Computational Linguistics for Mere Mortals
Powerful but Easy-to-use Linguistic Processing for Scientists in the Humanities

Rüdiger Gleim, Alexander Mehler

Department for Computing in the Humanities, Goethe University, Georg-Voigt Straße 4, Frankfurt, Germany
Texttechnology/Applied Computer Science, Bielefeld University, Universitätsstraße 25, Bielefeld, Germany
gleim@em.uni-frankfurt.de, mehler@em.uni-frankfurt.de

Abstract

Delivering linguistic resources and easy-to-use methods to a broad public in the humanities is a challenging task. On the one hand users rightly demand easy to use interfaces but on the other hand want to have access to the full flexibility and power of the functions being offered. Even though a growing number of excellent systems exist which offer convenient means to use linguistic resources and methods, they usually focus on a specific domain, as for example corpus exploration or text categorization. Architectures which address a broad scope of applications are still rare. This article introduces the eHumanities Desktop, an online system for corpus management, processing and analysis which aims at bridging the gap between powerful command line tools and intuitive user interfaces.

1. Introduction

With the availability of tools, resources and standards for annotating and processing data in the humanities, we are in need of frameworks that integrate information objects and corresponding operations for handling these objects. Moreover, systems are needed with interfaces that provide a low barrier to their use. The reason is that humanists look at these systems not as experts in computational linguistics or machine learning, but as non-experts, that is, as users not as researchers. Thus, versatility of easy-to-use methods in the humanities is a major criterion beyond expressiveness of the underlying data model and the efficiency of algorithms that operate on this data.

Humanists have an integrated view on text-technology and related disciplines as they want to use resources (e.g., text, image or video corpora as well as lexica and ontologies) together with machine learning facilities in such a way that they can abstract from the underlying data formats and any restrictions of their transformation and further processing. This makes the development of text-technological frameworks for eHumanities a challenging task as one needs both high-level processing routines for handling linguistic and, more generally, semiotic data as well as information systems that encapsulate these routines by versatile interfaces. Moreover, these systems should be open in the sense of not only being completely web-based (such that any installation effort can be disregarded). Rather, they should allow for linking system-internal data with the web and its numerous knowledge resources (as part, for example, of the Wikimedia foundation).

In this paper, we describe the eHumanities Desktop as such a system for corpus management, processing and analysis which aims at bridging the gap between powerful command line tools and intuitive user interfaces. More specifically, in Section (1.1.) we discuss related work in this field of research. In Section (2.) we describe the Architecture of the eHumanities Desktop while in Section (3.) we describe several (linguistic) applications that can be used with this desktop. Finally, we conclude and give a prospect on future work in Section (4.).

1.1. Related Work

In the past there has been much effort to develop environments and frameworks for managing corpora and machine learning facilities in computational linguistics and related disciplines. Amongst others, this relates to the GATE system, the WEKA framework and the TextGrid environment.

- GATE (Cunningham et al., 2002) has been developed as a framework for deploying and developing programs in language engineering (LE) including, amongst others, tokenizers, sentence splitters, taggers, named entity recognizers and many related modules. That is, as a *development environment*, GATE does not only provide facilities in text technology, but also addresses the development of systems for solving tasks in LE such as, information extraction or text summarization. GATE includes modules for representing, editing and using ontologies (Bontcheva et al., 2004) as well as for processing of multimedia data (Dowman et al., 2005). Moreover, GATE integrates machine learning modules as based, for example, on *Support Vector Machines* (SVM) (Li et al., 2009). From the point of view of a humanist, GATE offers an overwhelming range of tools and development facilities that actually demand an expertise in computational linguistics and machine learning. This may discourage humanists to adapt GATE to their research needs whenever they look for easy-to-use tools that address some task in eHumanities. Therefore, a framework is needed which basically encapsulates the underlying text-technology and machine learning while providing tools on a level of understanding that is accessible to humanists. The eHumanities Desktop approaches this goal.

- WEKA (Hall et al., 2009) is a widespread workbench of tools in the area of machine learning that integrates tools for preprocessing, selecting, exploring, evaluating and visualizing data as input or output of machine learning. WEKA focuses on data mining. In contrast to this, the eHumanities Desktop includes systems for the management, annotation and (pre-)preprocessing

of information objects in the humanities – including texts and images (Gleim et al., 2010). This also means that while preprocessing in WEKA relates to converting data as input to machine learning, in the eHumanities it means, for example, tagging and lemmatization linguistic data for further processing by manual annotation or machine learning. However, the WEKA workbench provides excellent facilities so that integrating the Desktop with this workbench will be a task in the near future.

- With the rise of SVMs in machine learning, specialized libraries for computing SVMs in the area of text mining became more and more prominent in computational linguistics (Joachims, 2002). Currently, SVM^{light} (Joachims, 2002) and LIBSVM (Chang and Lin, 2001) are two such libraries that are heavily used in text categorization. Consequently, the eHumanities Desktop integrates a so called *ClassifierBuilder* that integrates SVM^{light} on the input and output level.

- TextGrid (Kerzel et al., 2009) has been developed as a web-based research environment to support information processing in the humanities. It allows for the collaborative generation, processing annotation, management and search of textual data by integrating lexica and dictionaries as basic linguistic resources. TextGrid mainly focuses on textual resources. In contrast to this, the eHumanities Desktop handles both symbolic (e.g. texts) and iconographic data (e.g. images) (Gleim et al., 2010). Further, other than TextGrid – but in the line of GATE (see above) – the eHumanities Desktop additionally integrates machine learning facilities. In this way, the Desktop is in support of managing, annotating and exploring semiotic data by humanists.

GATE, WEKA and TextGrid are excellent examples of processing frameworks that are specialized according to their focus on program development (GATE), machine learning (WEKA) or the handling of textual data (TextGrid). In order to integrate the eHumanities Desktop into this text-technological landscape, it will be further developed such that it provides corresponding input-output interfaces for these systems. In this way, the Desktop may find users not only in the area of humanities but also in the area of language engineering, computational linguistics and text-technology.

2. Architecture

The primary development objective was to build a platform which offers elaborate means to let users and groups organize resources and enable affiliated research projects to develop applications to process and analyze their linguistic data. So the system is sort of divided into two parts: The foundation is a core system which offers functionalities for resource, user and right management. On this basis a collection of APIs allow for modular development of tools and applications to work on the managed data.
Users can login to the eHumanities Desktop and work on their resources online using a common web browser. The look & feel is designed to imitate a typical desktop environment including document handling and using applications. This section is to give a brief overview of the architecture behind this graphical front end. Figure 1 depicts the layer structure of the components which constitute the system architecture. The client is implemented primarily in Java Script and communicates with a Java Servlet which handles requests to algorithms and resources. The servlet relies on a relational database to manage the master data, incorporates a set of different storage back ends to store resources according to their format and calls external applications for computation of results as necessary. In the following the central components are explained in more detail.

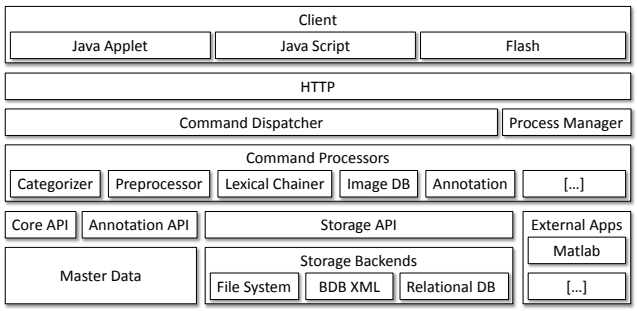

Figure 1: Diagram depicting the layer structure of the server architecture.

2.1. Client

Regarding the architecture of client/server systems the user interface may appear to be a minor matter. But the demands made on the client from both the users as well as the developers perspective are complex. As user one is interested in getting started without bothering too much with installation efforts and learning all the details of a new application environment. This means that not only interaction with the various application modules should follow established conventions but also the general workflow including the organization of the functions and resources. The latter aspect is especially important for the system being presented because it offers a variety of functions from different areas. Finally using a certain application should be made simple and straight-forward- but without oversimplification of parameters and functionality. From the developers perspective on the other hand easy maintenance and extensibility of the client are important.
The eHumanities Desktop attempts to meet these requirements by offering a desktop environment in a web browser which imitates the look and feel of modern operating systems. Figure 2 shows a typical working scenario. Applications can be accessed via a start menu or shortcut icons on the virtual desktop. Multiple applications are displayed in separate windows which can be maximized or minimized to a task bar at the bottom. Complex operations such as computing support vector machine models which can take minutes or hours are executed as processes on the server. The user can check the progress of such operations in a graphical process manager (see upper right corner in figure 2) and cancel them if necessary. It is not required to stay online for the time of such computations- users can login later

Figure 2: Screenshot of a typical working scenario.

to check upon the processes and review the results once they are ready. In order to minimize system requirements the eHumanities Desktop can be accessed platform independently via common web browsers. There is no need for installation whatsoever. The client is implemented in Java Script and relies on the ExtJS[1] Framework to offer the desktop look and feel. Some applications need a Flash or Java Plugin to be installed to allow for more complex operations such as interactive graph visualization. All computations are performed on the server, that is the client is purely used for user interaction. The processing of user requests is discussed in the next section.

2.2. Server

The server architecture is hierarchically structured into layers which build upon another as illustrated in figure 1. Its primary task is to receive requests from the clients and perform computations on resources or the master data accordingly. Request handling and execution to start with is based on an adaption of the *command pattern* (Freeman et al., 2004): The client sends a command and its parameters to the server and waits for it to respond with the results or feedback on an operation being done. Commands are represented as JSON[2] objects and exchanged with the server via HTTP requests.

Incoming commands, such as to fetch the annotation of a document are handled by the *Command Dispatcher* which checks well formedness and user authentication. The execution of the command constitutes a new *process* which is registered in the *Process Manager* so that the user can monitor and abort it if necessary. The set of commands which are understood by the system are grouped according to their function and managed in a registry. This registry is used by the dispatcher to decide which *Command Processor* has to be called in order to execute the command. The Command Processor performs the actual computation according to the specific command which usually incorporates at least one of the following components: (i) The *Master Database* which represents users, resources and their interrelations; (ii) the *Storage API* which manages access to the underlying *Storage Back ends*; or (iii) an external application such as MATLAB. After having completed the computation control is handed back to the dispatcher which returns the results to the client.

The servlet provides several APIs which abstract from technical details of the underlying layers. That way developers of new functions (i.e. Command Processors) can concentrate on the specific task rather than to reinvent the wheel over and over again by coding user management, storage handling and so on. Because of the modular approach the system is easily extensible as new demands for features arise.

2.3. Storage Back Ends

A foundation of the eHumanities Desktop is to offer means to upload documents, organize them in repositories and share those resources with other users. On this basis application modules can then operate to perform analysis on the data. Therefore it is important to offer efficient means to access and process such documents. When a document is newly inserted into the system a file format (or mime type) detection is performed in order to determine which storage back end should be used. Currently three back ends are supported: XML documents are stored using Oracle BerkeleyDB XML[3], a native XML database management system (DBMS). This allows for performing XQuery statements over stored documents and entire collections. Also transactional updates of existing documents are supported. Relational databases can be managed as documents as well: They can either be newly created by uploading an appropriate SQL script or by specifying authentication information to include an existing database. Relational databases are for example used to efficiently represent and query lexicons. Binary data and any other file formats which do not fit in the former categories are stored as files. The *Storage API* helps abstracting from the details of how contents of a specific document are stored. From the users perspective the distinction between the storage back ends is transparent.

2.4. Master Data Model

The master data model basically serves the representation of users, resources and their access permissions. The demands on this data model need to meet the requirements of typical research project settings: Some resources should only be accessible by certain users. Some users should be able to edit documents, while others should not. Users should be organizable into groups etc.

The basic idea behind the data model used in the eHumanities Desktop (see figure 3) is a mixture of concepts from unix operating systems and relational database systems. The model basically distinguishes between *Authorities* which have a certain degree (read, write, delete, grant) of *Access Permission* on *Resources*. An authority is either a *User* or a *Group*. A user can be member of an arbitrary number of groups and thus inherits their respective privileges. A group has a designated owner (for example the head of a research project) who can add or remove users from the group. *Resources* on the other hand are primarily distinguished into *Documents*, *Repositories* and *Fea-*

[1] http://extjs.com
[2] http://www.json.org
[3] http://www.oracle.com/database/berkeley-db/xml/index.html

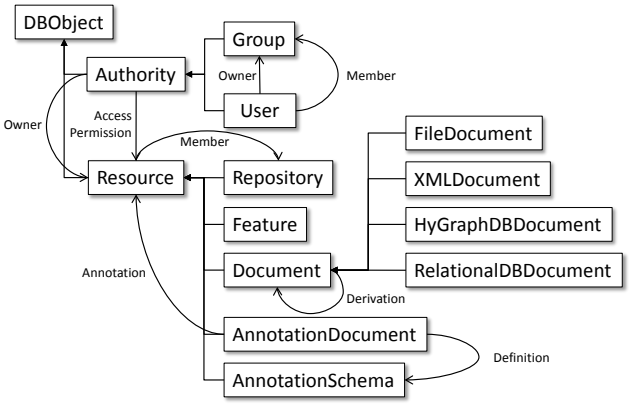

Figure 3: Master Data Model of the eHumanities Desktop.

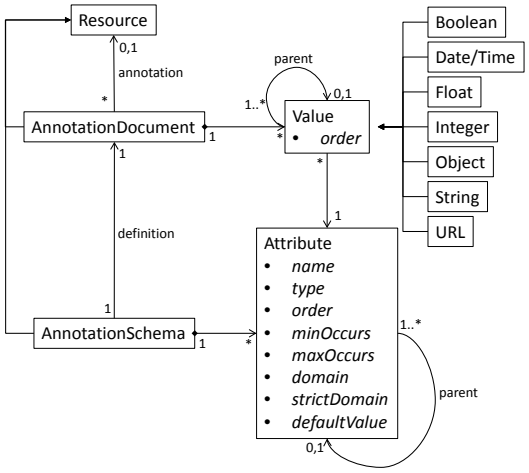

Figure 4: UML class diagram of the annotation subsystem.

tures. In contrast to common file systems, documents are not tightly linked to a specific directory. They can rather be member of an arbitrary number of repositories. Repositories can be member of other repositories as well. That way it is possible to freely structure the same set of documents in different manners which is very useful when compiling linguistic corpora for example. Since the eHumanities Desktop allows for online conversion and processing of documents which lead to derived documents, such relations are also stored in the database to track back their origin. *Features*, that is application modules of the system are also considered to be resources. This allows for a fine grained control not only over which user can access a specific resource but also which functions he is allowed to use. In terms of object oriented programming documents are further distinguished to allow for methods which are specific for the storage back end being used. However this distinction is not relevant to the user.

2.5. Annotation Subsystem

The eHumanities Desktop puts emphasis on extensible resource annotation to reflect the dynamics of how resources are annotated in practice. Figure 4 depicts the core classes of the annotation subsystem. Resources can be annotated with *Annotation Documents*. The structure of the annotation documents is defined by *Annotation Schemas*. A schema basically defines the hierarchy of attributes which can be used for annotation. An *Attribute* is defined among others by its name, data type, value domain and default value. Furthermore it can be defined how many times an attribute may be set within an annotation. This is for example necessary to allow an attribute which should represent keywords for an annotated book to be set multiple times, whereas the title of a book should be set exactly once. The value types supported include boolean and numeric values, dates, URLs, strings and references to other resources in the system. Attributes can be organized into an ordered tree hierarchy. Since both annotation documents and schemas are derivations of resources, they can be subject of annotation themselves. This also means that they fall under the right management of the system. This enables annotators of resources to exactly define who can read or write their annotations.

The system does not restrict users to a fixed set of annotation schemas but offers a graphical interface to define custom ones. Figure 5 shows a screenshot of editing a sample schema which could be used for annotating scientific articles: On top level the attribute hierarchy offers fields to annotate the title, the author and references to other articles. The author attribute exemplifies the use of the attribute hierarchy: An author consists of two sub attributes which represent the name and affiliation. The reference attribute allows to refer to other articles in the system and thus help building a network of annotated documents. This example is rather simple. However it can be extended and edited even if the schema is already being used. Existing annotations are automatically adjusted as changes are made.

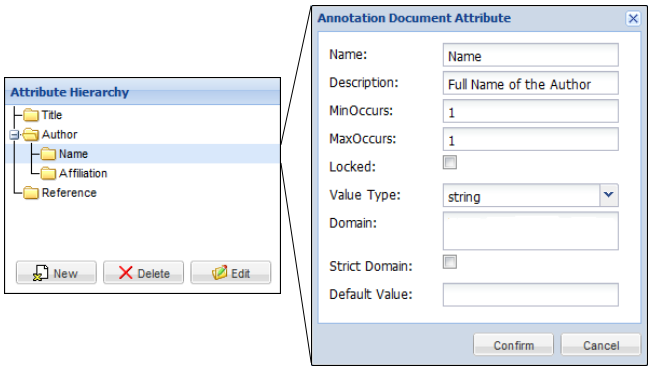

Figure 5: Example of editing an Annotation Schema.

3. Applications

The eHumanities Desktop highly benefits from affiliated research projects and cooperations (see acknowledgements) which give input in terms of new requirements as well as new application modules. This section presents some of the major linguistic applications. In order to give an impression of a typical workflow we assume an exemplary research project as guideline where a corpus of newspaper articles shall be constructed.

3.1. Corpus Manager

The *Corpus Manager* is so to speak the heart of the system. It offers means to upload, organize, query and down-

load resources as well as sharing them with other users and groups. The graphical user interface (see figure 6) offers a look and feel similar to the Windows Explorer. A tree on the left side of the window shows the hierarchy of the repositories visible to the user. The center part of the window shows the documents of the currently selected repository. The list of documents can be filtered and sorted in many ways to help users quickly find the resources they are looking for. Besides the basic fields like size, mime type, date of creation and alike which are available for all documents it is also possible to add fields to the view which stem from annotations of the respective documents. If for example documents were annotated based on a schema which provides an attribute for category information, this could be displayed seamlessly in the Corpus Manager. A virtual repository called *All Documents* gives access to all documents the user has access to. The documents and repositories can quickly be re-structured using common drag&drop or cut&paste operations. Finally the corpus manager offers means to annotate resources according to previously defined annotation schemata.

A researcher who wishes to construct a newspaper corpus would start here by creating an initial structure of repositories, for example as shown in figure 6. In this example there are some PDF documents which have to be uploaded from the local computer and some HTML documents which shall directly be taken from the web. The former can be uploaded using the *Upload Manager* which provides means to uploads single files or an entire directory into a designated repository. The latter can be accomplished by an URL upload assistant which based on a given URL downloads a resource from the web and inserts it as new document into a selected repository. The advantage for the user here is that the download is being done by the server and does not affect the users bandwidth.

The eHumanities Desktop offers a conversion matrix which allows users to convert uploaded documents online into other formats, for example from PDF to plain text.

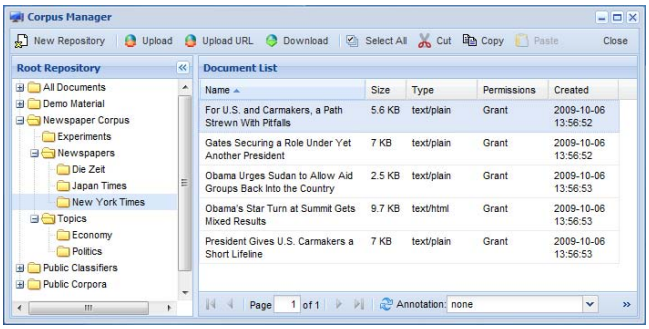

Figure 6: Screenshot showing the Corpus Manager.

3.2. Preprocessor

A common task in NLP is the preprocessing of input texts. The *Preprocessor* module enables users to preprocess their uploaded documents. Besides language detection, sentence boundary detection, tokenization, lemmatization, stemming and name entity recognition, the preprocessing system includes a trigram HMM-Tagger (Mehler et al., 2008). The currently released implementation supports English, German and Latin texts. The tagging module was trained and evaluated based on the German Negra Corpus (Uszkoreit et al., 2006) (F-score of .975), the Tübinger Spoken Language Corpus (0.982) and the English Penn Treebank (Marcus et al., 1994) (0.956). The language identification component was successfully evaluated against samples from Wikipedia articles, proving that only a small amount of input data is needed (F-score of 0.956 for 50 characters and 0.97 for 100 characters). The output of the preprocessor can be displayed directly in the browser or saved as a new TEI P5 (Burnard, 2007) document in the Oracle BDB XML database.

The preprocessor supports ad hoc preprocessing of texts which have been inserted via cut & paste into a form as well as batch processing of documents which have already been uploaded into the system. To follow the exemplary research project the user would now select all collected documents and let them be transformed into TEI P5 XML documents. Since the preprocessor attempts to automatically convert input documents as required, usually no explicit conversion from PDF, HTML and so on is necessary.

3.3. Annotation

The TEI P5 documents which have been created in the preceding step are the artifacts on which all subsequent studies are based. Now the annotation subsystem can be used to annotate these documents to help improve later analysis. The researcher uses the *Annotation Schema Editor* to define a schema to be used for annotating newspaper articles. Based on this schema the articles can now be annotated as shown in figure 7.

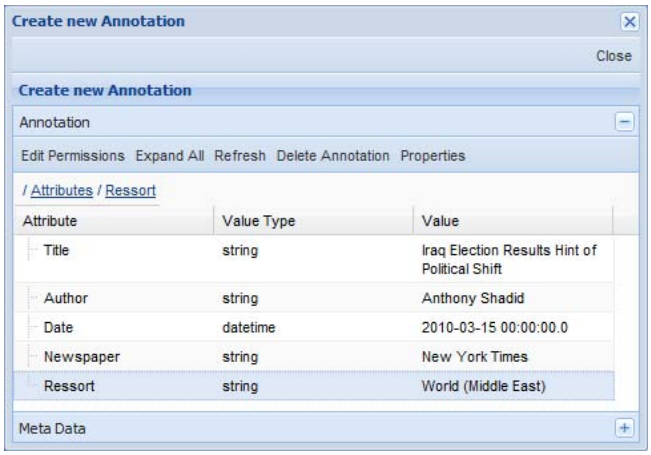

Figure 7: Screenshot showing the annotation of a newspaper article.

3.4. HSCM

Now that the texts have been uploaded into the system, properly preprocessed and annotated, an interface is needed to browse and search the documents. If queries for quotations of single tokens or phrases are needed, the *Historic Semantics Corpus Management* (HSCM) is a good place to start with. The HSCM (Jussen et al., 2007) aims at a text-technological representation and quantitative analysis

of chronologically layered corpora. A prominent instance is the *Patrologia Latina*[4] which has been tagged and converted into the TEI P5 format. The system has been integrated into the eHumanities Desktop and extended to be usable with arbitrary corpora. Using the HSCM enables users to search for texts which contain a certain phrase, for example *economic crisis*. Subsequently all sentences containing this phrase with a selected number of neighbors can be listed. Figure 8 shows a screenshot of the HSCM with quotes of the phrase "Helmut Schmidt" in a German newspaper. The corpus on which the query was based contained 87761 articles which have been searched to identify 165 quotes.

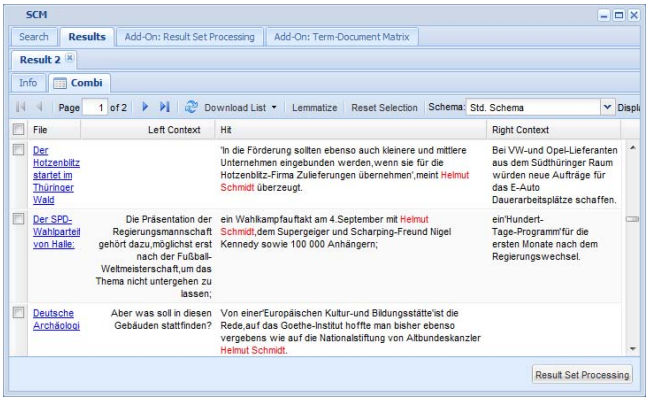

Figure 8: Screenshot of the HSCM showing quotes of the phrase "Helmut Schmidt" in a German newspaper.

3.5. XML Query Interface

The browsing and querying functions which the HSCM offers are quietly transformed into XQuery statements which are executed on the XML document collections. Since the HSCM can only offer a set of queries which are commonly used there may be the need to ask more specific questions which are not covered per default. In such cases it is possible to enter and execute XQueries directly via the Desktops *XML Query Interface*. Since XQueries can be very expensive in terms of computation time this feature is currently restricted to single documents.

3.6. Categorizer

The eHumanities Desktop offers an approach for automatic text categorization based on a set of predefined classifiers. The current release of the eHumanities Desktop supports *Dewey-Decimal Classification* (DDC) classifiers (Mehler and Waltinger, 2009) for English and German as well as a classifier based on the German newspaper *Süddeutsche Zeitung* (Waltinger et al., 2009) using support vector machines (Joachims, 2002). However these classifiers are not implemented statically but defined using a light-weight XML based *Classifier Definition Language*. Such a definition defines which algorithm to use, its parameters and which additional files are needed (e.g. model files in the case of SVMs). In order to use the categorizer a user selects the classifier to be applied and specifies either an input document or plain text via cut&paste. The result is presented in a list of category labels which is ordered according to the respective relevance. Figure 9 shows the categorization result of a newspaper articles based on the English DDC. In the line of the newspaper research projects a user can apply the categorizer to perform an automatic categorization of articles. It is also possible to let the categorizer annotate the processed documents based on the annotation system of the eHumanities Desktop.

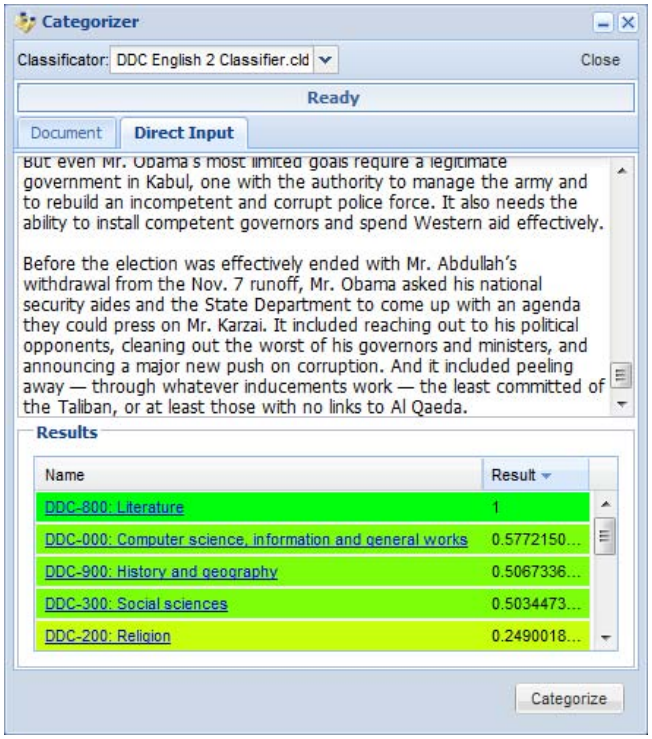

Figure 9: Screenshot showing the categorization of a newspaper articles based of the DDC.

3.7. Lexical Chainer

As a further linguistic application module a lexical chainer (Waltinger et al., 2008; Teich and Fankhauser, 2004) has been included in the online desktop environment. That is, semantically related tokens of a given text are tracked and connected by means of a lexical reference system. The system currently uses two different terminological ontologies – *WordNet* (Fellbaum, 1998) and *GermaNet* (Kunze and Lemnitzer, 2002) – as chaining resources which have been mapped onto the database format. The list of resources for chaining can easily be extended. The lexical chainer can generate HTML output which offers intuitive means to examine the results.

3.8. Lexicon Browser

Section 2.3. pointed out the importance of applying storage mechanisms which best fit the usage scenario of a given resource. On the other hand the resource management should abstract from the details so that the user can manage all documents equally irrespective of their internal representation. An example of this approach is the representation of lexica within the eHumanities Desktop. Internally the system manages lexicons as relational databases which provide efficient means for random access as well as complex queries.

[4] http://pld.chadwyck.co.uk/

Lexicon documents can be organized in the corpus manager and shared with other users. But in order to browse and query lexica the eHumanities Desktop offers a decicated module: The *Lexicon Browser* enables users to query lexcial resources and apply various filters which include support for wildcards, part of speech and numerical constraints like frequency, inverse document frequency (based on on a specified reference corpus) and alike. The measures which are based on a reference corpus are not fixed but can be computed online by selecting a lexicon and a collection of TEI P5 documents which serve as reference.

4. Conclusion

We presented the eHumanities Desktop, an online system for resource management, processing and analysis in the humanities. The architecture puts emphasis on an easy usability of linguistic resources and methods. However it is not restricted to linguistics- there are other applications as for example the *ImageDB* which have not been presented here. The Desktop is in ongoing development and feedback is always welcome. Currently emphasis is put on enabling users to build custom classifiers to help them automatically annotate their resources as well as improving management of lexical resources. Researchers interested in testing the system are welcome to contact[5] us in order to get an account.

5. Acknowledgements

Support of the German Federal Ministry of Education (BMBF) through the research project *Linguistic Networks* at Bielefeld University is gratefully acknowledged.

6. References

K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. 2004. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3/4):349—373.

Lou Burnard. 2007. New tricks from an old dog: An overview of tei p5. In Lou Burnard, Milena Dobreva, Norbert Fuhr, and Anke Lüdeling, editors, *Digital Historical Corpora- Architecture, Annotation, and Retrieval*, number 06491 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: an architecture for development of robust hlt applications. In *In Recent Advanced in Language Processing*, pages 168–175.

M. Dowman, V. Tablan, H. Cunningham, C. Ursu, and B. Popov. 2005. Semantically Enhanced Television News through Web and Video Integration. In *Second European Semantic Web Conference (ESWC'2005)*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge.

Elisabeth Freeman, Eric Freeman, Bert Bates, and Kathy Sierra. 2004. *Head First Design Patterns*. O' Reilly & Associates, Inc.

Rüdiger Gleim, Paul Warner, and Alexander Mehler. 2010. eHumanities Desktop — an architecture for flexible annotation in iconographic research. In *Proceedings of the 6th International Conference on Web Information Systems and Technologies (WEBIST '10), April 7-10, 2010, Valencia*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Thorsten Joachims. 2002. *Learning to classify text using support vector machines*. Kluwer, Boston.

Bernhard Jussen, Alexander Mehler, and Alexandra Ernst. 2007. A corpus management system for historical semantics. *Sprache und Datenverarbeitung. International Journal for Language Data Processing*, 31(1-2):81–89.

Martina Kerzel, Jens Mittelbach, and Thorsten Vitt. 2009. Textgrid. *KI – Zeitschrift Künstliche Intelligenz*, 4/09:36–39.

Claudia Kunze and Lothat Lemnitzer. 2002. GermaNet – representation, visualization, application. In *Proc. of LREC 2002*, pages 1485–1491.

Y. Li, K. Bontcheva, and H. Cunningham. 2009. Adapting SVM for data sparseness and imbalance: A case study on information extraction. *Natural Language Engineering*, 15(2):241–271.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Alexander Mehler and Ulli Waltinger. 2009. Enhancing document modeling by means of open topic models: Crossing the frontier of classification schemes in digital libraries by example of the DDC. *Library Hi Tech*, 27(4).

Alexander Mehler, Rüdiger Gleim, Alexandra Ernst, and Ulli Waltinger. 2008. WikiDB: Building interoperable wiki-based knowledge resources for semantic databases. *In Sprache und Datenverarbeitung. International Journal for Language Data Processing*.

Elke Teich and Peter Fankhauser. 2004. WordNet for lexical cohesion analysis. In *Proc. of the 2nd Global WordNet Conference, January 20-23, 2004*, pages 326–331.

Hans Uszkoreit, Thorsten Brants, Sabine Brants, and Christine Foeldesi. 2006. NEGRA Corpus. `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/`.

Ulli Waltinger, Alexander Mehler, and Gerhard Heyer. 2008. Towards automatic content tagging: Enhanced web services in digital libraries using lexical chaining. In *Proc. of WEBIST'08, Funchal, Portugal*. Barcelona.

Ulli Waltinger, Alexander Mehler, and Rüdiger Gleim. 2009. Social semantics and its evaluation by means of closed topic models: An SVM-classification approach using semantic feature replacement by topic generalization. In *Proceedings of the GSCL-Conference, Potsdam (DE), 2009.*

[5] `http://www.hucompute.org/hudesktop`