eHumanities Desktop - An Online System for Corpus Management and Analysis in Support of Computing in the Humanities

Rüdiger Gleim[1], **Ulli Waltinger**[2], **Alexandra Ernst**[2], **Alexander Mehler**[1],
Tobias Feith[2] & Dietmar Esch[2]

[1]Goethe-Universität Frankfurt am Main, [2]Universität Bielefeld

Abstract

This paper introduces *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 initial set of applications which are offered by the system.

1 Introduction

Since there is an ongoing shift towards computer based studies in the humanities new challenges in maintaining and analysing electronic resources arise. This is all the more because research groups are often distributed over several institutes and universities. Thus, the ability to collaboratively work on shared resources becomes an important issue. This aspect also marks a turn point in the development of Corpus Management Systems (CMS). Apart from the aspect of pure resource management, processing and analysis of documents have traditionally been the domain of desktop applications. Sometimes even to the point of command line tools. Therefore the technical skills needed to use for example linguistic tools have effectively constrained their usage by a larger community. We emphasise the approach to offer low-threshold access to both corpus management as well as processing and analysis in order to address a broader public in the humanities.

The *eHumanities Desktop*[1] is designed as a general purpose platform for scientists in humanities. Based on a sophisticated data model to manage authorities, resources and their interrelations the system offers an extensible set of application modules to process and analyse data. Users do not need to undertake any installation efforts but simply can login from any computer with internet connection using a standard browser. Figure 1 shows the desktop with the *Document Manager* and the *Administration Dialog* opened.

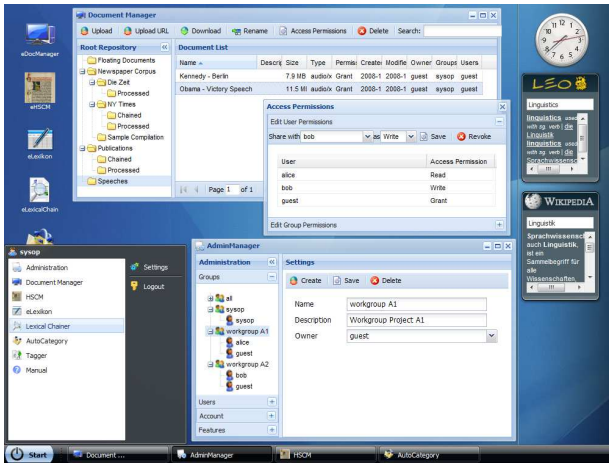

Figure 1: The eHumanities Desktop environment showing the document manager and administration dialog.

In the following we describe the general architecture of the system. The second part addresses an initial set of application modules which are currently available through *eHumanities Desktop*. The last section summarises the system description and gives a prospect of future work.

2 System Architecture

Figure 2 gives an overview of the general architecture. The *eHumanities Desktop* is implemented as a client/server system which can be used via any JavaScript/Java capable Web Browser. The GUI is based on the ExtJS Framework[2] and provides a look and feel similar to Windows Vista. The server side is based on Java Servlet technology using the Tomcat[3] Servlet Container. The core of the system is the *Command Dispatcher* which

[1] http://hudesktop.hucompute.org

[2] http://extjs.com

[3] http://tomcat.apache.org

manages the communication with the client and the execution of tasks like downloading a document for example. The *Master Data* include information about all objects managed by the system, for example users, groups, documents, resources and their interrelations. All this information is stored in a transactional Relational Database (using MySQL[4]). The underlying data model is described later in more detail. Another important component is the *Storage Handler*: Based on an automatic mime type[5] detection it decides how to store and retrieve documents. For example videos and audio material are best stored as files whereas XML documents are better accessible via a XML Database Management System or specialized DBMS (e.g. HyGraphDB (Gleim et al., 2007)). Which kind of *Storage Backend* is used to archive a given document is transparent to the user- and also to developers using the *Storage Handler*. The *Document Indexer* allows for structure sensitive indexing of text documents. That way a full text search can be realised. However this feature is not fully integrated at the moment and thus subject of future work. Finally the *Command Dispatcher* connects to an extensible set of application modules which allow to process and analyse stored documents. These are briefly introduced in the next section.

To get a better idea of how the described components work together we give an example of how the task to perform PoS tagging on a text document is accomplished: The task to process a specific document is sent from the client to the server. As a first step the *Command Dispatcher* checks based on the *Master Data* if the requesting user is logged in correctly, authorized to perform PoS tagging and has permission to read the document to be tagged. The next step is to fetch the document from the *Storage Handler* as input to the *PoS Tagger* application module. The tagger creates a new document which is handed over to the *Storage Handler* which decides how to store the resource. Since the output of the tagger is a XML document it is stored as a XML Database. Finally the information about the new document is stored in the *Master Data* including a reference to the original one in order to state from which document it has been derived. That way it is possible to track on which basis a given document has been created. Finally the *Command Dispatcher* signals the successful completion of the task back to the *Client*.

Figure 3 shows the class diagram of the master data model. The design is woven around the general concept that *authorities* have access permissions on *resources*. Authorities are distinguished into *users* and *groups*. Users can be members of one or more groups. Furthermore authorities can have permissions to use *features* of the system. That way it is possible to individually configure the spectrum of functions someone can effectively use. Resources are distinguished by *documents* and *repositories*. Repositories are containers, similar to directories known from file systems. An important addition is that resources can be member of an arbitrary number of repositories. That way a document or a repository can be used in different contexts allowing for easy corpus compilation.

A typical scenario which benefits from such a data model is a distributed research group consisting of several research teams: One team collects data from field research, a second processes and annotates the raw data and a third team performs statistical analysis. In this example every group has the need to share resources with others while keeping control over the data: The statistics team should be able to read the annotated data but must not be allowed to edit resources and so on.

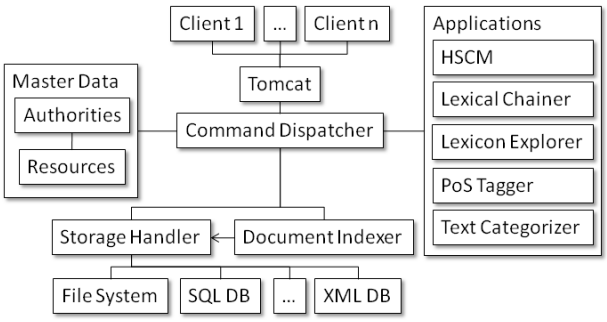

Figure 2: Overview of the System Architecture.

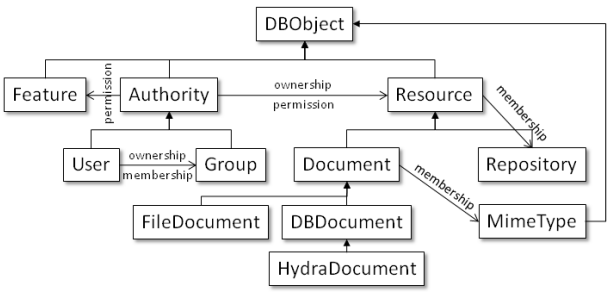

Figure 3: UML Class Diagram of the Master Data.

[4] http://dev.mysql.com

[5] http://www.iana.org/assignments/media-types/

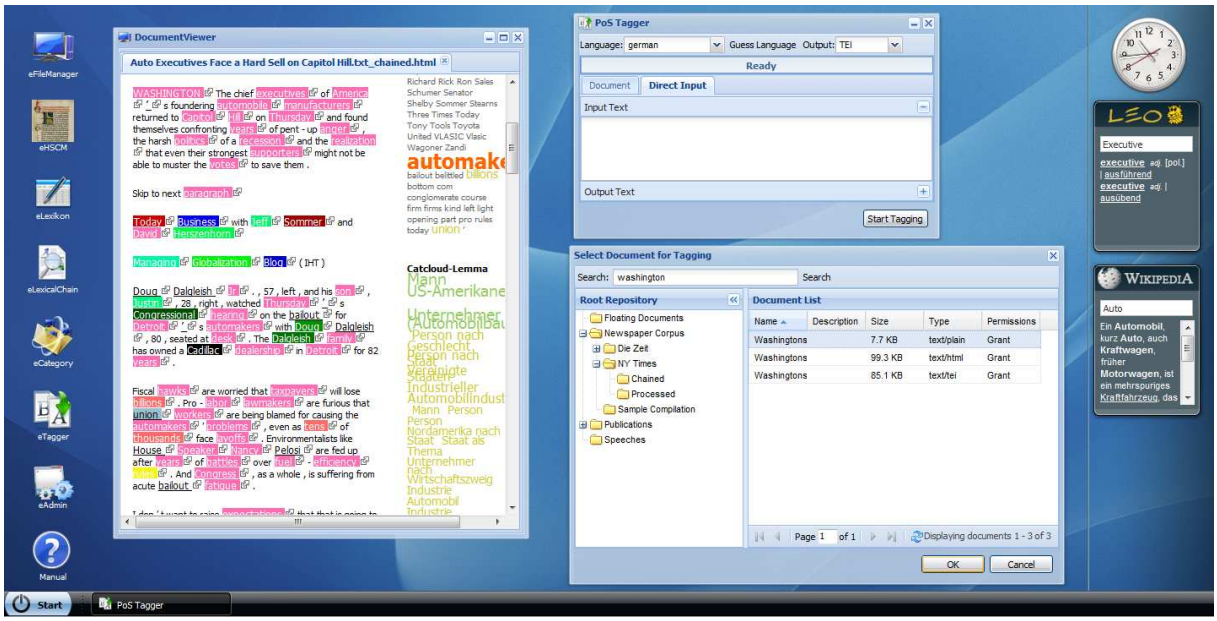

Figure 4: The eHumanities Desktop environment showing a chained document and the PoS Tagger dialog.

3 Applications

In the following we outline the initial set of applications which is currently available via *eHumanities Desktop*. Figure 4 gives an idea of the look and feel of the system. It shows the visualisation of a chained document and the PoS Tagger window with an opened document selection dialog.

3.1 Document Manager

The *Document Manager* is the core of the desktop. It allows to upload and download documents as well as sharing them with other users and groups. It follows the look and feel of the *Windows Explorer*. Documents and repositories can be created and edited via context menus. They can be moved via drag and drop between different repositories. Both can be copied via drag and drop while pressing the Ctrl-key. Note that repositories only contain references- so a copy is *not* a physical reduplication. Documents which are not assigned to any repository the current user can see are gathered in a special repository called *Floating Documents*. A double click on a file will open a document viewer which offers a rendered view of textual contents. The button 'Access Permissions' opens a dialog which allows to edit the rights of other users and groups on the currently selected resources. Finally a search dialog at the top makes documents searchable.

3.2 PoS Tagging

The PoS-Tagging module enables users to preprocess their uploaded documents. Besides tokenisation and sentence boundary detection, a trigram HMM-Tagger is implemented in the preprocessing system (Waltinger and Mehler, 2009). The tagging module was trained and evaluated based on the German Negra Corpus (Uszkoreit et al., 2006) (F-measure of 0.96) and the English Penn Treebank (Marcus et al., 1994) (F-measure of 0.956). Additionally a lemmatisation and stemming module is included for both languages. As an unifying exchange format the component utilises TEI P5 (Burnard, 2007).

3.3 Lexical Chaining

As a further linguistic application module a lexical chainer (Mehler, 2005; Mehler et al., 2007; Waltinger et al., 2008a; Waltinger et al., 2008b) has been included in the online desktop environment. 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.

3.4 Lexicon Exploration

With regards to lexicon exploration, the system aggregates different lexical resources including English, German and Latin. In this module, not only co-occurrence data, social and terminological ontologies but also social tagging enhanced data are available for a given input token.

3.5 Text Classification

An easy to use text classifier (Waltinger et al., 2008a) has been implemented into the system. In this, an automatic mapping of an unknown text onto a social ontology is enabled. The system uses the category tree of the German and English *Wikipedia-Project* in order to assign category information to textual data.

3.6 Historical Semantics Corpus Management

The HSCM is developed by the research project *Historical Semantics Corpus Management* (Jussen et al., 2007). The system aims at a texttechnological representation and quantitative analysis of chronologically layered corpora. It is possible to query for single terms or entire phrases. The contents can be accessed as rendered HTML as well as TEI P5[6] encoded. In its current state is supports to browse and analyse the *Patrologia Latina*[7].

4 Conclusion

This paper introduced *eHumanities Desktop*- a web based corpus management system which offers an extensible set of application modules which allow online exploration, processing and analysis of resources in humanities. The use of the system was exemplified by describing the Document Manager, PoS Tagging, Lexical Chaining, Lexicon Exploration, Text Classification and Historical Semantics Corpus Management. Future work will include flexible XML indexing and queries as well as full text search on documents. Furthermore the set of applications will be gradually extended.

References

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, Dagstuhl, Germany. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

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

Rüdiger Gleim, Alexander Mehler, and Hans-Jürgen Eikmeyer. 2007. Representing and maintaining large corpora. In *Proceedings of the Corpus Linguistics 2007 Conference, Birmingham (UK)*.

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

Bernhard Jussen, Alexander Mehler, and Alexandra Ernst. 2007. A corpus management system for historical semantics. *Appears in: Sprache und Datenverarbeitung*.

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, Ulli Waltinger, and Armin Wegner. 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, Osnabrück. Universität Osnabrück.

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

Hans Uszkoreit, Thorsten Brants, Sabine Brants, and Christine Foeldesi. 2006. Negra corpus.

Ulli Waltinger and Alexander Mehler. 2009. Web as preprocessed corpus: Building large annotated corpora from heterogeneous web document data. In preparation.

Ulli Waltinger, Alexander Mehler, and Gerhard Heyer. 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.

Ulli Waltinger, Alexander Mehler, and Maik Stührenberg. 2008b. An integrated model of lexical chaining: Application, resources and its format. In Angelika Storrer, Alexander Geyken, Alexander Siebert, and Kay-Michael Würzner, editors, *Proceedings of KONVENS 2008 — Ergänzungsband Textressourcen und lexikalisches Wissen*, pages 59–70.

[6] http://www.tei-c.org/Guidelines/P5

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