDATA #IMPLIED NeType CDATA #IMPLIED>
<!ATTLIST RollbackTextUnits RollbackID CDATA #IMPLIED>
<!ELEMENT RollbackTextUnit (#PCDATA | NeRef)*>

<!ELEMENT NeRef EMPTY>
<!ATTLIST NeRef NeID CDATA #IMPLIED>
```

4.2 DIAsDEM Batch Scripts

DIAsDEM Workbench is capable of executing batch scripts (i.e., XML documents) that conform to the following XML document type definition `DiasdemBatchScript.dtd`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!ELEMENT DIAsDEMscript (Label, DIAsDEMscriptTask*, Notes?, Log?, Status?,
  StartTimeStamp?, EndTimeStamp?)>

<!ELEMENT DIAsDEMscriptTask (Label, Parameter, Result?, Notes?, Log?, Status?,
  StartTimeStamp?, EndTimeStamp?)>
<!ATTLIST DIAsDEMscriptTask ClassName CDATA #IMPLIED Execute CDATA #IMPLIED>

<!ELEMENT Label (#PCDATA)>
<!ELEMENT ClassName (#PCDATA)>
<!ELEMENT Parameter (ParameterAttributes)>
<!ATTLIST Parameter ClassName CDATA #IMPLIED>
<!ELEMENT Result (ResultAttributes)>
<!ATTLIST Result ClassName CDATA #IMPLIED>
<!ELEMENT Notes (#PCDATA)>
<!ELEMENT Log (#PCDATA)>
<!ELEMENT Status (#PCDATA)>
<!ELEMENT StartTimeStamp (#PCDATA)>
<!ELEMENT EndTimeStamp (#PCDATA)>

<!ELEMENT ParameterAttributes (ParameterAttribute*)>
<!ELEMENT ParameterAttribute (AttributeName, AttributeValue)>
<!ELEMENT ResultAttributes (ResultAttribute*)>
<!ELEMENT ResultAttribute (AttributeName, AttributeValue)>

<!ELEMENT AttributeName (#PCDATA)>
<!ELEMENT AttributeValue (#PCDATA)>
```

4.3 Text Pre-Processing

4.3.1 Create Text Units

Abbreviations File: Valid local file name of existing text file, which contains known abbreviations in the following format: Each line of *Abbreviations File* contains exactly one abbreviation whose capitalization is relevant. However, this task only matches abbreviations if they either occur at the beginning of the text or if they follow one of certain special characters (i.e., the blank space and `(),;:/-’"`). Comment lines starting with “#” are ignored and can hence be used to structure the file. Example:

```
# Format: One case-sensitive, single- or multi-token abbreviation per line
Ph.D.
SAT.1
E.ON
a.d.
a. d.
a.D.
```

Full Stop Regex File: Valid local file name of existing text file, which 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 matched full stops with asterisks. Therefore, the replacement string usually includes references, such as `$1`, to captured subsequences, such as `(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`. Before these regular expressions are matched against the text, full stops in abbreviations listed in *Abbreviations File* have been replaced by asterisks. Comment lines starting with "#" are ignored. Example:

```
# full stops in 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
# full stops in abbreviations not preceded by a letter
(str|Str|pl|Pl)\.([\ ]*\d*)  $1\*$2
(ges|Ges)\.([\ ]*mbH) $1\*$2
# full stops in generic abbreviations such as titles (e.g., 'Dipl.-Kfm.')
(Dipl)\*([\ ]*|[\ ]*\-[\ ]*)([a-z~~A-Z~~+)\.   $1\*$2$3\*
(Art)\.\ (\d)  $1\*\ $2
# full stops in numbers (longest numbers have to matched first)
(\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`.

4.3.2 Tokenize Text Units

Tokenize Regex File: Valid local file name of existing text file, which contains regular expressions in the following format: Each line of *Tokenize Regex File* contains a regular expression matching characters to be separated from letters, exactly one tab stop, and thereafter a corresponding replacement string that separates tokens. Therefore, the replacement string usually includes 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 "#" are ignored. Example:

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

Normalize Regex File: Valid local file name of existing text file, which 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})\.[\ ]*([0-9]{1})\.[\ ]*([\d]{2,4}) $1.0$2.$3
# numbers
(\s[0-9]{1,3})\.([0-9]{3,3})\.([0-9]{3,3}\s) $1$2$3
(\s[0-9]{1,3})\.([0-9]{3,3}\s) $1$2
# amounts of money
([\-]?\s*{0,1}\d{1,}[,.\d]{1,})\s(DM|DEM|Deutsche\s+Mark|D\s+\-\s+Mark) $1\ DEM
([\-]?\s*{0,1}\d{1,}[,.\d]{1,})\s(Euro|EUR[O]?|Euros) $1\ EUR
```

Multi-Token Words File: Valid local file name of existing text file, which 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 "#" are ignored. Example:

```
# Format: One case-sensitive multi-token term per line
Gesellschaft mit beschränkter Haftung
mit beschränkter Haftung
Offene Handelsgesellschaft
offene Handelsgesellschaft
```

Token Replacement File: Valid local file name of existing text file, which contains tokens that sould be replaced by other tokens. For example, composite nouns (e.g., Gewinnanstieg) could be split (e.g., Gewinn Anstieg) or English clitics (e.g., wont and ’ ll) can be expanded (e.g., will not and will). Each line of *Token Replacement File* contains single- or multi-token terms to be searched for and the corresponding replacement tokens. Search and replacement tokens are separated by exactly one tab stop. Comment lines starting with "#" are ignored. Example:

```
# Format: single- or multi-token to search<TAB> single- or multi-token to replace
Gewinnanstieg Gewinn Anstieg
Ge - winnanstieg Gewinn Anstieg
Ge - winn Gewinn
```

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

4.3.3 Replace Named Entities 2.1

Regex NE File: Valid local file name of existing file, which contains regular expressions for instantiating basic named entities (i.e., dates, amounts of money, URLs, and e-mail addresses). Each line contains a java.util.regex.Pattern regular expression that matches sequences of tokens as well as the corresponding name of the basic named entity separated by exactly one tab stop. The following basic named entities could be instantiated using regular expressions: "number", "date", "time", "amount_of_money", "paragraph", "email", "url", "organization_id", "document_ id", "court", "postal_code", "reference_number", "percentage", "newspaper", "wkn" (i.e., German securities identification number), "isin" (i.e., international securities identification number), as well as "stock_ exchange", "number_of_shares", and "amount_of_ money_per_share". Comment lines starting with "#" are ignored. Example:

```
# Format: searchRegex<TAB>namedEntityType
# normalized amounts of money
\d{1,}[,\.\d]{1,}\s(DEM|EUR|ATS)    amount_of_money
# normalized dates
\d{1,2}\.\d{1,2}\.\d{2,4}    date
```

Organization Indicators File: Valid local file name of existing file, which contains terms and term groups which frequently precede names of organizations. Each line contains one indicator term (group), which are processed case-sensitively. Note, groups of indicator terms, such as "Gesellschafterin:", must be entered tokenized and in reverse order (e.g., ": Gesellschafterin") because NEEX 2.1 employs a backward search algorithm. All organization indicators containing full stops must be listed in *Abbreviations File* to ensure correct sentence splits. Comment lines starting with "#" are ignored. Example:

```
# Format: One case-sensitive, indicator term or reversed term group per line
: Gesellschafterin
Gesellschafterin
: Gesellschafter
Gesellschafter
Mitgesellschafter
```

Organization Suffixes File: Valid local file name of existing file, which contains a list of frequent organization suffixes in the following format: Each line contains exactly one suffix whose capitalization is relevant. NEEX 2.1 can process both single- (e.g., "KG") and multi-token suffixes containing, for example, blank spaces (e.g., "GmbH & Co. KG"). Note, the term "mit_beschränkter_Haftung" is a multi-token term whose real counterpart "mit beschränkter Haftung" is listed in *Multi-Token Words File* All suffixes containing full stops, such as "e.Kfr.", must be listed in *Abbreviations File* to ensure correct sentence splits. Comment lines starting with "#" are ignored. Example:

```
# Format: One case-sensitive, single- or tokenized multi-token
# organization suffix per line
GmbH & Co. KG
KG
Gesellschaft_mit_beschränkter_Haftung
mit_beschränkter_Haftung
AG
```

Organization Affixes File: Valid local file name of existing file, which contains a list of terms that frequently follow organization suffixes, such as "GmbH" or "AG", in the following format: Each line contains exactly one affix whose capitalization is irrelevant. NEEX 2.1 can process both single- (e.g., "Import") and multi-token organization affixes containing, for example, blank spaces (e.g., "Import und Export"). Organization affixes containing special characters, such as "(Deutschland)" or "Wach- und Werkschutz", must be entered in their tokenized form: "(Deutschland)" or "Wach - und Werkschutz". To that end, *Actions → Miscellaneous → Tokenize Parameter Text File* might be employed. Comment lines starting with "#" are ignored. Example:

```
# Format: One case-sensitive, single- or tokenized multi-token affix per line
Import
Export
Import und Export
( Deutschland )
Wach - und Werkschutz
```

Organizations File: Valid local file name of existing file, which contains complete names of large organizations. NEEX 2.1 can process both single- (e.g., "Adidas") and multi-token organizations containing, for example, blank spaces (e.g., "Adidas - Salomon"). They are processed case-sensitively. All organizations containing full stops must be listed in *Abbreviations File* to ensure correct sentence splits. Organizations containing special characters, such as "Adidas-Salomon" or "E.ON", must be entered in their tokenized form: "Adidas - Salomon" or "E.ON". Note, "E.ON" is a abbreviation listed in *Abbreviations File.* Include organizations in this file if their occurrences without known organization suffixes have to be extracted or if they include terms listed in the file containing organization indicators.

```
# Format: One case-sensitive, single- or tokenized multi-token organization per line
Adidas
Adidas - Salomon
Allianz
Altana
BASF
```

Place Indicators File: Valid local file name of existing file, which contains terms and term groups which frequently precede places to be extracted as named entities. Each line contains one indicator term (group), which are not processed case-sensitively. Note, groups of indicator terms, such as “mit Niederlassung in”, must be entered tokenized and in reverse order (e.g., “in Niederlassung in”) because NEEX 2.1 employs a backward search algorithm. All place indicators containing full stops must be listed in *Abbreviations File* to ensure correct sentence splits. Comment lines starting with “#” are ignored. Example:

```
# Format: One case-sensitive, indicator term or reversed term group per line
,
:
in
und
Sitz
```

Places File: Valid local file name of existing file, which contains a list of frequently occurring places, which are not processed case-sensitively. NEEX 2.1 can process both single- (e.g., “Berlin”) and multi-token places containing, for example, blank spaces (e.g., “Baden Baden”). Places containing special characters, such as “Frankfurt am Main”, “Frankfurt (Main)”, “Frankfurt/Oder”, or “Halle/Westf.” must be entered in their tokenized form: “Frankfurt am Main”, “Frankfurt (Main)”, “Frankfurt / Oder”, or “Halle / Westf.” as “Westf.” is contained in the list of known abbreviations. To that end, *Actions → Miscellaneous → Tokenize Parameter Text File.* Place affixes such as names of rivers, districts or countries should be entered separately in *Place Affixes File.* Comment lines starting with “#” are ignored. Example:

```
# Format: One case-sensitive, single- or tokenized multi-token organization per line
Berlin
Hamburg
Frankfurt am Main
Frankfurt / Oder
Halle / Westf.
```

Place Affixes File: Valid local file name of existing file, which contains frequently occurring place affixes, such as names of rivers and districts. NEEX 2.1 can process both single- (e.g., "Main") and multi-token place affixes containing, for example, blank spaces (e.g., "/ Main"). Place affixes are processed case-sensitively. Place affixes containing special characters, such as "/Main" or "(Main)", must be entered in their tokenized form: "/ Main" or "(Main)". To that end, *Actions → Miscellaneous → Tokenize Parameter Text File* might be used. Each place affix can either be a weak or a strong place affix, whereas places cannot end with a weak affix. Each line contains the affix type (i.e., either "weak_place_affix" or "strong_place_affix") and the affix itself separated by exactly one tab stop. Comment lines starting with "#" are ignored. Example:

```
# Format: placeAffixType<TAB>single- or tokenized multi-token place affix
weak_place_affix      an der
weak_place_affix      a. d.
weak_place_affix      im
strong_place_affix    Main
strong_place_affix    ( Main )
strong_place_affix    / Main
strong_place_affix    / M.
```

Person Name Indicators File: Valid local file name of existing file, which contains terms and term groups which frequently precede person names. Person name indicators are processed case-sensitively. Groups of indicator terms, such as "Gesellschafter:", must be entered tokenized and in reverse order (e.g., ": Gesellschafter") because NEEX 2.1 employs a backward search algorithm. All person name indicators containing full stops must be listed in *Abbreviations File* to ensure correct sentence splits. Each line contains the type literal as well as the indicator term (group) separated by exactly one tab stop. Valid person name indicator types are "weak_pos_person_indicator", "strong_pos_person_indicator", and "strong_neg_person_indicator". Note, person name indicators of type "weak_pos_person_indicator" are not used in NEEX 2.1. The occurrence of negative indicators is checked before and after any person name candidate. Hence they should comprise of one token only. Comment lines starting with "#" are ignored. Example:

```
# Format: placeAffixType<TAB>single- or tokenized, reversed multi-token indicator term
strong_pos_person_indicator  Herr
strong_pos_person_indicator  Mr.
strong_pos_person_indicator  geb.
strong_neg_person_indicator  Firma
strong_neg_person_indicator  Straße
strong_neg_person_indicator  Flughafen
```

Titles File: Valid local file name of existing file, which contains frequent academic and professional titles. NEEX 2.1 can process single- (e.g., “Prof.”) and multi-token titles containing, for example, blank spaces (e.g., “Prof. Dr.”). Titles are processed case-sensitively. Title containing special characters, such as “Prof. Dr.”, “Dipl.-Ingenieurin” or “Dipl.-Kfm. (FH)” must be entered in their tokenized form: “Prof. Dr.”, “Dipl.-Ingenieurin”, or “Dipl.-Kfm. (FH)”. To that end, *Actions → Miscellaneous → Tokenize Parameter Text File* might be employed. All titles containing full stops must either be listed in *Abbreviations File* or be matched by a regular expression in *Full Stop Regex File* (e.g., “Dipl.-Ingenieurin”). Comment lines starting with “#” are ignored. Example:

```
# Format: One case-sensitive, single- or tokenized multi-token title per line
Prof. Dr.
Dipl.-Ingenieurin
Dipl.-Kfm. ( FH )
Prof.
Dr.
```

Forenames File: Valid local file name of existing file, which contains a list of forenames in the following format: Each line contains exactly one forename whose capitalization is relevant. NEEX 2.1 can process both single- (e.g., “Stanka”) and multi-token forenames containing, for example, blank spaces (e.g., “Stanka Cevdet”). Do not include multi-token forenames consisting of multiple forenames, such as “Hans-Joachim” or “Hans Joachim”, because they are extracted automatically. However, forenames containing special characters, such as “Hans-Joachim”, must be entered in their tokenized form: “Hans - Joachim”. To that end, *Actions → Miscellaneous → Tokenize Parameter Text File* might be employed. Comment lines starting with “#” are ignored. Example:

```
# Format: One case-sensitive, single- or tokenized multi-token forename per line
Stanka
Cevdet
Wolfgang
Vid
Joaquin
```

Middle Initials File: Valid local file name of existing file, which contains a list of middle initials in the following format: Each line contains exactly one middle initial whose capitalization is relevant. NEEX 2.1 can process both single- (e.g., ’A.’ or ’von’) and multi-token middle initials containing for example blank spaces (e.g., ’de la’). Comment lines starting with “#” are ignored. Example:

```
# Format: One case-sensitive, single- or tokenized multi-token middle initial per line
de la
A.
von
De
de
```

Surnames File: Valid local file name of existing file, which contains frequent surnames. NEEX 2.1 can process both single- (e.g., “Schöppe”) and multi-token surnames containing, for example, blank spaces (e.g., “Schöppe Rocher”). Surnames are processed case-sensitively. Do not include multi-token surnames consisting of multiple surnames, such as “Schöppe-Rocher” or “Schöppe Rocher”, because they are extracted automatically. However, surnames containing special characters, such as “Schöppe-Rocher”, must be entered in their tokenized form: “Schöppe - Rocher”. To that end, *Actions → Miscellaneous → Tokenize Parameter Text File* might be employed. Comment lines starting with “#” are ignored. Example:

```
# Format: One case-sensitive, single- or tokenized multi-token surname per line
Meier
Müller
Schulze
Schmidt
Schmitt
```

Surname Suffixes File: Valid local file name of existing file, which contains frequently occurring suffixes of surnames. Capitalized tokens following a forename or an academic title are assumed to be a surname if they end with a suffix listed in this file. Each line contains exactly one surname suffix whose capitalization is relevant. Comment lines starting with “#” are ignored. Example:

```
# Format: One case-sensitive, single-token surname suffix per line
aci
ack
ad
wsky
yer
```

Name Affixes File: Valid local file name of existing file, which contains frequently occurring name affixes. NEEX 2.1 can process both single- (e.g., “jun.”) and multi-token name affixes containing, for example, blank spaces (e.g., “, jun.”). Each line contains exactly one name affix whose capitalization is relevant. Name affixes containing special characters,such as “,jun.” or “Ph.D.”, must be entered in their tokenized form: “, jun.” or “Ph.D.” as both “jun.” and “Ph.D.” are contained in the list of known German abbreviations. To that end, *Actions → Miscellaneous → Tokenize Parameter Text File* might be employed. Comment lines starting with “#” are ignored. Example:

```
# Format: One case-sensitive, single- or tokenized multi-token name affix per line
Ph.D.
, Ph.D.
sen.
, sen.
jun.
```

Professions File: Valid local file name of existing file, which contains frequently occurring professions. NEEX 2.1 can process both single- (e.g., “Angestellter”) and multi-token terms containing, for example, blank spaces (e.g., “Kaufmännischer Angestellter”). Professions processed case-sensitively. Professions containing special characters, such as “Dipl.-Kaufmann” or “Kfz.-Schlosser”, must be entered in their tokenized form: “Dipl. - Kaufmann” or “Kfz. - Schlosser”. To that end, *Actions → Miscellaneous → Tokenize Parameter Text File.* All professions containing full stops must either be listed in *Abbreviations File* or be matched by a regular expression in *Full Stop Regex File.* For obvious reasons, do not include text unit descriptors, such as the term “Geschäftsführer” in this file. Comment lines starting with “#” are ignored. Example:

```
# Format: One case-sensitive, single- or tokenized multi-token profession per line
Angestellter
Angestellte
Dipl. - Kaufmann
Dipl. - Kauffrau
```

Composite NE File: Valid local file name of existing file, which contains rules for instantiating composite named entities (i.e., persons, companies, and company relocations) from previously identified basic named entities, such as person names, places, or dates. Each line contains one DIAsDEM-specific rule that matches sequences of tokens and basic named entities, as well as the corresponding composite named entity constructor separated by exactly one tab stop.

The DIAsDEM-specific rule is a simple regular expression that must be matched by a tokenized text unit to instantiate a composite named entity, such as “person” or “company”. This expression can include case-sensitive tokens (e.g., “mit”, “Sitz”) and generic placeholders for basic named entities (e.g., “<<organization>>”, “<<person_name>>”), as defined in `org.hypknowsys.diasdem.core.neex.NamedEntity`. The corresponding named entity constructor instantiates a composite NE of type “person”, "company", or "company_relocation". Each constructor references terms and generic placeholders of the corresponding regular expression, which are attribute values of the new composite named entity. Note, tokens in both expressions must be separated from each others by blank spaces because named entities are identified in tokenized text units. Comment lines starting with "#" are ignored.

```
# Format: DIAsDEM-specific rule<TAB>DIAsDEM-specific composite NE constructor
# Note, there are three DIAsDEM-specific composite NE constructors available:
# company( Name , Place , Street, DistrictCourt , CommercialRegisterID )
# person( Name , Surname , Forename , Title , MiddleInitial , DoB , MothersName ,
#   Place , Street , Occupation)
# company_relocation( Name , OriginPlace , OriginStreet , OriginDistrictCourt ,
#   OriginCommercialRegisterID , DestinationPlace , DestinationStreet ,
#   DestinationDistrictCourt , DestinationCommercialRegisterID )
# date_period( PeriodBeginDate , PeriodEndDate )
# amount_of_money_range( MinimumAmountOfMoney , MaximumAmountOfMoney )
# percentage_range( MinimumPercentage , MaximumPercentage )
# equity_stake( CompanyName , NumberOfShares , PercentageOfShares )
# key_figure( Name , Value )
# unit_of_company( NameOfUnit , PlaceOfUnitHeadquarter , StreetOfUnitHeadquarter ,
#   DistrictCourtOfUnit , CommercialRegisterIDOfUnit , NameOfParent ,
#   PlaceOfParentHeadquarter , StreetOfParentHeadquarter , DistrictCourtOfParent ,
#   CommercialRegisterIDOfParent )
<<organization>>     company( 0 , null , null , null )
<<organization>> <<place>>   company( 0 " " 1 , 1 , null , null )
<<organization>> in <<place>> ( <<organization_id>> ) company( 0 , 2 , null , 4 )
<<person_name>> person( 0 , null , null , null , null , null , null , null , null )
```

The language-specific subdirectories of `${PARAMETER_HOME}/replaceNamedEntities` contain defaults for all parameter files described above. However, NEEX 2.1 parameter files are provided in the subdirectory `neex21` only.

4.3.4 Remove Stopwords

Stopword File: Valid local file name of existing text file, which contains meaningless stopwords. DIAsDEM Workbench 2.2 can only process single-token terms (e.g., "und") that do not contain blank spaces. Stopwords are not processed case-sensitively. Hence, the stopword "aber" also matches the term "ABER". Each line contains exactly one stopword. Comment lines starting with "#" are ignored. Example:

```
# Format: One case-sensitive, single-token stopword per line
ab
abend
aber
acht
alle
```

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

4.3.5 Lemmatize Text Units

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 are replaced by the task. This file is created by DIAsDEM Workbench and includes text to be POS-tagged by TreeTagger. Example:

```
<Document_/home/.../tutorial/trainingProject/inputCollection/volume100000.xml:0>
<ProcessedTextUnit_0>
Die Heizungs - und Sanitärinstallation , Gastechnik und Gassicherheitstechnik ...
</ProcessedTextUnit_0>
<ProcessedTextUnit_1>
Weiter ist Gegenstand die Konzeption , Montage , Instandsetzung und Instandhaltung ...
</ProcessedTextUnit_1>
<ProcessedTextUnit_2>
Stammkapital : <<0>> .
</ProcessedTextUnit_2>
<ProcessedTextUnit_3>
Gesellschaft_mit_beschränkter_Haftung .
</ProcessedTextUnit_3>
<ProcessedTextUnit_4>
Der Gesellschaftsvertrag ist am <<1>> abgeschlossen und am <<2>> abgeändert .
</ProcessedTextUnit_4> ...
<ProcessedTextUnit_10>
Nicht eingetragen : Die Bekanntmachungen der Gesellschaft erfolgen im Bundesanzeiger .
</ProcessedTextUnit_10>
</Document_/home/.../tutorial/trainingProject/inputCollection/volume100000.xml:0> ...
```

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 is extended by this task. This file is created by TreeTagger and includes the results of POS-tagging for subsequent parsing by DIAsDEM Workbench. Example:

```
<Document_/home/.../tutorial/trainingProject/inputCollection/volume100000.xml:0>
<ProcessedTextUnit_0>
Die     ART     d
Heizungs        NN      <unknown>
-       $(      -
und     KON     und
Sanitärinstallation     NN      Sanitärinstallation
...
```

```
</ProcessedTextUnit_0>
<ProcessedTextUnit_1>
Weiter  ADV     weiter
ist     VAFIN   sein
Gegenstand      NN      Gegenstand
die     ART     d
...
```

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: 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 "#" are ignored. Example:

```
# Format: term<TAB>lemmaFormOfTerm
Datenverarbeitungssysteme    Datenverarbeitungssystem
Arbeitsgemeinschaften Arbeitsgemeinschaft
Formaten        Format
Deetz   Deetz
Fanartikel      Fanartikel
Biographien     Biographie
```

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 is extended by the task. 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:

```
# Format: unknownTerm<TAB>correspondingTokenizedTextUnit
lit.    1. Der An - und Verkauf von Immobilien sowie die Beteiligung an ...
Art.    Der Gesellschaftsvertrag ist am <<1>> abgeschlossen und am <<2>> ...
Dip.    Dip. - <<26>> und Dr. jur. <<27>> , sind zu Geschäftsführern bestellt .
Dipl.-Kaufm.  Dipl.-Kaufm. <<47>> , ist zum Geschäftsführern bestellt .
```

4.4 Iterative Clustering

4.4.1 Vectorize Text Units 2.2

Vector File Format: DIAsDEM Workbench supports the export of four file formats: comma separated values (CSV files) and fixed width values (TXT files), as well as regular and sparse ARFF files in the Weka-specific format described in [WF05]. See below an example of a text unit vector file in comma separated values format:

```
DocumentType,Document,TextUnit,D1_Aktie,D2_Gesellschafter,D3_Inhaber,...,D73_Anspruch
"null","/home/.../volume100000.xml: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","/home/.../volume100984.xml:0",7,0,0,0,0,0,0,0,0.879,0,1.407,0,0,0,0,0,0,0,0,0,...
"null","/home/.../volume100984.xml:0",8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...
"null","/home/.../volume100984.xml:0",9,0,0,0,0,0,0,0,0,0,2.500,0,2.539,0,0,0,0,0,...
```

In the current version of DIAsDEM Workbench, “DocumentType” is always “null” due to legacy reasons. The attribute “Document” contains the DIAsDEM document ID. 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 respective *Collection File*. The following metadata file summarize information about attributes of the above CSV file:

```
DocumentType
Document
TextUnit
D1_Aktie = Aktie; Descriptor Frequency = 37; Descriptor Weight = 5.521
...
D73_Anspruch = Anspruch; Descriptor Frequency = 9; Descriptor Weight = 6.935
```

As of DIAsDEM Workbench 2.2, text unit vectors should not be exported as TXT files with fixed width values. The task *Vectorize Text Units* cannot properly process DIAsDEM documents whose IDs comprise more than 25 characters. However, below is an example of a text unit vector file in fixed width values format:

```
DocumentType␣␣␣␣␣␣␣␣Document␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣TextUnit␣␣D1_Aktie␣␣␣␣␣␣␣␣␣␣␣␣D2_Gese...
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/volume100000.xml:0␣␣␣␣␣␣0␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣...
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/volume100984.xml:0␣␣␣␣␣␣7␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣...
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/volume100984.xml:0␣␣␣␣␣␣8␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣...
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/volume100984.xml:0␣␣␣␣␣␣9␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣0␣␣␣␣␣␣...
```

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

```
1-20   DocumentType
21-45  Document
46-55  TextUnit
56-75  D1_Aktie = Aktie; Descriptor Frequency = 37; Descriptor Weight = 5.521
...
```

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

```
@relation 'DIAsDEM'
@attribute DocumentType string
@attribute Document string
@attribute TextUnit string
@attribute D1_Aktie real ...
@attribute D73_Anspruch real
@data
null,/home/.../volume100000.xml:0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...
null,/home/.../volume100984.xml:0,7,0,0,0,0,0,0,0,0.879,0,1.407,0,0,0,0,0,0,0,0,0,...
null,/home/.../volume100984.xml:0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...
null,/home/.../volume100984.xml:0,9,0,0,0,0,0,0,0,0,0,2.500,0,2.539,0,0,0,0,0,0,...
```

The following metadata file corresponds to the regular ARFF-file above:

```
DocumentType
Document
TextUnit
D1_Aktie = Aktie; Descriptor Frequency = 37; Descriptor Weight = 5.521
...
D73_Anspruch = Anspruch; Descriptor Frequency = 9; Descriptor Weight = 6.935
```

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

```
@relation 'DIAsDEM'
@attribute DocumentType string
@attribute Document string
@attribute TextUnit string
@attribute D1_Aktie real ...
@attribute D73_Anspruch real
@data
{0 null, 1 /home/.../volume100000.xml:0, 2 0}
{0 null, 1 /home/.../volume100000.xml:0, 2 3, 20 2.063002815336676,
   21 2.89591193827178, 25 0.6341488397979895, 62 3.0012724539296065}
{0 null, 1 /home/.../volume100003.xml:0, 2 1, 10 3.0012724539296065,
   64 2.89591193827178, 69 4.099884742597716}
```

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. Indirect references from one non-descriptor term via other non-descriptors to the corresponding descriptor term are supported. Note that DIAsDEM-specific thesauri must only include grammatical root forms of terms (i.e., their so-called lemma forms) as determined by the task *Actions → Prepare Data Set → Vectorize Text Units 2.2*. Thesauri can be created by *Actions → Understand Domain → Establish Initial Thesaurus* and modified by *Tools → Thesaurus Editor 2.2*. Example:

```
# Terms of Thesaurus /home/.../data/parameters/thesauri/de/Case123Thesaurus.dth
19535 "<<company>>" 1 "TY=D" "HL=-" "SY=-" "BT=-" "NT=-" "UD=-" "SN=Case1"
19534 "<<person>>" 1 "TY=D" "HL=-" "SY=-" "BT=-" "NT=-" "UD=-" "SN=Case1"
19461 "Ablehnung" 1 "TY=D" "HL=-" "SY=-" "BT=-" "NT=-" "UD=-" "SN=Case2"
10628 "abschließen" 1 "TY=D" "HL=-" "SY=-" "BT=-" "NT=-" "UD=-" "SN=Case1 Case2" ...
12299 "Änderung" 1 "TY=D" "HL=-" "SY=-" "BT=-" "NT=-" "UD=-" "SN=Case1 Case2" ...
10859 "Zwecke" 1 "TY=N" "HL=-" "SY=-" "BT=-" "NT=-" "UD=Tätigkeit" "SN=-"
10797 "ändern" 1 "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, hierarchy level (“HL=-”), synonyms (“SY=-”), broader term (“BT=-”), and narrower term (“NT=-”) are not used in DIAsDEM Workbench 2.2.

4.4.2 Cluster Text Unit Vectors (Weka)

Text Unit Vectors File: Valid local file name containing text unit vectors to be clustered in regular ARFF format as specified above in Subsection 4.4.1. 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, which contains the mappings of text units onto clusters in CSV format as specified below in Subsection 4.4.3. 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 Workbench, which 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:

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

4.4.3 Monitor Cluster Quality 2.2

Result File Format: *Cluster Result File* maps text unit vectors onto their respective clusters that 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 DIAsDEM document ID 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) must exactly correspond to the attributes "Document" and "TextUnit" of text unit vector files, as described in Subsection 4.4.1. 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:

```
/home/.../tutorial/trainingProject/inputCollection/volume100000.xml:0,0,25
/home/.../tutorial/trainingProject/inputCollection/volume100000.xml:0,1,25
/home/.../tutorial/trainingProject/inputCollection/volume100000.xml:0,2,9
/home/.../tutorial/trainingProject/inputCollection/volume100000.xml:0,3,48
/home/.../tutorial/trainingProject/inputCollection/volume100000.xml:0,4,34
```

As of DIAsDEM Workbench 2.2, clustering results should not be exported as TXT files with fixed width values. The tasks *Monitor Cluster Quality 2.2* and *Tag Text Units* cannot properly process DIAsDEM documents whose IDs comprise more than 25 characters. However, below is an example of a text unit vector file in fixed width values format:

```
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/volume100000.xml:0␣␣␣␣␣␣1␣␣␣␣␣␣␣␣␣25
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/volume100000.xml:0␣␣␣␣␣␣2␣␣␣␣␣␣␣␣␣25
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/volume100000.xml:0␣␣␣␣␣␣3␣␣␣␣␣␣␣␣␣9
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/volume100000.xml:0␣␣␣␣␣␣4␣␣␣␣␣␣␣␣␣48
null␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/volume100000.xml:0␣␣␣␣␣␣5␣␣␣␣␣␣␣␣␣34
```

As indicated above, only blank spaces are allowed to separate attribute values from each others. The following metadata 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 Subsection 4.4.1.

4.4.4 Tag Text Units

Result File Format: One of two result file formats (i.e., comma separated values and fixed width values), which are supported by DIAsDEM Workbench and described in Subsection 4.4.1.

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

Cluster Label File: Valid local file name of existing file created by DIAsDEM Workbench in *Actions* → *Postprocess Patterns* → *Monitor Cluster Quality 2.2* and possibly modified by *Tools* → *Cluster Label Editor 2.2.*

4.5 XML Tagging of Texts

4.5.1 Derive Conceptual DTD 2.2

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

```
#This is an automatically created file: Please do not edit this file manually!
#Sun Sep 02 16:44:06 CEST 2007
NUMBER_OF_UNTAGGED_TEXT_UNITS=1171
ELEMENTS_FILE_NAME=conceptualDtd.dcd.elements
MIN_ATTRIBUTE_REL_SUPPORT=0.1
ROOT_ELEMENT=CommercialRegisterEntry
CONCEPTUAL_DTD_FILE_NAME=/home/.../tutorial/trainingProject/conceptualDtd.dcd
NUMBER_OF_TEXT_UNITS=9254
XML_DTD_FILE_NAME=conceptualDtd.dcd.xml
CONCEPTUAL_DTD_REMARKS=Created Sun Sep 02 16\:43\:59 CEST 2007
NUMBER_OF_TAGGED_TEXT_UNITS=8083
NUMBER_OF_DOCUMENTS=985
TRAINING_COLLECTION_FILE_NAME=/home/.../tutorial/trainingProject/collection.dcf
ATTRIBUTES_FILE_NAME=conceptualDtd.dcd.attributes
```

The file `/home/.../tutorial/trainingProject/conceptualDtd.dcd.elements` contains metadata about DTD elements (i.e., XML tags). The file `/home/.../tutorial/trainingProject/conceptualDtd.dcd.attributes` contains metadata about attributes of DTD elements (i.e., XML tags)

4.5.2 Tag Documents 2.2

Conceptual DTD File: Valid local file name of existing file, which is created by DIAsDEM Workbench and contains metadata about the concept-based XML document type definition, its XML tags, and their attribues in a DIAsDEM-specific format.

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 *Conceptual DTD File*, this file is input to the task *Tools → Tagging Quality Evaluator 2.2*. For example, see below three lines of *Random Sample File* as created in this case study. Note that the first three line correspond semantically annotated sentences whereas the fourth ones contains an untagged sentence.

```
/home/.../tutorial/trainingProject/inputCollection/volume100002.xml:0
  <IfAppointmentOfOneManagingDirector_SolePowerToRepresent>Ist nur ein
  Geschäftsführer bestellt, so vertritt er die Gesellschaft einzeln.
  </IfAppointmentOfOneManagingDirector_SolePowerToRepresent>
/home/.../tutorial/trainingProject/inputCollection/volume100003.xml:0
  <PurposeOfCompany>(Gegenstand: Durchführung von vermessungstechnischen
  Arbeiten).</PurposeOfCompany>
/home/.../tutorial/trainingProject/inputCollection/volume100005.xml:0
  <NameOfMerchant Person="3; Stefan Thümmler; null; null; null; null;
  null; null; null; null">Inhaber: Stefan Thümmler, Kaufmann, Wustermark.
  </NameOfMerchant>
/home/.../tutorial/trainingProject/inputCollection/volume100135.xml:0
  Die Gründer der Gesellschaft, die sämtliche Aktien übernommen haben ist ECC
  Treuhand- und Verwaltungsgesellschaft mbH mit Sitz in München.
```

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)
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
TF	term frequency
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 also includes a concise introduction to the German Commercial Code [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

[CMB$^+$02] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Cristian Ursu, and Marin Dimitrov. *Developing Language Processing Components with GATE (a User Guide): For GATE version 2.0.* The University of Sheffield, Sheffield, 2002.

[ESK04] Levent Ertöz, Michael Steinbach, and Vipin Kumar. Finding topics in collections of documents: A shared nearest neighbor approach. In Weili Wu, Hui Xiong, and Shashi Shakhar, editors, *Clustering and Information Retrieval*, volume 11 of *Network Theory and Applications*, pages 83–103. Kluwer Academic Publishers, Boston, Dordrecht, 2004.

[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.

[JP73] R. A. Jarvis and A. Patrick, Edward. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, C-22(11):1025–1034, November 1973.

[Koh01] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer-Verlag, Berlin, Heidelberg, third edition, 2001.

[LYRL04] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, April 2004.

[MTU+01] Diana Maynard, Valentin Tablan, Christan Ursu, Hamish Cunningham, and Yorick Wilks. Named entity recognition from diverse text types. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP-2001)*, Tzigov Chark, Bulgaria, September 2001.

[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.

[RSW02] Tony G. Rose, Mark Stevenson, and Miles Whitehead. The Reuters Corpus Volume 1 - from yesterday's news to tomorrow's language resources. In *Proceedings of Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 827–833, Las Palmas, Canary Islands, Spain, May 2002. European Language Resources Association.

[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 2007-09-15.

[SKK00] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *Workshop on Text Mining at the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–110, Boston, MA, USA, August 2000.

[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.

[WF05] Ian H. Witten and Eibe Frank. *Data Mining: Practical Mechine Learning Tools and Techniques.* Morgan Kaufman Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, second edition, 2005. Weka is available at http://www.cs.waikato.ac.nz/∼ml/weka, accessed 2007-09-15.

[Win03] Karsten Winkler. Getting started with DIAsDEM Workbench 2.0: A case-based approach. HHL Working Paper No. 58, HHL – Leipzig Graduate School of Management, Leipzig, Germany, 2003. DIAsDEM Workbench 2.0 is available at http://www.hypknowsys.org/, accessed 2007-09-15.

[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.