eHumanities Desktop
An extensible Online System for Corpus Management and Analysis

Rüdiger Gleim
Goethe Universität Frankfurt
gleim@em.uni-frankfurt.de

Alexander Mehler
Universität Bielefeld
alexander.mehler@uni-bielefeld.de

Ulli Waltinger
Universität Bielefeld
ulli_marc.waltinger@uni-bielefeld.de

Peter Menke
Universität Bielefeld
pmenke@googlemail.com

Abstract

This paper presents the *eHumanities Desktop* - an online system for corpus management and analysis in support of computing in the humanities. Design issues and the overall architecture are described, as well as an outline of the applications offered by the system.

1 Introduction

Starting from quantitative corpus linguistics and related fields, the extensive application of computer-based methods increasingly reaches all kinds of disciplines in the humanities. The spectrum of processed documents spans from textual contents (e.g. historical texts, newspaper articles, lexica and dialogue transcriptions), over annotation formats (e.g. of multimodal data), to images and multimedia. This creates new challenges in maintaining, processing and analysing resources, especially because research groups are often distributed over several institutes. Finally sustainability and harmonisation of linguistic resources becomes an important issue. Thus, the ability to work collaboratively on shared resources and ensure interoperability are important issues in the field of corpus linguistics (cf. Dipper et al. 2006, Ide and Romary 2004, Rehm et al. 2009).

The *electronic Humanities Desktop* (eHumanities Desktop) is designed as a general purpose platform for sciences in the humanities. It offers web-based means to

1. manage documents of *arbitrary type* and organise them freely in (possibly nested) repositories
2. share and collaboratively work on resources with other users and groups
3. use full-text search and views to browse through the database (including e.g. XQueries on XML documents)
4. process and analyse documents via an extensible Tool API (including e.g. Part of Speech Tagging in multiple languages, categorisation and lexical chaining)

The system is platform independent and offers a full-fledged desktop environment that supports work on corpora via any modern browser.

The eHumanities Desktop is in the line of systems which aim at the integration of resources and methods in different areas. Chiarcos et al (2008) presented *ANNIS*, a general framework for integrating annotations from different tools and tagsets. *GATE* (Cunningham 2002) is a system for flexible text categorisation and engineering. *Clarin* (Váradi et al. 2008) aims at a large-scale European research infrastructure to establish an integrated and interoperable infrastructure of language resources and technologies.

This paper is divided into two topics. The first part takes a more technical view of the system and describes the architecture, the master data model and presents aspects of resource

representation and management. The second part presents an outline of linguistics-related applications and functions which are available to the user.

2 Architecture

The primary objective during the development of the system was to achieve a light-weight core system which is *flexible* enough to be adapted to virtually any linguistic application, yet *powerful* enough to allow for mass data and distributed usage. The core functions include user and group management, repository and document handling and finally their interrelations in terms of access permissions. The core layer of the eHumanities Desktop offers a programmatic API which offers means to control the authorities (users and groups) and the resources they work on. It is designed so it is abstracted from internal representation and storage. All other components of the system are designed around this core API.

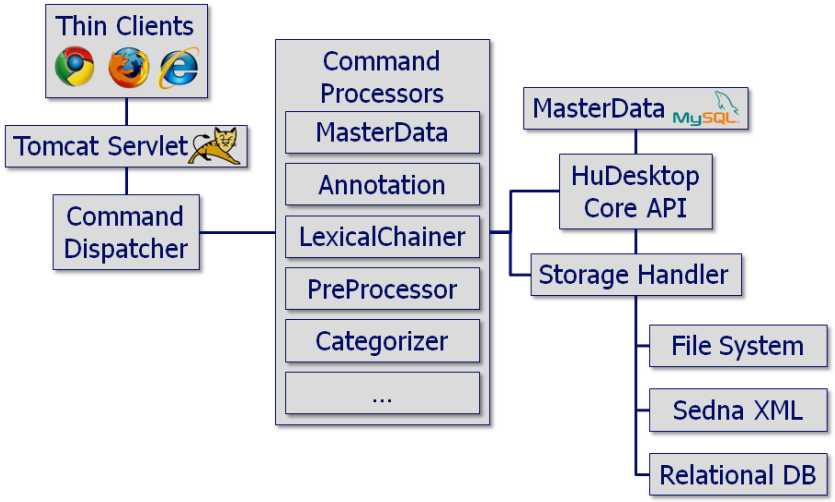

Figure 1: Diagram showing the overall architecture of the eHumanities Desktop.

The support of distributed and concurrent work on shared resources and applications is the central requirement from the users' perspective. Thus a central storage and management is needed, and this is implemented using a Java-based client/server approach. Figure 1 shows the overall architecture of the system. The client side is kept as lightweight as possible in order to avoid installation of special software. In order to use the eHumanities Desktop a user opens the system's URL[1] in a Javascript-enabled browser, types in his or her login data and is ready to go. The client offers a look and feel which imitates a typical desktop environment with desktop icons, start menu and a task bar. Figure 2 shows a typical working scenario. This way a user has full access to corpus management, processing and analysis tools from any laptop or desktop PC with an internet connection.

The server side is mostly implemented as a Java servlet which is run by an Apache Tomcat[2] servlet engine. The central component on the server side which accounts for the communication with the clients is the *Command Dispatcher*. Its task is to take commands from the client over the internet, perform security checks and hand them over to the *Command Processor* component in order to execute the request and compute the results. The results are then handed back to the Command Dispatcher which sends them back to the client where they are displayed. Adding a new application to the system usually means writing some Javascript code for the client and implementing a new Command Processor on the server side to implement requests. That way a good separation between different applications is achieved. Furthermore developers of Command Processors do not need any knowledge about the internals of the eHumanities Desktop. They rely on the *eHumanities Desktop Core API* which offers all methods needed to manipulate authorities and resources. Furthermore

Command Processors have the option to incorporate external applications and tools to perform their tasks. That way it is possible, for example, to include programs for statistical analysis like Matlab to work on data using the eHumanities Desktop.

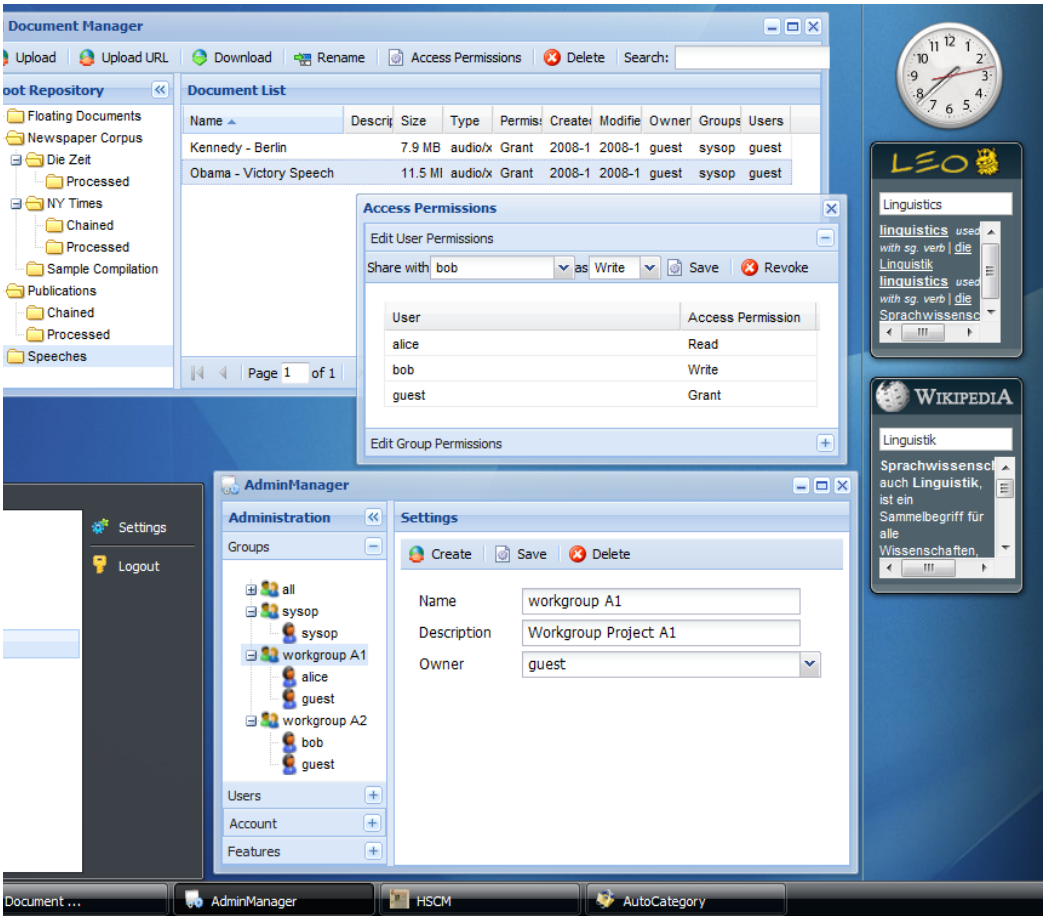

Figure 2: Screenshot of the eHumanities Desktop showing a typical working scenario.

The following example illustrates how a typical command is executed. In this example the user wants to preprocess a raw text. To do so he opens the preprocessor module on the client desktop environment, adds the text and triggers the raw text process. This command is then sent to the server along with the text to be processed. The Command Dispatcher checks if the user is valid and has the permission to use the text preprocessor. Then the request is forwarded to the *PreProcessor* which calls an external program to perform the task. If the user chose to store the results permanently in the system as a new document, the *Storage Handler* is asked to pick a fitting storage back-end (which depends on the format of the data, or more specific: the MIME type) and persistently stores the data. Finally the preprocessed text, represented in Text Encoding Initiatives P5 format (cf. Burnard 2007) is returned via the Command Dispatcher back to the browser where it is displayed.

2.1 Master Data Model

Typical usage scenarios include distributed research groups that work collaboratively on shared resources. This requires fine-grained access management on both resources and applications. Furthermore sophisticated means are required to organise documents and repositories. Figure 3 illustrates the object oriented design approach of the master data which is implemented in the system.

The principal concept is that *Authorities* have a certain level of *Access Permission* on *Resources*. The access permission can be either *read*, *write*, *delete* or *grant,* as commonly

understood from widely-used relational database systems or file systems. Table 1 explains the different levels of access on documents and repositories. An authority can either be a *User* or a *Group*. A user can be member of an arbitrary number of groups and inherits all permissions of those respective groups. A group has a designated owner who can control the memberships.

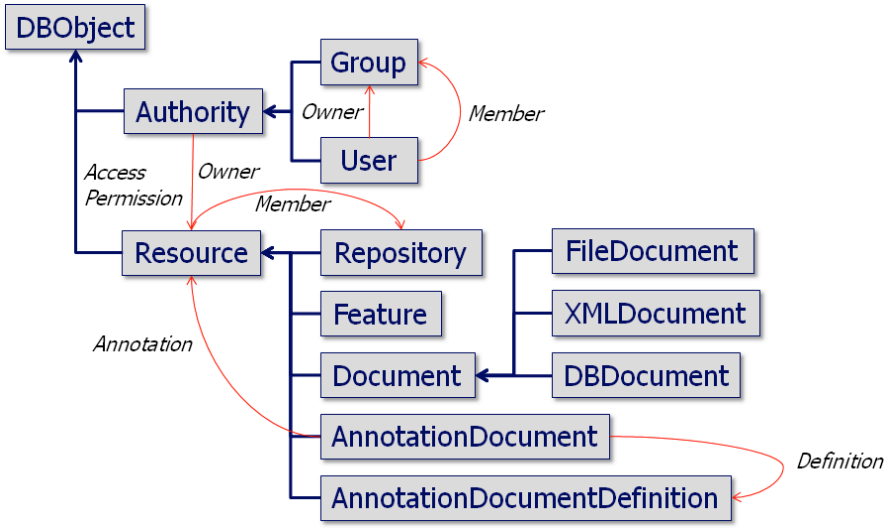

Figure 3: Diagram depicting the master data model of the eHumanities Desktop.

The term *Resource* is used in a rather broad sense. The most important instance is the *Document* which represents a document that is physically stored in an XML database or a relational database. In contrast to common file systems, documents are not strictly bound to a specific directory. A document can be part of an arbitrary number of *Repositories* (including none). This allows for compiling different corpora without the need to create physical copies of the contained documents. Furthermore users or groups can organise shared documents in repositories according to their needs - they do not necessarily need to agree on one specific structure.

Many application scenarios require means to annotate documents, for example newspaper articles, videos from experiments, or audio files. It is also helpful to have means to annotate repositories as well. The eHumanities Desktop supports the creation of custom annotation schemes, called *Annotation Document Definitions* (or ADD for short). An ADD specifies attributes and their properties. Attributes have specific data types (i.e., string, integer, url, etc.), may be hierarchically ordered and can be set multiple times if required. Furthermore it is possible to define a value domain (like e.g. all possible entries for the attribute *Language of Resources*). See sections 3.2 and 3.3 for more details of the user perspective of the annotation system.

An annotation document definition is instantiated by an *Annotation Document*, that is, it contains valid values for all attributes defined according to the underlying ADD. An annotation document is tightly connected to the resource it annotates. Note that a resource can be annotated by different users and by multiple annotation documents. This offers means to, for example, let a group of researchers annotate resources and then aggregate their results according to an agreed specification. Note that annotation documents as well as annotation document definitions are resources and thus fall under the permission management system. That way a user can explicitly control who may read or write his or her annotations. This is done independently from the resource being annotated. Figure 4 shows an example of a resource being annotated independently by two different users using the same annotation

document definition. In this example the user 'Alice' has also given 'Clara' access permissions to her annotation.

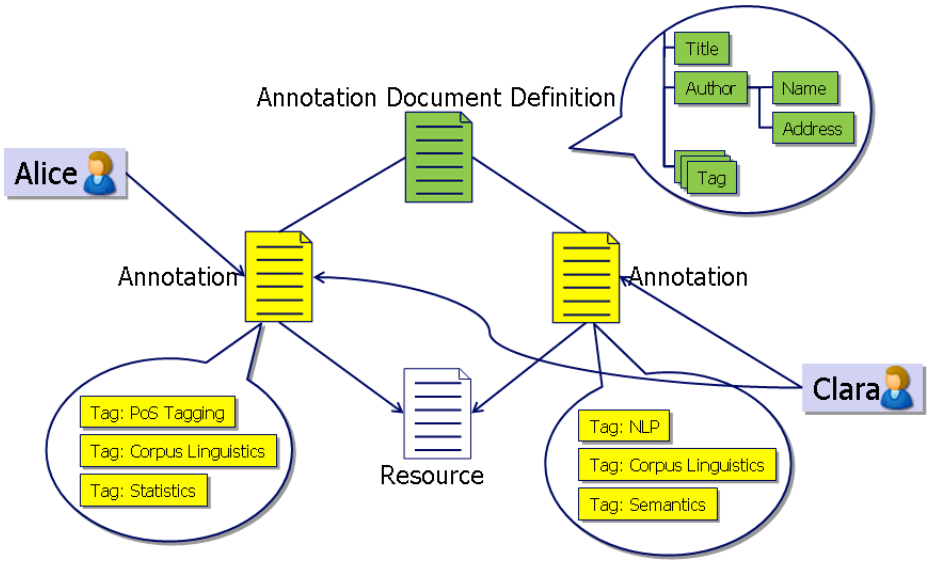

Figure 4: Example of multiple annotation of a resource by different users.

Finally *Features*, that is applications within the desktop, are considered resources. This approach enables the writers of applications to control who can actually use their works. This becomes interesting when legal issues are involved - for example if the use of a function requires a specific software licence. In such cases it would be problematic if all users had access. Having access to features under access permission control allows users to regulate who can use which function of the system.

2.2 Resource Representation and Management

The previous section briefly introduced the *Storage Handler* which we now explain in more detail. Its task is to organise the storage and retrieval of resources. The eHumanities Desktop is designed to handle virtually any kind of data ranging from text documents and XML over images and videos to audio files. Each format has a storage method which is optimized for fast storage and access: binary data is best saved as a file, XML in a native XML database to allow for XQueries, and relational database dumps can be inserted into a relational DBMS. The Storage Handler hides the way the data is being stored by offering common methods of access. Note that this approach would also allow for distributed data storage. Two other important aspects in this context are format conversion and representation of text documents.

Sadly it is more the rule than the exception that, despite the existence of standards to represent certain kinds of documents, a large variety of formats exist. This holds for representing images, videos, multi-modal annotations as well as for texts. The eHumanities Desktop accounts for this problem in two ways. Firstly it emphasises specific standard formats to improve the interoperability of tools. Secondly it offers a conversion matrix which allows for permanent or ad hoc conversions between formats. For example the text preprocessor needs raw text as input. But the desktop also allows you to perform preprocessing of HTML or PDF documents for example. This is realised by internally converting the documents before processing them. The conversion matrix relies on external tools to perform its tasks and is thus easily extensible (even without having to touch the source code).

Since linguistic applications are a focus of the eHumanities Desktop the question of how to represent texts is important. We rely on TEI P5 (cf. Burnard 2007), an XML based representation which is being developed and maintained by the Text Encoding Initiative. On the one hand it is expressive enough to accurately represent any sort of text while on the other hand it is flexible and light-weight because of its modular design. The ePreprocessor which is integrated into the desktop generates valid TEI P5 as output. A good example of how this representation is being used is the *Patrologia Latina*, a large corpus of Latin texts. This corpus has been converted into TEI P5 and tagged (Jussen, Mehler and Ernst 2007).

Of course an XML representation of texts alone is not very useful. What makes it interesting is the possibility to perform XQueries. But what works well on single documents can become awkwardly slow on large documents or entire collections. We have evaluated several native XML DBMS and chose Sedna XML to account for XML document storage. This open source system scales well on large collections, supports ACID transactions as well as XQuery.

3 Applications

Although the eHumanities Desktop puts emphasis on linguistic applications it can generally be adapted to other tasks as well. It simply offers a solid base on which applications can be built without having to bother with user and resource management. This section describes some of the application modules which are currently available to the user.

3.1 Corpus Manager

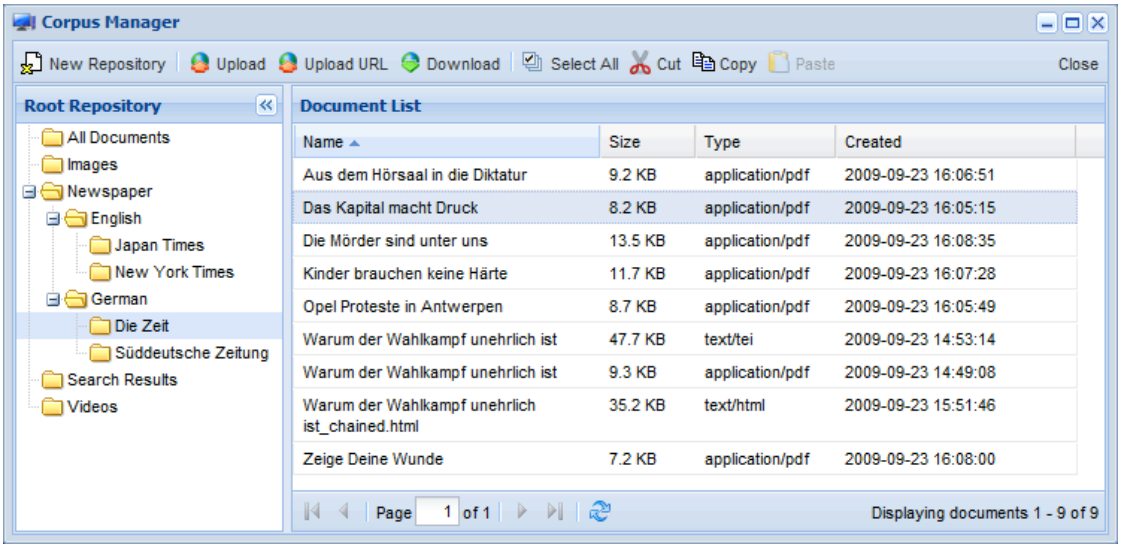

Figure 5: Screenshot showing the use of the corpus manager.

The corpus manager is so to speak the heart of the eHumanities Desktop. It offers means to upload, access and organise documents and repositories in the system. From the user's perspective the look and feel is quite close to the *Explorer* from the *MS Windows* operating system. Figure 5 shows a screenshot of the corpus manager. The left side shows the repository structure while the right side shows the documents of the currently selected repository. Repositories and documents can easily be organised by using drag and drop or cut/copy and paste functions. Documents can be uploaded from the local desktop or from the internet by specifying a resource URL. But the corpus manager also offers functions which go way beyond typical file managers. The list of documents can be filtered by various attributes which include wild cards and value delimiters. That way it is possible to, for example, quickly sort by document name, filtering all documents, for example, in which

names contain the string 'Experiment', are of a specific mime type, were last modified at most three days ago and are at least 2 MB in size. This works for a few documents as well as for thousands of documents in a repository. Of course it is also possible to search for documents over all repositories on the same basis. Searching for documents not only takes basic attributes like name, size, date of creation etc into account, but also attributes which stem from annotations the documents were annotated with. A search for documents can be composed successively by choosing attributes to be searched, the logical operator (e.g. less-than, equals or more) and a value. Figure 6 shows an example of how such a list of query terms can be composed. The results of a query are stored in a designated repository and can then be used for further processing. Other functions include ad hoc conversions of documents for downloading. For example one can download a PDF directly as plain-text. Some conversions may require additional parameters.

Search Attribute	Search Mode	Search Term
Title	like	Linguistics
Date of publication	>	2008-01-01
Price	<=	20.00
Keyword	like	syntax

Figure 6: Screenshot of the composition of a query for documents.

The corpus manager also offers means to share resources with other users. Access permissions can be granted explicitly to users or to groups. All users who are members of a given group also inherit the respective permissions. Table 1 explains the different levels of access on documents and repositories.

level	**document**	**repository**
read	general access and download	general access
write	update of content and meta data	adding and removing resources
delete	delete document	delete repository
grant	give permissions to others	give permissions to others

Table 1: Semantics of the different levels of access permission.

There is another difference between the corpus manager and a typical file manager which is not visible at first glance. The daily routine of researchers working on experimental data and linguistic corpora includes juggling with documents and finding means to organise them. In the eHumanities Desktop documents are basically not structured at all; that is, they do not specifically belong to some directory. Consequently the corpus manager contains a special repository called *All Documents*. On that basis documents can be part of an *arbitrary number* of repositories without the need for duplication. That way it is possible to, for example, create different perspectives on how to organise documents. One could organise a collection of newspaper articles via the publisher and also via the topic they deal with. Repositories can also be part of several other repositories. So strictly speaking the repository tree which is displayed in the corpus manager shows a tree-view of a *repository graph*. Another benefit of this approach is that a user who was given access to some repositories and documents from another one can choose to put the resources in a personal structure as he pleases without altering the structure of the granter.

Finally the corpus manager offers means to help synchronise concurrent work on the same resources. A user may decide to lock documents he or she is currently working on. A locked

document can only be viewed and edited by the user who locked it. This avoids accidentally overwriting each other's contributions.

The corpus manager also gives access to the resource annotation mechanisms which are explained in more detail in the following subsections.

3.2 Annotation Document Definition Editor

Usually a document when being used in experiments or corpus linguistic analysis needs some elementary meta data to be useful. In case of a newspaper article for example it would be important to annotate at least the language, the publisher, date and maybe some keywords or topics. Then this annotation needs to be stored in a way that it can effectively be searched in order to retrieve the associated document.

The eHumanities Desktop supports annotating documents and repositories by means of annotation documents which adhere to a specific annotation document definition. Even though there are certain standard annotation schemes like the Dublin Core[3], the system does not restrict the user to any predefined definitions. The user has rather the possibility to freely create his own annotation schemes and to alter them as necessary in the life system. That way, standard schemes *can* be used, but one is not restricted to that.

The annotation document definitions can directly be created and edited in the eHumanities Desktop environment. Figure 7 shows a screenshot of a typical working scenario: A hierarchy of attributes is shown and the properties of one attribute are currently being edited in a separate dialogue. Beside the name, description and data type it can be defined how many values can be set for that specific attribute. For example an attribute representing the title of a book should have exactly one value. In case of the keywords describing a book zero values up to a certain maximum should be permitted instead. This range can be explicitly defined.

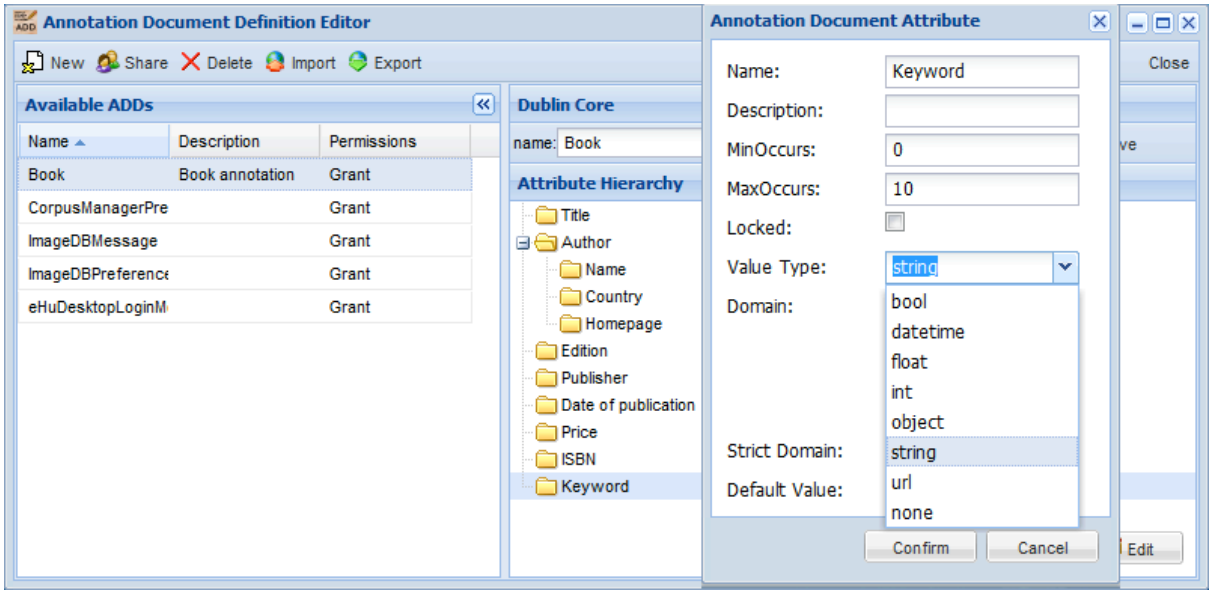

Figure 7: Screenshot showing the Annotation Document Definition Editor.

The so defined annotation document definitions can be exported (and re-imported) into an XML representation which conforms to the Annotation Document Definition Language[4].

3.3 Document Annotation

The last section introduced the basic concepts of annotation using the eHumanities Desktop. To actually perform an annotation the user selects a resource and picks an annotation document definition to create a new annotation document. If one (or more)

annotations already exist a user may decide to update an annotation. In both cases an annotation editor appears which gives access to the attributes and the values being set. Figure 8 shows an example of such an annotation.

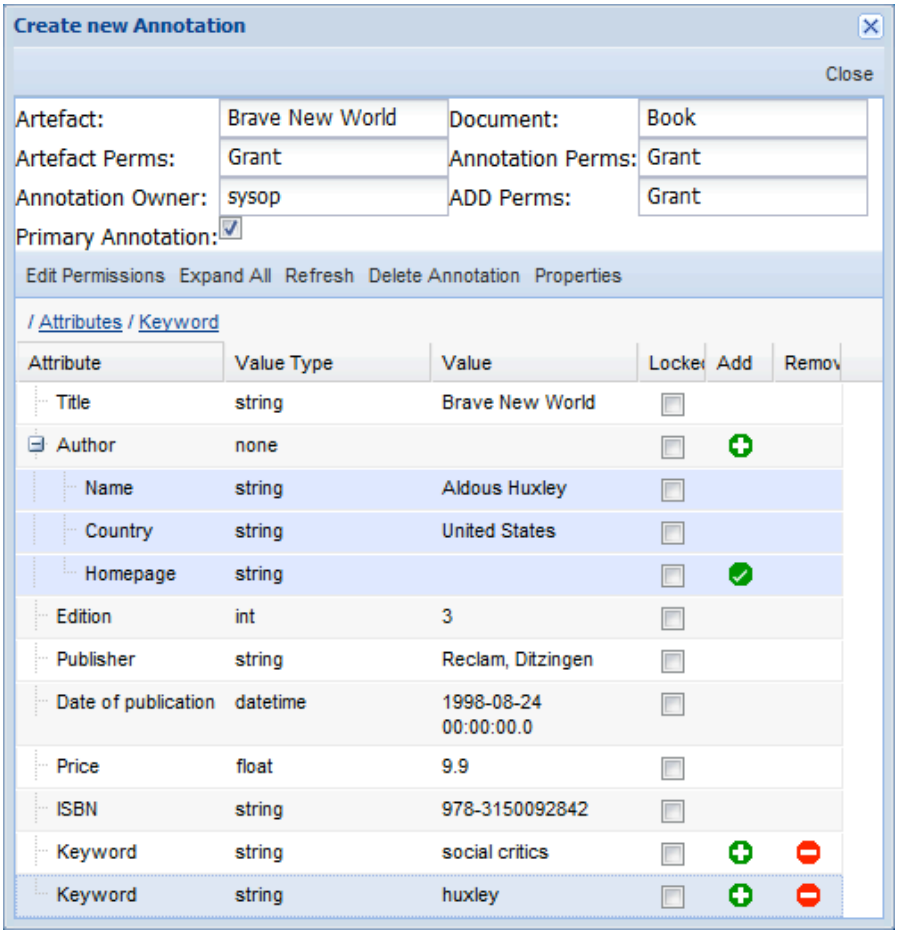

Figure 8: Screenshot showing the edit perspective of an Annotation Document.

An important aspect is that annotation documents can also be shared with other users. That way it is possible to grant read permissions to specific groups in order to let them search for documents based on selected annotations, or to grant write permissions so that others are also permitted to edit the annotations. In addition, the values of certain attributes can be locked so that they cannot be edited by others even if they haven write permission. This makes sense if for example some attribute like an inventory number are considered to be static, but other attribute like keywords or topics should still be editable.

3.4 Text Preprocessing

A common yet demanding task in natural language processing is the preprocessing of input texts and their accurate representation according to the needs of linguistic analysis. The *Preprocessor* module enables users to preprocess their uploaded documents. Non-text formats like PDF can automatically be converted by using the conversion matrix of the eHumanities Desktop. Besides language detection, sentence boundary detection, tokenisation, lemmatisation, stemming and name entity recognition, a trigram HMM-Tagger is implemented in the preprocessing system (Mehler et al. 2009).

The currently released implementation supports English, German and Latin. The tagging module was trained and evaluated based on the German Negra Corpus (Negra 2006) (F-score

of 0.975), 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). For the complete evaluation results see Table 2.

The output of the preprocessor can directly be displayed in the browser or saved as a new TEI P5 document in the Sedna XML database.

module	language	parameters	F-score	test corpus
PoS tagging	de	3,000 sentences	0.975	NEGRA corpus
	de-spoken	3,900 sentences	0.982	Tübinger spoken language corpus
	en	5,000 sentences	0.956	Penn Treebank
lemmatisation	de	888,573 word forms	0.921	NEGRA corpus
language identification	21 languages	50 characters	0.956	Wikipedia
	21 languages	100 characters	0.970	Wikipedia

Table 2: The table shows the evaluation results of the Preprocessor.

3.5 Lexical Chaining

As a further linguistic application module a lexical chainer (Mehler 2005:b, Mehler et al. 2007:a, Waltinger et al. 2008, Waltinger, Mehler and Stuehrenberg:2008) has been included. That is, semantically related tokens of a given text can be tracked and connected by means of a lexical reference system. The system currently uses two different terminological ontologies - *WordNet* (Fellbaum 1998) and *GermaNet* (Hamp and Feldweg 1997) - as chaining resources which have been mapped onto the database format. However the list of resources for chaining can easily be extended. The lexical chainer can generate an HTML output which offers intuitive means to examine the results.

3.6 Open Topic Classification

With the *Open Topic Classificator* an easy to use text classifier (Waltinger,Mehler and Heyer 2008) has been integrated into the system. It supports automatic mapping of an unknown text onto a social ontology (Waltinger and Mehler 2009d), currently Wikipedia. The system uses the category tree of the German and English Wikipedia-Project in order to assign category information to textual data.

3.7 Categorisation

When dozens of new text documents are uploaded into the system it would be very useful to have means for a first automatic annotation of the new resources. Such an annotation could stem from a defined text classificator which assigns category labels to each resource. Systems like GATE (Cunningham 2002) offer efficient means for flexible text categorisation and their integration will be an important aspect of future work. In current development a user interface to work with text classification mechanisms is offered by the *text classification* module of the eHumanities Desktop (Mehler 2009c, Waltinger, Mehler and Gleim 2009c). From the users perspective it allows one to either specify an input document which is already stored in the system, or directly insert a text via cut and paste. After choosing a classifier and computing

the categorisation, a table is displayed which shows all category labels and their significance regarding the classified document. Figure 9 shows the user interface of the Categoriser.

The architecture of the classification module is designed generically in a way that supports the configuration of different classification algorithms. Classifiers are defined using the *Classifier Description Language*, a light-weight XML-based language used to describe the preprocessing of input texts, their mapping onto a specific machine learning algorithm as well as its proper parametrisation. These preprocessing steps may include an automatic PoS tagging and lemmatisation using the preprocessor module of the eHumanities Desktop. The current release of the eHumanities Desktop offers DDC classifiers for English and German (Mehler and Waltinger 2009b) as well as a classifier based on the German newspaper *Süddeutsche Zeitung* using support vector machines (cf. Joachims 2002).

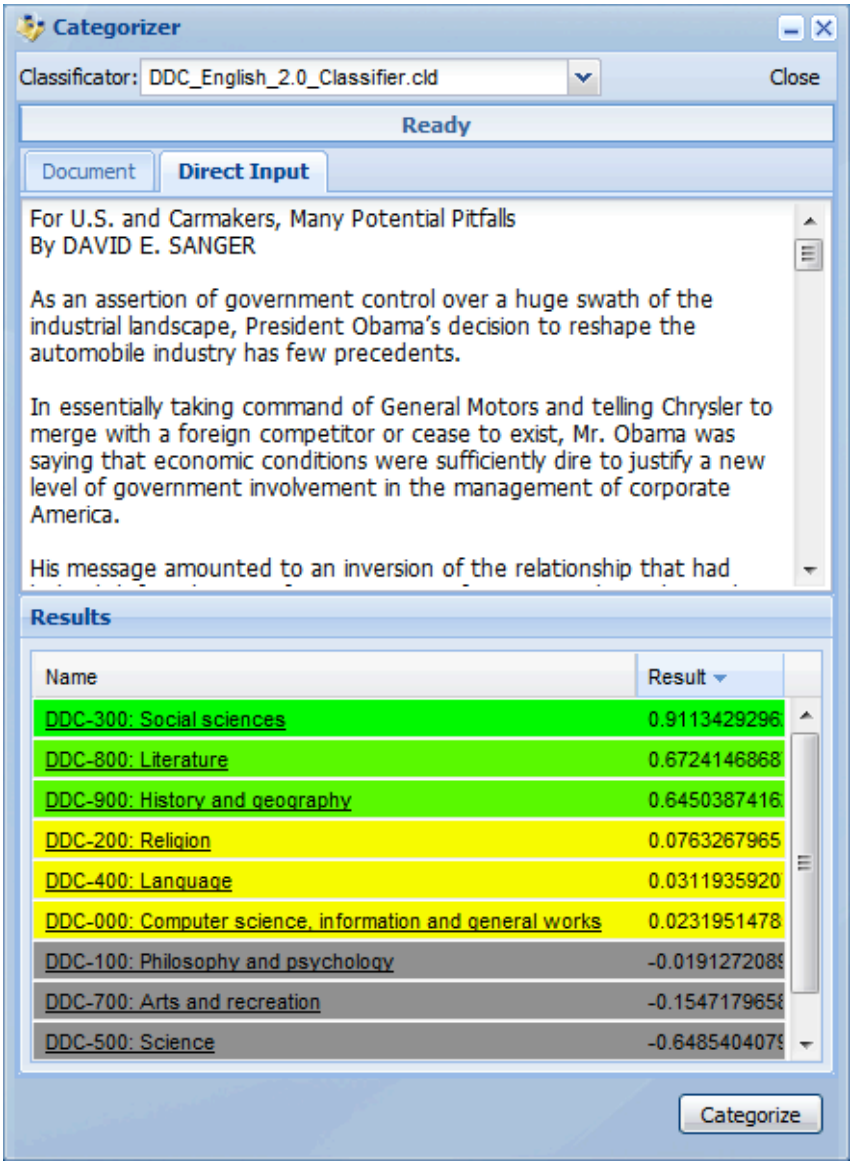

Figure 9: Screenshot of the Categoriser showing an exemplary categorisation of an input text.

3.8 Historical Semantics Corpus Management

The HSCM is developed by the research project *Historical Semantics Corpus Management* (Jussen, Mehler and Ernst 2007). The system aims at a text-technological representation and quantitative analysis of chronologically layered corpora. A prominent instance is the *Patrologia Latina*[5] which has been tagged and converted into the TEI P5

format. Since it is stored in a native XML DBMS the full power of XQuery can be used to browse and analyse the documents. The user interface offers an easy-to-learn query language to search for single terms or entire phrases. Besides browsing entire documents, a concordance mode is supported in order to examine the context of retrieved words or phrases. The HSCM is not restricted to the HSCM; users can generally add own documents as long as they adhere to the TEI P5 encoding.

4 Conclusion

We presented the eHumanities Desktop, a working environment for corpus management and analysis. The architecture of the system has been discussed as well as an excerpt of the application modules that are available. The system is an advancement of Ariadne[6] which has been developed in the DFG funded project SFB673/X1 *Multimodal alignment corpora: statistical modelling and information management*[7]. In contrast to Ariadne, the eHumanities Desktop targets a broader audience of researchers in the humanities. Users interested in using the system are welcome to contact us.

Acknowledgement

Financial support of the German Federal Ministry of Education (BMBF) through the research project *Linguistic Networks* and of the German Research Foundation (DFG) through the Excellence Cluster 277 *Cognitive Interaction Technology* (via the Project *Knowledge Enhanced Embodied Cognitive Interaction Technologies* (KnowCIT)), the SFB 673 *Alignment in Communication* (via the Project X1 *Multimodal Alignment Corpora: Statistical Modeling and Information Management*), and the LIS-Project *Content-Based P2P-Agents for Thematic Structuring and Search Optimization in Digital Libraries* all at Bielefeld University is gratefully acknowledged.

References

Burnard, L. (2007). *New tricks from an old dog: An overview of TEI P5*. In Burnard, L., Dobreva, M., Fuhr, N., and Lüdeling, A., editors, *Digital Historical Corpora-Architecture, Annotation, and Retrieval*, number 06491 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Chiarcos, Ch., Dipper, S., Götze, M., Leser, U., Lüdeling, A., Ritz, J., and Stede, M. (2008). *A flexible framework for integrating annotations from different tools and tagsets*. In: Traitement Automatique des Languages, 2008.

Cunningham, H. (2002). *Gate, a General Architecture for Text Engineering.* In Computing and the Humanities 36:223-254, 2002.

Dipper, S., Hinrichs, E., Schmidt, T., Wagner, A., and Witt, A. (2006). *Sustainability of linguistic resources*. In Hinrichs, E., Ide, N., Palmer, M., Pustejovsky, J. (Eds.), Proceedings of the LREC 2006 Workshop on Merging and Layering Linguistic Information. Genoa, Italy.

Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge.

Hamp, B. and Feldweg, H. (1997). *GermaNet - a lexical-semantic net for german*. In Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications, pages 9–15.

Ide, N. and Romary, L. (2004). *International standard for a linguistic annotation framework*. *Nat. Lang. Eng.*, 10(3-4):211–225.

Joachims, T. (2002). *Learning to classify text using support vector machines*. Kluwer, Boston.

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

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

Mehler, A. (2005*). Lexical chaining as a source of text chaining*. In Patrick, J. and Matthiessen, C., editors, Proceedings of the 1st Computational Systemic Functional Grammar Conference, University of Sydney, Australia, pages 12–21.

Mehler, A., Waltinger, U., and Wegner, A. (2007). *A formal text representation model based on lexical chaining*. In Proceedings of the KI 2007 Workshop on Learning from Non-Vectorial Data (LNVD 2007) September 10, Osnabrück, pages 17–26, Universität Osnabrück.

Mehler, A., Gleim, R., Ernst, A., Esch, D. and Feith, T. (2009). *eHumanities Desktop - eine webbasierte Arbeitsumgebung für die geisteswissenschaftliche Fachinformatik.* In Proc. of the Symposium Sprachtechnologie und eHumanities, 26. and 27. February, Duisburg-Essen University.

Mehler, A. and Waltinger, U. (2009b). Enhancing Document Modeling by Means of Open Topic Models: *Crossing the Frontier of Classification Schemes in Digital Libraries by Example of the DDC.* In Library Hi Tech 27(4).

Mehler, A. (2009c). *A Quantitative Graph Model of Social Ontologies by Example of Wikipedia*. In: Towards an Information Theory of Complex Networks: Statistical Methods and Applications. Dehmer, Emmert-Streib, Mehler (Eds.), Birkhäuser, Boston/Basel, 2009.

Rehm, G., Schonefeld, O., Witt, A., Hinrichs, E. and Reis, M. (2009). *Sustainability of Annotated Resources in Linguistics: A Web-Platform for Exploring, Querying and Distributing Linguistic Corpora and Other Resources*. In: Literary and Linguistic Computing, 24 (2), pp. 193–210, 2009.

Uszkoreit, H., Brants, T., Brants, S., and Foeldesi, C. (2006). *Negra corpus*. URL: http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/.

Váradi, T., Krauwer, S., Wittenburg, P., Wynne, M. and Koskenniemi, K. (2008). *CLARIN: Common Language Resources and Technology Infrastructure.* In: Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), European Language Resources Association (ELRA), Marrakech, Morocco, 2008.

Waltinger, U., Mehler, A., and Heyer, G. (2008a). *Towards automatic content tagging: Enhanced web services in digital libraries using lexical chaining*. In 4th Int. Conf. on Web Information Systems and Technologies (WEBIST '08), 4-7 May, Funchal, Portugal. Barcelona.

Waltinger, U., Mehler, A., and Stührenberg, M. (2008b). *An integrated model of lexical chaining: Application, resources and its format*. In Storrer, A., Geyken, A., Siebert, A., and Würzner, K.-M., editors, Proceedings of KONVENS 2008 — Ergänzungsband Textressourcen und lexikalisches Wissen, pages 59–70.

Waltinger, U. and Mehler, A., Gleim, R. (2009c). *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.

Waltinger, U. and Mehler, A. (2009d). *Social Semantics And Its Evaluation By Means Of Semantic Relatedness And Open Topic Models*. In Proceedings of the 2009 IEEE/WIC/ACM International Conference on Web Intelligence.

---

[1] http://hudesktop.hucompute.org
[2] http://tomcat.apache.org
[3] http://dublincore.org
[4] http://dtd.hucompute.org/add.xsd
[5] http://pld.chadwyck.co.uk
[6] http://ariadne-cms.eu
[7] http://www.sfb673.org/projects/X1/