DIGITAL KNOWLEDGE STORE

Technical Reader
OVERVIEW

The Digital Knowledge Store of the Academy is a software infrastructure, which is designed as a typical client-server architecture. It aims to store all metadata and full texts of all digital resources of the Berlin-Brandenburg Academy of Sciences and Humanities and provide a centralized access to them.

Apache / Django Framework (Python)

- Servlet Module
  - communicates with GUI and index, e.g. processes queries; initializes indexing & updating
- Libraries
  - Statistics, Scoring System, Text Mining
  - **Java Core Modul**
    - Indexing, Parsing, Harvesting, Translation, Morphological Analysis
  - External Libraries
    - e.g. mpiwg, quartz
- Solr Server
  - Lucene Document and Node Indexes, Metadata Index

Apache Tomcat (Java Servlet Container)

- BBAW Resources
  - heterogeneous formats
    - e.g. PDF, TEI, SQL, JPG, DOC ...

HTML5, CSS, JavaScript

Queries via HTTP

Responds with JSON/XML-encoded results

Responds with

Extracts via HTTP

Administrates and queries the index via Lucene API
EXTRACTING & PARSING

As one concept of the Knowledge Store is to avoid redundancy of data, a harvesting software was developed which collects the URLs of the resource collections on different peripheral servers. So far 136 collections have been registered, which include hundreds of thousands of resources. At this early stage of the project the index contains 2050 documents from nine BBAW-collections. The resources are imported recursively into a central Lucene index. This is realized by XML-parsers based on Saxon, which are able to tokenize the full text with regard to contents, as well as analyze the full text morphologically and add it to the index. Various approaches to handle the metadata are currently evaluated. In the XML full texts, morphological analyzed page fragments are created, which contain links to different dictionaries. This is implemented providing for high query performance.

APACHE LUCENE

A web interface offers a way to query all the data stored in the index via the Lucene query language in full Lucene complexity. This covers the term- and phrase query, query in index fields, boolean query, wildcard query, fuzzy- and proximity query, and combinations of these queries. The search results of a query across all documents in the index contain – among other information – a link to the original document and one or more fragments from the resulting document to indicate the context of the term. In addition the query term is highlighted in the fragment. The prototype presented at the conference distinguishes between the query in the metadata, query in the full text, and morphological query in the full text.

Furthermore, the Knowledge Store offers the possibility to choose a document and then search for more documents with similar content. This service uses the frequency of the terms in a document to determine the content relevancy of a term in a document. As documents are assumed to be similar when the most relevant terms are similar, term vectors containing the most relevant terms are created for every document and compared to each other.

MORPHOLOGY

The basis of the morphological analysis of the full texts is the DONATUS software developed at the Max-Planck-Institut für Wissenschaftsgeschichte (MPIWG) in Berlin. It is capable of analyzing the languages Latin, ancient Greek, Arabic, Italian, German, English, French, and Dutch. In order to analyze ancient Greek, it was necessary to convert the betacode entries of an ancient Greek dictionary into unicode. In combination with the Lucene system the morphological component of the MPIWG establishes a morphological index. When the user queries the full text, the query term is reduced to its basic form and queried against the morphological index automatically. Again, the query term and its word form of the basic form are highlighted in
the search result. Further cooperation with the MPIWG is in planning. In the future we want to enable the search for similar content on the basis of morphologically analyzed terms.

TRANSLATION

A translation service is implemented which provides the option to automatically translate the query term or its lemmatized base form into the languages English, German, and French. Thus, the amount of results is extended to documents of these languages. This is a first approach which will be expanded in the future. The translation service also helps to detect the language of a document during the process of indexing if no corresponding metadata is available. At the moment the language detection only works for modern languages. The planned extension of the translation service will enable the search for similar content across language boundaries, e.g. having a base of German texts and finding similar texts in ancient Greek.

USER INTERFACE

The current user interface is implemented using the Python-based Django-Frame-work. Request URLs are used for querying the separated Apache Tomcat server based on user input. Responses from the server are encoded in JSON and contain the full search results and corresponding further links. Using the approach of isolated services for front end and backend, communicating with JSON enables greater flexibility in experimenting with new UI approaches and ensures platform independence. The user facing web pages are rendered by Django templates and are written as standard-oriented HTML5 with CSS and JavaScript (mainly using the JQuery library). At the moment the UI is implemented as a fairly conventional search interface, however plans are being made to experiment with new visualization techniques to facilitate new approaches of ‘knowledge browsing’.

Technical resources
http://tomcat.apache.org/
http://saxon.sourceforge.net/
http://lucene.apache.org/core/
http://www.djangoproject.com/
http://www.mpiwg-berlin.mpg.de/de/index.html

Further information
wsp@bbaw.de
http://wsp.bbaw.de