A Semantic Wiki Approach to Cultural Heritage Data Management

René Witte, Thomas Gitzinger, Thomas Kappler, Ralf Krestel

Institut für Programmstrukturen und Datenorganisation (IPD)
Universität Karlsruhe (TH), Germany

Abstract

Providing access to cultural heritage data beyond book digitization and information retrieval projects is important for delivering advanced semantic support to end users, in order to address their specific needs. We introduce a *separation of concerns* for heritage data management by explicitly defining different *user groups* and analyzing their particular requirements. Based on this analysis, we developed a comprehensive system architecture for accessing, annotating, and querying textual historic data. Novel features are the deployment of a Wiki user interface, natural language processing services for end users, metadata generation in OWL ontology format, SPARQL queries on textual data, and the integration of external clients through Web Services. We illustrate these ideas with the management of a historic encyclopedia of architecture.

1. Introduction

The amount of publicly available knowledge increases faster than we can imagine—hence the term "Information Explosion" used by several authors (Lyman and Varian, 2003; Sweeney, 2001). With the barrage of newly created content—news, blogs, web pages, research papers—existing, "analog" documents and their users often receive less attention than the quality of the content deserves.

In this paper, we present the results from a project aimed at developing enhanced semantic support for users of textual cultural heritage data. A particular feature of our approach is the integration of different concerns into a single, cohesive system architecture that addresses requirements from end users, software engineering aspects, and knowledge discovery paradigms. The ideas were implemented and tested with a historic encyclopedia of architecture and a number of different user groups, including building historians, architects, and NLP system developers.

2. User Groups and Requirements

Nowadays, the baseline for cultural heritage data management of book-type publications is the production of a scanned (digitized) version that can be viewed and distributed online, typically with some kind of Web interface. Before we can deliver more advanced access methods, we have to be more precise about the targeted end users. Who needs access to heritage data, and for what purpose?

2.1. User Groups

Within our project, we had to consider the requirements from four different user groups; each of them having a different background and expectations concerning the management of historic textual data.

(1) Historians: Within this group, we target users that deal with historic material from a scientific motivation, namely, historians. They require an electronic presentation that provides for a direct mapping to the printed original, e.g., for citation purposes. Additionally, semantic analysis tools should support their work through the formulation and verification of hypotheses.

(2) Practitioners: Under this group, we are concerned with users that need access to the historic material for their contemporary work. In our example scenario, the handbook on architecture, these are today's architects that need information on the building processes and materials used, e.g., within a restoration project of an old building. Here, the historic material contains knowledge that is not readily accessible in modern sources. Another example for such a user group are musicians dealing with old music scores and their descriptions, or lexicographers analyzing documents for the development of dictionary entries.

(3) Laypersons: Historic materials are a fascinating source of knowledge, as they preserve information over centuries. Providing widespread online access to materials that are otherwise only available in a controlled environment to scientists due to their fragile nature is perhaps one of the greatest benefits of digitization projects.

(4) Computational Linguists: Similarly to practitioners, linguists are often interested in historic documents from a functional point of view. However, their domain focuses on the properties of the language and its development over time rather than the underlying domain of discourse. They also have particular requirements for corpus construction, access, and annotation to support automated NLP analysis workflows.

2.2. Detected Requirements

We can now derive a number of requirements a system needs to fulfill, based on the user groups defined above:

Web Interface. To make the historic data available over the Internet, and to provide easy access within a familiar metaphor, the system needs to support a Web interface. This concerns all user groups to various degrees, but in particular the historians and laypersons.

Annotation Support. Users working with the historic data from a scientific point of view—in particular group (1)—often need to comment, add, and collaborate on the historic data. This should be supported within the same interface as the primary (historic) data, to avoid unnecessary context and application switches for the end users. At the same time, these annotations must be maintained by the architecture on clearly separated layers, to keep the integrity of the historic data intact.

Corpus Generation. While a Web interface is helpful for a human user, automated analyses using NLP tools and frameworks (user group (4)) can be better supported with a corpus in a standard (XML-based) markup, since HTML pages generated through Web frameworks typically mix content and layout information (menus, navigation bars, etc.). Thus, the architecture should provide a separate corpus that is automatically derived from the historic data and contains appropriate markup (for headlines, footnotes, figure captions, etc.). Ideally, it should allow to cross-link entities with the Web interface.

NLP Services. For large collections of (historic) documents, manual inspection of all content or even a subset obtained through information retrieval (IR) is not feasible. Here, NLP analyses can deliver additional benefit to end users, in particular groups (1)–(3), by integrating NLP analysis services (and their results) into the overall architecture. It should allow the execution of any service, developed by user group (4), and also deliver the results back to the clients. Examples for such NLP services are summarization, index generation, or named entity detection.

Metadata Generation. While NLP results can be useful for a human user, we also need to support further automated analysis workflows. User group (2) in particular requires access to the historic data, as well as its metadata, from external tools and applications relevant for their domain. To support external access to metadata from many different clients, the architecture should be capable of generating standards-compliant data formats, such as RDF and OWL.

Application Integration. As pointed out in the last requirement, external applications should be provided with automated access to the historic data and its metadata. Generally speaking, this requires the introduction of a client/server model, where the communication, like the metadata format, should use open, established standards.

3. Related Work

Before we describe our approach in detail, we discuss related work relevant for the detected requirements.

The Cultural Heritage Language Technologies (CHLT) project (Rydberg-Cox, 2002; Rydberg-Cox, 2005) describes the use of NLP methods to help students and scholars to work with classic Greek and Latin corpora. Similar to our approach, collaboration is an important goal of the project. Not only for sharing metadata about the text itself, but also to offer users the possibility to annotate, comment, or correct the results of automated analysis. This metadata can also contain hyperlinks to connect related texts with each other. The importance of correct morphological analysis is stressed as a baseline technology for users in the humanities, a statement which is also reflected in our work by integrating a self-learning lemmatizer for the German language (Perera and Witte, 2005) for accurate index generation. Further processing in the CHLT project includes information retrieval and data visualization. Identifying keywords, clustering subsets of the data, and visualizing the resulting groups supports the users in grasping concepts or performing search. In contrast, our approach uses open, standardized data formats like an automatically populated ontology to facilitate searching and browsing through the corpus and a Wiki system to share information between users.

As outlined in (Mavrikas et al., 2004), access to cultural heritage data available in natural language can be facilitated using various NLP techniques. In the context of the Semantic Web, the proposed system extracts CH data from different sources in the Internet and processes the data afterwards. An ontology (Doerr, 2003) is used to organize the mined data. Templates are used to extract relevant information, and the use of multi-document summarization is also proposed, as a way to present relevant information in a condensed way to the user. Here, we present an actual implementation of a system addressing these problems and extend the use of ontologies to allow easy browsing and querying of the document content for different user groups.

Another approach based on the CIDOC-CRM[1] ontology is presented in (Généreux, 2007). The system described there consists of two parts, one for extracting CH knowledge from natural language texts and saving the information in the ontology format, and one for using natural language to query the database. The natural language is reformatted to a SPARQL query using WordNet. This approach, in contrast to our system, stresses more the search aspect to find relevant data and offers no further possibilities for collaboration or processing of the data.

In (Sinclair et al., 2005), a system is presented that enables the user to explore, navigate, link, and annotate digitized cultural heritage artifacts like videos, photos, or documents. The system also supports user-generated descriptions and content. The focus in this project lies on the integration of the different metadata formats of the source content, whereas we additionally focus on the processing and collaboration part.

From a technical perspective, semantic extensions to Wiki systems based on *Semantic Web* technologies like OWL ontologies and RDF are similar in that they provide the means for content structuring beyond the syntactical level. In these systems, the properties of and relations between objects can be made explicit, with the Wiki system "knowing" about them. This allows for automated processing of Wiki content, e.g., through software agents. Current implementations of these ideas can be found in systems like Semantic MediaWiki (SMW) (Krötzsch et al., 2006) or IkeWiki (Schaffert, 2006). It is important to note that these tools are different from and complementary to our approach: While in our context, the content of a Wiki is subject to semantic analysis via NLP methods (with the Wiki engine itself not needing to have semantic capabilities), semantic Wikis like SMW have explicit notational and internal semantic capabilities. Using a semantic Wiki in our system in the future would allow the Wiki engine itself direct access to the facts derived from semantic text analysis.

4. Semantic Heritage Data Management

In this section, we present our approach to cultural heritage data management, which integrates a number of different technologies in order to satisfy the requirements of the various user groups: (i) A Wiki user interface, (ii) text mining support using an NLP framework, (iii) Semantic Web ontologies based on OWL and RDF for metadata management,

[1]CIDOC Conceptual Reference Model, http://cidoc.ics.forth.gr/

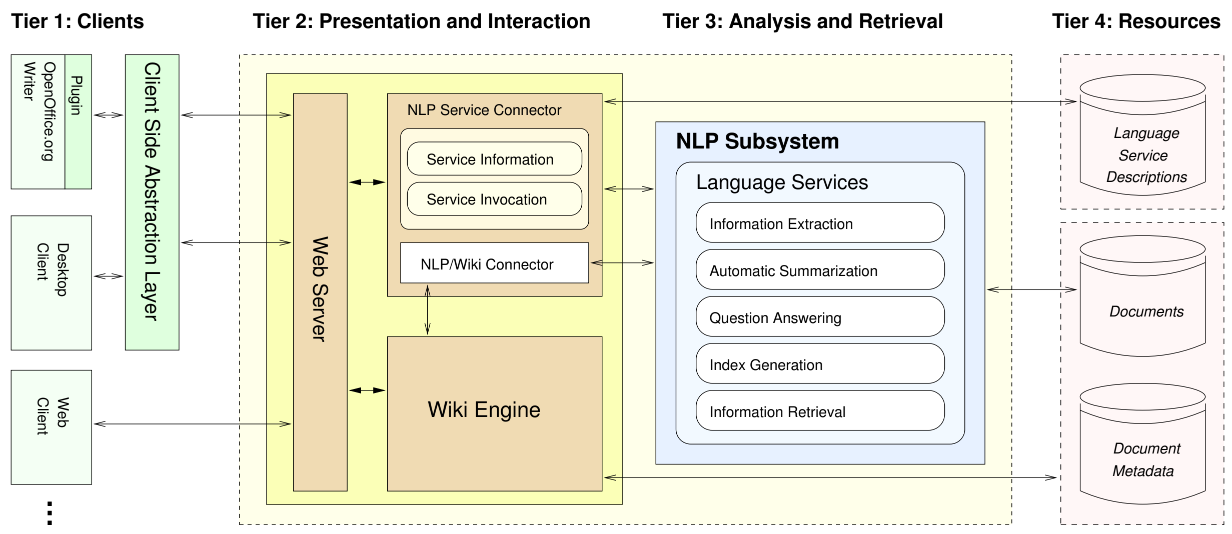


Figure 1: System architecture overview

and (iv) W3C Web Services for application integration. We first present an overview of our system in the next subsection. The various subsystems are illustrated using examples from a productive, freely accessible[2] Web resource built around the German *Handbuch der Architektur* (handbook on architecture) from the 19th century, described in detail in Section 4.2. The digitization process is described in Section 4.3. Necessary format conversions for the digital version are covered in Section 4.4. To support our user groups, we integrated several NLP analysis services, which are covered in Section 4.5. Finally, our semantic extensions for generating OWL/RDF metadata and application integration are covered in Section 4.6.

4.1. Architectural Overview

As stated above, our goal is the development of a unified architecture that fulfills the requirements (Section 2.2.) of the different user groups defined in Section 2.1., by integrating means for content access, analysis, and annotation.
One of the central pieces of our architecture is the introduction of a *Wiki* system (Leuf and Cunningham, 2001). Wiki systems provide the Web interface stipulated in our first requirement, while also allowing users to add meta-content in form of separate *discussion* or *annotation* pages. This capability directly addresses our second requirement, by allowing users to discuss and collaborate on heritage data, using an online tool and a single interface, while keeping the original data intact.[3]
Other clients, NLP services, and the actual content have to be integrated into this model. Figure 1 shows how these and the remaining components are systematically assembled to form the overall architecture of our system.
The architecture comprises four tiers. Tier 1 consists of clients that the users employ to access the system. Plug-in capable existing clients, like the OpenOffice.org application suite, can also be extended to be integrated with our architecture. New applications can have that functionality built in, like the "Desktop Client" depicted in the diagram. The "Client-Side Abstraction Layer" (CSAL) facilitates connecting clients by providing common communication and data converting functionality.
The clients communicate with a Web server on Tier 2, behind which we find the Wiki engine and a software module labeled "NLP Service Connector." The functionality of this module is offered as an XML Web service, as standardized by the W3C.[4] This means that there is a publicly accessible interface definition, written in the Web Service Description Language (WSDL), from which clients know how to use the offered functionality. The functionality itself is used through a Web service *endpoint*, to which the client sends and from where it receives messages. The main task of the NLP Service Connector is to receive input documents and have the NLP subsystem (Tier 3) perform various text analysis procedures on them. A sub-module of the NLP Service Connector, labeled "NLP/Wiki Connector," allows for the automatic retrieval, creation, and modification of Wiki content.
Finally, on Tier 4, we have metadata on the employed text analysis services (top), which the NLP Service Connector requires in order to operate these services. The bottom rectangle contains the documents maintained by the Wiki system as well as their metadata, which might have been provided by hand, or generated through automatic analysis methods.

4.2. Source Material

We implemented and evaluated the ideas described here for a particular set of historic documents: the German *Handbuch der Architektur*, a comprehensive multi-volume encyclopedia of architecture.[5] The full encyclopedia was written between the late 19th and early 20th century; It aimed to include all architectural knowledge at the time, both past and present, within the fields of architectural history, architectural styles, construction, statics, building equipment, physics, design, building conception, and town planning. The full encyclopedia comprises more than 140 individual

[2]See http://durm.semanticsoftware.info
[3]Assuming the Wiki has been properly configured for this scenario; the technical details depend on the concrete Wiki system.
[4]Web Services Architecture, http://www.w3.org/TR/ws-arch/
[5]Edited by Joseph Durm (⋆14.2.1837 Karlsruhe, Germany, †3.4.1919 ibidem) and three other architects since 1881.

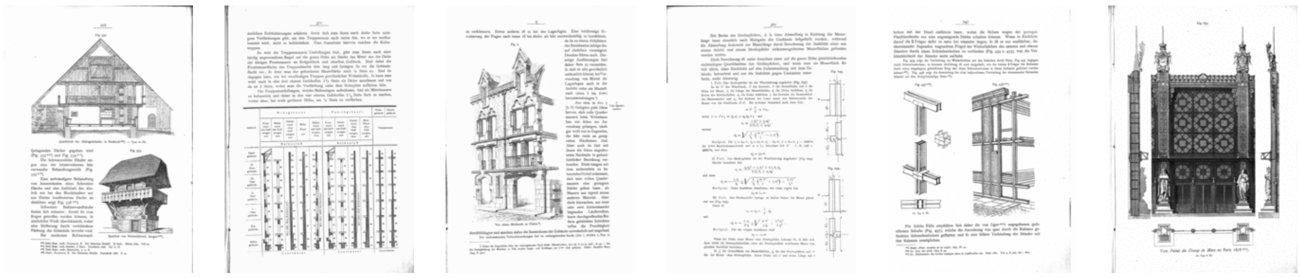


Figure 2: Source material examples: Scanned pages from *Handbuch der Architektur* (1900)

publications and contains at least 25 000 pages.

Due to the ambitious scope, the long publication process, and the limitations of the technologies available at that time, it is extremely difficult to gain an overview of a single topic. Information is typically distributed over several parts containing a number of volumes, which in turn are split into books. Most of these do not contain any kind of index. In addition, some of the volumes were edited and reprinted and a supplement part was added.

Due to funding limitations, we only dealt with a single volume[6] within the project described in this paper. However, the concepts and technologies have been developed with the complete dataset in mind.

4.3. Digitization and Error Correction

The source material is first digitized using specialized book scanners, producing a TIFF file for each physical page; in our case, with a grayscale resolution of 600dpi.

In a second step, the image files need to be converted to machine-readable text to support, amongst others, NLP analysis and metadata generation. We initially planned to automate this process using OCR software. However, due to the complex layout of the original material (see Figure 2), which contains an abundance of figures, graphs, photos, tables, diagrams, formulas, sketches, footnotes, margin notes, and mixed font sizes, as well as the varying quality of the 100-year old source material, this proved to be too unreliable. As the focus of this project was on developing enhanced semantic support for end users, not basic OCR research, we decided to manually convert the source material into an electronic document. This provided for not only a faster and more reliable conversion, but also accurately captured layout formation in explicit markup, such as footnotes, chapter titles, figure captions, and margin notes. This task was outsourced to a Chinese company for cost reasons; Manual conversion was performed twice to allow an automatic cross-check for error detection. The final, merged version contained only a very small amount of errors, which were eventually hand-corrected during the project.

4.4. Format Transformation and Wiki Upload

The digitized content was delivered in the *TUSTEP*[7] format. This content was first converted to XML, and finally to Wiki markup.

4.4.1. TUSTEP Format

TUSTEP is a toolkit for the "scientific work with textual data" (Uni, 2008), consisting of a document markup standard along with tools for text processing operations on TUSTEP documents. The markup is completely focused on layout, so that the visual structure of printed documents can be captured well. Structurally, it consists both of XML-like elements with an opening and closing tag, such as `<Z>` and `</Z>` for centered passages; and elements serving as control statements, such as `#H:` for starting text in superscript. The control statements remain in effect until another markup element cancels them out, such as `#G:` for adjusting the following text on the baseline.

TUSTEP predates XML, and while it is still in use at many universities, we found it makes automatic processing difficult. The control statements, for instance, make it hard to determine the range of text they affect, because their effect can be canceled by different elements. In addition, in the manual digitization process, markup was applied inconsistently. Therefore, we chose to first convert the data to a custom XML format, designed to closely match the given TUSTEP markup. This also enabled easier structural analysis and transformation of the text due to the uniform tree structure of XML and the availability of high-quality libraries for XML processing.

4.4.2. Custom XML

We developed a custom tool to transform TUSTEP data into XML. The generated XML data intends to be as semantically close to the original markup as possible; as such, it contains mostly layout information such as line and page breaks and font changes. Except for the exact placement of figures and tables, all such information from the original book is retained.

Parsing the XML into a DOM[8] representation provides for easy and flexible data transformation. The resulting XML format can be directly used for NLP corpus generation.

4.4.3. Wiki Markup

To make the historic data accessible via a Wiki, we have to further transform it into the data format used by a concrete Wiki engine. Since we were dealing with an encyclopedic original, we chose the *MediaWiki*[9] system, which is best known for its use within the *Wikipedia*[10] projects.

[6]E. Marx: *Wände und Wandöffnungen* (Walls and Wall Openings). In "Handbuch der Architektur," Part III, Volume 2, Number I, Second edition, Stuttgart, Germany, 1900. Contains 506 pages with 956 figures.

[7]TUebingen System of TExt processing Programs (TUSTEP), http://www.zdv.uni-tuebingen.de/tustep/tustep_eng.html

[8]Document Object Model (DOM), http://www.w3.org/DOM/

[9]MediaWiki, http://en.wikipedia.org/wiki/MediaWiki

[10]Wikipedia, http://www.wikipedia.org

A challenging question was how to perform the concrete conversion from content presented in physical book layout to Wiki pages. Obviously, translating a single book page does not translate well into a single web page. We first attempted to translate each book chapter into a single page (with its topic as the Wiki entry). However, with only 15 chapters in a 500-page book, the resulting Web pages were too long to be used comfortably in the MediaWiki interface. Together with our end users, we finally decided to convert each sub-chapter (section) into a single Wiki page, with additional internal structuring derived from the margin notes preserved by the manual conversion.
MediaWiki uses the markup language *Wikitext*, which was designed as a "simplified alternative to HTML,"[11] and as such offers both semantic markup, like headings with different levels, as well as visual markup, like italic or bold text. Its expressiveness is largely equal to that of HTML, despite the simplified approach, because it lets users insert HTML if Wikitext does not suffice.

Example: Footnote conversion. Footnotes were delivered in TUSTEP in the form `#H:n#G:)` for each footnote n. The markup indicates text being set to superscript (`#H:`), then back to the standard baseline (`#G:`). The footnote reference in the text and the anchor in the footnote section of a page have the same markup, as they look the same. The tool converting to XML locates footnotes using a regular expression, and creates `<footnote to="n" />` resp. `<footnote from="n">...</footnote>` tags. Finally, the conversion to Wikitext transforms the references to `<span id="fn8ref" /><sup>[[#fn8|8)]]</sup>`. The HTML sup tag sets the text as superscript, and its content is a link to the anchor "fn8" on the same page, with the link text simply being "8". The footnote itself is represented by `<span id="fn8"/>''8)'' ... [[#fn8ref|^]]`. We see the anchor linked to from the reference, and vice versa a link to jump back upwards to the reference.

4.4.4. Wiki Interface Features.

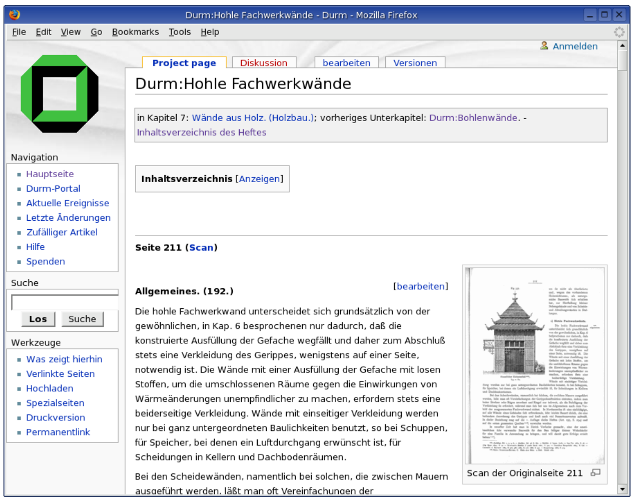
The conversion to Wikitext inserts further information for the Wiki users, such as links to scans of the original pages, and link/anchor combinations to emulate the page-based navigation of the book (see Figure 3). For instance, the beginning of page 211, which is indicated in TUSTEP by `@@1@<S211><`, looks as follows in the resulting Wikitext:

```
<span id="page10" />
'''Seite 211 ([[Media:S211_large.gif|Scan]])'''
[[Image:S211_large.gif|thumb|200px|Scan der
                   Originalseite 211]]
```

4.5. NLP Integration

One of the main goals of our work is to support the end users—groups (1) to (3)—with semantic analysis tools based on NLP. To make our architecture independent from the application domain (architecture, biology, music, . . .) and their custom NLP analysis pipelines, we developed a general integration framework that allows us to deploy any kind of language service. The management, parametrization, and execution of these NLP services is handled in our framework

[11]Wikitext, http://en.wikipedia.org/wiki/Wikitext



Durm:Hohle Fachwerkwände

in Kapitel 7: Wände aus Holz. (Holzbau.); vorheriges Unterkapitel: Durm:Bohlenwände. - Inhaltsverzeichnis des Heftes

Inhaltsverzeichnis [Anzeigen]

Seite 211 (Scan)

Allgemeines. (192.) [bearbeiten]

Die hohle Fachwerkwand unterscheidet sich grundsätzlich von der gewöhnlichen, in Kap. 6 besprochenen nur dadurch, daß die konstruierte Ausfüllung der Gefache wegfällt und daher zum Abschluß stets eine Verkleidung des Gerippes, wenigstens auf einer Seite, notwendig ist. Die Wände mit einer Ausfüllung der Gefache mit losen Stoffen, um die umschlossenen Räume gegen die Einwirkungen von Wärmeänderungen unempfindlicher zu machen, erfordern stets eine beiderseitige Verkleidung. Wände mit einseitiger Verkleidung werden nur bei ganz untergeordneten Baulichkeiten benutzt, so bei Schuppen, für Speicher, bei denen ein Luftdurchgang erwünscht ist, für Scheidungen in Kellern und Dachbodenräumen.

Bei den Scheidewänden, namentlich bei solchen, die zwischen Mauern ausgeführt werden, läßt man oft Vereinfachungen der

Scan der Originalseite 211

Figure 3: The Wiki interface integrating digitized text, scanned originals, and separate "Discussion" pages

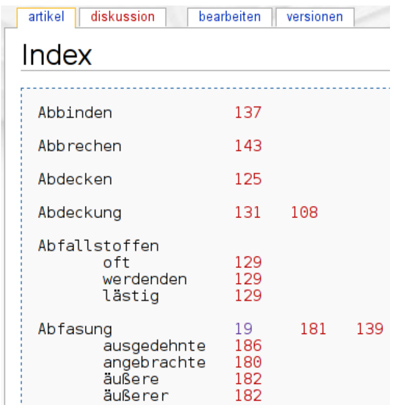
(see Figure 1, Tier 3, "NLP Subsystem") by GATE, the *General Architecture for Text Engineering* (Cunningham et al., 2002). To allow a dynamic discovery of newly deployed language services, we added service descriptions written in OWL to our architecture (see Section 4.1.).
Language services should help the users to find, understand, relate, share, and analyze the stored historic documents. In the following subsections, we describe some of the services we deployed in our implementation to support users of the historic encyclopedia, including index generation, automatic summarization, and OWL metadata generation.

4.5.1. Index Generation

Many documents—like the discussed architectural encyclopedia—do not come with a classical back-of-the-book index. Of course, in the absence of an index, full-text search can help to locate the various occurrences of a single term, but only if the user already knows what he is looking for. An index listing all nouns with their modifiers (adjectives), with links to their locations of occurrence, can help the user find useful information he was not expecting, which is especially important for historical documents, which often contain terminology no longer in use.
For our index, we process all noun phrases found in the analyzed texts. For each noun phrase, we compute the lemma of the head noun and keep track of its modifiers, page number, and corresponding Wiki page. To deal with the problem of correctly lemmatizing historic terminology no longer in use, we developed a self-learning lemmatizer for German (Perera and Witte, 2005). Nouns that have the same lemma are merged together with all their information. Then, we create an inverted index with the lemma as the main column and their modifiers as sub-indexes, as shown in Figure 4. The generated index is then uploaded from the NLP subsystem into the Wiki through a connector ("NLP/Wiki Connector" in Figure 1).

4.5.2. Automatic Summarization

Large text corpora make it impossible for single users to deal with the whole documents in total. The sheer amount of information encoded in natural language in huge text collections poses a non-trivial challenge to information sys-



Index

Abbinden	137		
Abbrechen	143		
Abdecken	125		
Abdeckung	131	108	
Abfallstoffen			
oft	129		
werdenden	129		
lästig	129		
Abfasung	19	181	139
ausgedehnte	186		
angebrachte	180		
äußere	182		
äußerer	182		

Figure 4: NLP-generated full text index, integrated into the Wiki interface (page numbers are hyperlinks to Wiki pages)

tems in order to adequately support the user. To find certain information, to get an overview of a document, or just to browse a text collection, automatic *summarization* (Mani, 2001) offers various methods of condensing texts.[12]
Short, headline-like summaries (around 10 words) that incorporate the most important concepts of a document or a Wiki page facilitate the search for particular information by giving a user an overview of the content at a glance. In addition, full-text summaries can be created for each page, e.g., with a length of 100 words or more. These summaries in free-text form can be read much more quickly than a full-length article, thereby helping a user to decide which Wiki pages he wants to read in full.
More advanced types of summaries can support users during both content creation and analysis. *Multi-document summaries* can combine knowledge from several pages within a Wiki or even across Wiki systems. *Update summaries* keep track of a user's reading history and only present information he has not read before, thereby further reducing the problem of information overload. And *focused summaries* enable the user to formulate a query (natural language questions) the generated summary focuses on. This is especially useful to get a first impression of the available information about a certain topic in a collection. In (Witte et al., 2005), we illustrate the usefulness of focused summaries for a particular architectural scenario.

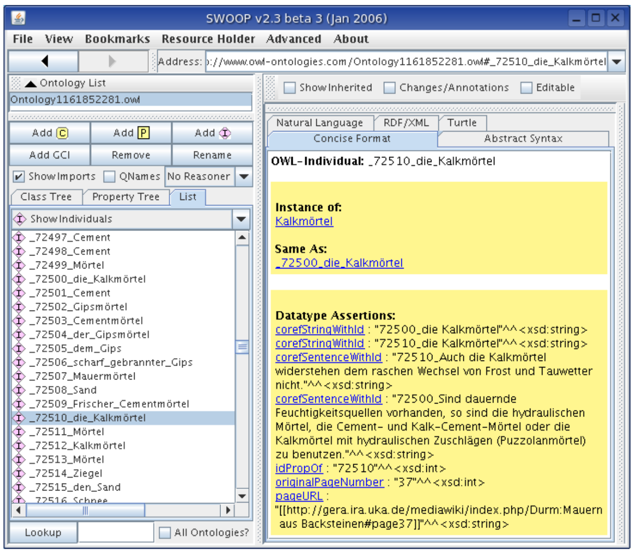
4.5.3. Other NLP Services

The examples presented so far are by no means exhaustive. Depending on the type of data under investigation and the demands of the users concerned with their analysis (groups (1) and (2)), additional NLP services will need to be introduced. Due to our service-oriented approach (cf. Section 4.1.), new services can be added at any time, as they are automatically detected by all connected clients through the metadata repository, without any changes on the client side. Likewise, new user clients can be added dynamically to the architecture, without requiring any changes to the NLP server.

4.6. Semantic Extensions

The NLP analysis services introduced so far are aimed at supporting the user groups (1) and (3): Summaries, full-text

[12]See, e.g., the *Document Understanding Conference* (DUC), http://duc.nist.gov



_72497_Cement
_72498_Cement
_72499_Mörtel
_72500_die_Kalkmörtel
_72501_Cement
_72502_Gipsmörtel
_72503_Cementmörtel
_72504_der_Gipsmörtel
_72505_dem_Gips
_72506_scharf_gebrannter_Gips
_72507_Mauermörtel
_72508_Sand
_72509_Frischer_Cementmörtel
_72510_die_Kalkmörtel
_72511_Mörtel
_72512_Kalkmörtel
_72513_Mörtel
_72514_Ziegel
_72515_den_Sand
_72516_Schnee

OWL-Individual: _72510_die_Kalkmörtel

Instance of:
Kalkmörtel

Same As:
_72500_die_Kalkmörtel

Datatype Assertions:
corefStringWithId : "72500_die Kalkmörtel"^^<xsd:string>
corefStringWithId : "72510_die Kalkmörtel"^^<xsd:string>
corefSentenceWithId : "72510_Auch die Kalkmörtel widerstehen dem raschen Wechsel von Frost und Tauwetter nicht."^^<xsd:string>
corefSentenceWithId : "72500_Sind dauernde Feuchtigkeitsquellen vorhanden, so sind die hydraulischen Mörtel, die Cement- und Kalk-Cement-Mörtel oder die Kalkmörtel mit hydraulischen Zuschlägen (Puzzolanmörtel) zu benutzen."^^<xsd:string>
idPropOf : "72510"^^<xsd:int>
originalPageNumber : "37"^^<xsd:int>
pageURL : "[[http://gera.ira.uka.de/mediawiki/index.php/Durm:Mauern aus Backsteinen#page37]]"^^<xsd:string>

Figure 5: An ontology instance created through NLP

indices, and question-answering all produce new natural language texts, which are convenient for humans. But they are less useful for providing further automated access to the historic data, e.g., through desktop tools targeted at user group (2). In our example scenario, the architects need to integrate the historic knowledge "stored" in the encyclopedia within contemporary architectural design tools: While viewing a certain construction element, the relevant content from the handbook should be extracted and presented alongside other project information. This requires the generation of metadata in a machine-processable data format. In our architecture, this is provided through the NLP-driven population of formal ontologies. We discuss our ontology model in the next subsection, followed by a description of the automatic population process and the querying of the result format.

4.6.1. Ontology Model

Our NLP-generated metadata is formally represented using the *Web Ontology Language* (OWL),[13] which is a standard defined by the World Wide Web Consortium (W3C). Specifically, we use the sub-format OWL-DL, which is based on description logics (DL). OWL is also the foundation of the Semantic Web initiative, which allows us to immediately make use of a large variety of tools and resources developed for OWL-based information processing (editors, storage systems, query languages, reasoners, visualization tools, etc.).
Our ontology has two parts: a *document* ontology describing the domain of NLP (documents, sentences, NPs, coreference chains, etc.) and a *domain* ontology. While the document ontology is independent of the content in the historic documents, the domain ontology has to be developed specifically for their discourse domain. In our example, this ontology needs to contain architectural concepts, such as doors, walls, or windows. By combining both ontologies, we can run semantic *queries* against the ontology, e.g., asking for all sentences where a certain concept appears. The incorporation of CIDOC/CRM could extend our model in the future.

Document Ontology Model. Our document ontology models a number of concepts relevant for the domain of

[13]OWL, http://www.w3.org/2004/OWL/

NLP. One of the main concepts is *document*, representing an individual text processed by an NLP pipeline, containing: the *title* of the document; its *source* address (typically a URL or URI); and a relation *containsSentence* between a document and all its *sentences*.

Likewise, sentences are also represented by an ontology class, with: the start and end position (*beginLocation*, *endLocation*) within the document, given as character offset; the sentence's *content*, stored as plain text, i.e., without additional markup; and a relation *contains* between a sentence and all *named entities* that have been detected in it.

Each of the named entities has, in addition to its ontology class, a number of additional properties: a unique id (*idPropOf*) generated for this instance; the page number (*originalPageNumber*), where the instance can be found in the (printed) source; and the full URL (*pageURL*) for direct access to the instance in the Wiki system.

Additionally, we can represent the result of the coreference resolution algorithm using the OWL language feature *sameAs:* If two instances appear in the same coreference chain, two separate ontology instances are created (containing different ids and possibly different page/URL numbers), but both instances are included in such a *sameAs* relation. This allows ontology reasoners to interpret the syntactically different instances as semantically equivalent. Additionally, a relation *corefStringWithId* is created for every entity in the coreference chain, refering to its unique id stored in the *idPropOf* property; and the content of the sentence containing the co-refering entity is stored in *corefSentenceWithId*.

Domain Ontology Model. In addition to the generic NLP ontology, a domain-specific ontology can be plugged into the system to allow further structuring of the NLP results. If such an ontology is developed, it can also be used to further facilitate named entity detection as described below.

In our approach, we rely on a hand-constructed ontology of the domain. This could be enhanced with (semi-)automatic *ontology enrichment* or *ontology learning*. In general, the design of the domain ontology needs to take the requirements of the downstream applications using the populated ontology into account.

4.6.2. Automatic Ontology Population

We developed an *ontology population* NLP pipeline to automatically create OWL instances (individuals, see Figure 5) for the ontology described above. An overview of the workflow is shown in Figure 6.

The pipeline runs on the XML-based corpus described in Section 4.4. After a number of standard preprocessing steps, including tokenization, POS tagging, and NP chunking, named entities (NEs) are detected using a two-step process. First, an *OntoGazetteer* (Bontcheva et al., 2004) labels each token in the text with all ontology classes it can belong to. And secondly, ontology-aware grammar rules written in the JAPE[14] language are used to find named entities (NEs). Evaluation of the correctness of the generated instances can be conducted using precision and recall measures (Maynard et al., 2006).

[14]Java Annotations Pattern Engine, a regular expression-based language for writing grammars over document annotation graphs.

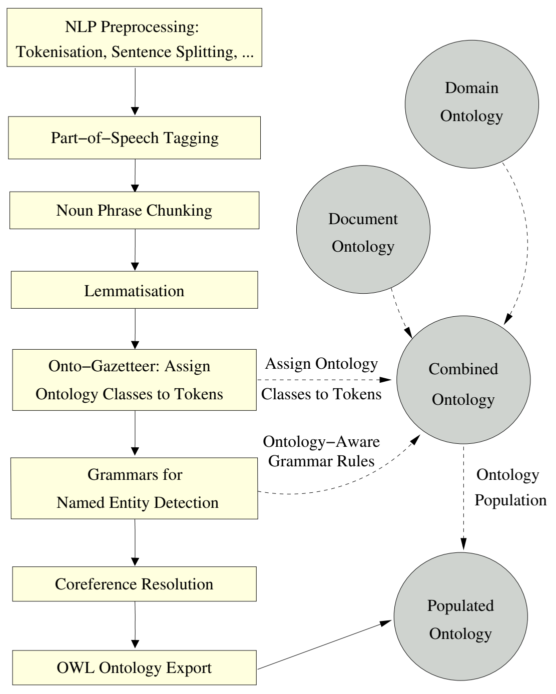

NLP Preprocessing: Tokenisation, Sentence Splitting, ...
Part-of-Speech Tagging
Noun Phrase Chunking
Lemmatisation
Onto-Gazetteer: Assign Ontology Classes to Tokens
Grammars for Named Entity Detection
Coreference Resolution
OWL Ontology Export
Domain Ontology
Document Ontology
Combined Ontology
Populated Ontology
Assign Ontology Classes to Tokens
Ontology-Aware Grammar Rules
Ontology Population

Figure 6: NLP pipeline for ontology population

Finally, the created instances are exported into the result ontology, combining a number of domain and document features. An example instance, of the ontology class *Kalkmörtel* (lime mortar), is shown in Figure 5.

4.6.3. Ontology Queries

The automatically populated ontology represents a machine-readable metadata format that can be *queried* through a number of standardized ontology query languages, such as SPARQL.[15] Queries are a much more expressive paradigm for analyzing text mining results than simple IR; in particular, if a domain model is available, they allow queries over the analyzed documents on a semantic level.

An example SPARQL query is shown in Figure 7. The query shown in the left box represents the question *"Which building materials are mentioned in the handbook together with the concept 'Mauer' (wall), and on which page?"* The result of this query (executed using Protégé[16]), is shown on the right. The first column ("type") shows what kind of entity (stone, plaster, concrete, ...) was found, i.e., a sub-class of "material" in the domain ontology. The results can now be directly inspected by the user or used for further automatic processing by another application.

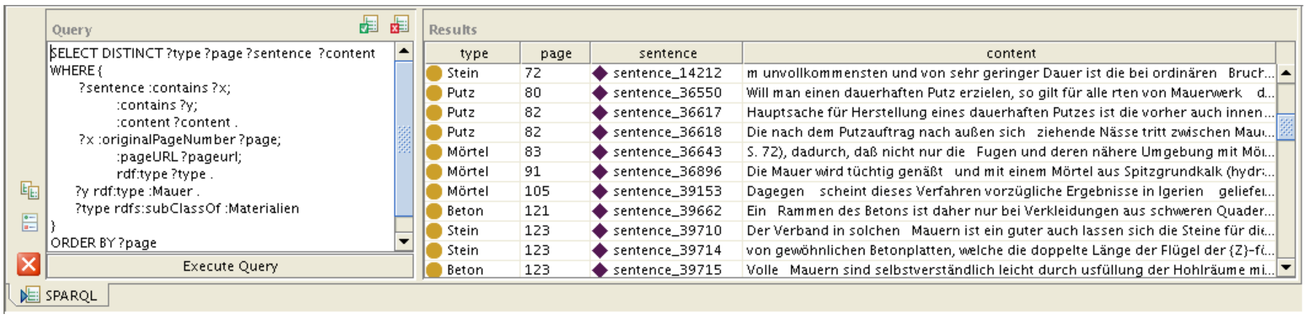
More abstractly speaking, ontology queries support automated problem-solving using a knowledge base. A user of our system, like a historian, might want to formulate hypotheses concerning the source material. Translated into an OWL query, the result can be used to confirm or refute the hypothesis. And as a standardized NLP result format, it also facilitates direct integration into an end-user application or a larger automated knowledge discovery workflow.

4.6.4. Application Integration

The populated ontology also serves as the basis for our final requirement, application integration. With "application" we

[15]SPARQL, http://www.w3.org/TR/rdf-sparql-query/

[16]Protégé, http://protege.stanford.edu/



```
SELECT DISTINCT ?type ?page ?sentence  ?content
WHERE {
      ?sentence :contains ?x;
              :contains ?y;
              :content ?content .
      ?x :originalPageNumber ?page;
              :pageURL ?pageurl;
              rdf:type ?type .
     ?y rdf:type :Mauer .
     ?type rdfs:subClassOf :Materialien
}
ORDER BY ?page
```

type	page	sentence	content
● Stein	72	◆ sentence_14212	m unvollkommensten und von sehr geringer Dauer ist die bei ordinären Bruch...
● Putz	80	◆ sentence_36550	Will man einen dauerhaften Putz erzielen, so gilt für alle rten von Mauerwerk d...
● Putz	82	◆ sentence_36617	Hauptsache für Herstellung eines dauerhaften Putzes ist die vorher auch innen...
● Putz	82	◆ sentence_36618	Die nach dem Putzauftrag nach außen sich ziehende Nässe tritt zwischen Mau...
● Mörtel	83	◆ sentence_36643	S. 72), dadurch, daß nicht nur die Fugen und deren nähere Umgebung mit Mö...
● Mörtel	91	◆ sentence_36896	Die Mauer wird tüchtig genäßt und mit einem Mörtel aus Spitzgrundkalk (hydr...
● Mörtel	105	◆ sentence_39153	Dagegen scheint dieses Verfahren vorzügliche Ergebnisse in Igerien geliefe...
● Beton	121	◆ sentence_39662	Ein Rammen des Betons ist daher nur bei Verkleidungen aus schweren Quader...
● Stein	123	◆ sentence_39710	Der Verband in solchen Mauern ist ein guter auch lassen sich die Steine für di...
● Stein	123	◆ sentence_39714	von gewöhnlichen Betonplatten, welche die doppelte Länge der Flügel der {Z}-f...
● Beton	123	◆ sentence_39715	Volle Mauern sind selbstverständlich leicht durch usfüllung der Hohlräume mi...

Figure 7: Posing a question to the historic knowledge base through a SPARQL query against the NLP-populated ontology

mean any end-user accessible system that wants to integrate the historic data within a different context. For example, in a museum setting, such an application might allow a visitor to access content directly relevant to an artifact. A lexicographer might want to query, navigate, and read content from historical documents while developing a lexical entry. And in our application example, an architect needs access to the knowledge stored in the handbook while planning a particular building restoration task. Here, construction elements displayed in a design tool (such as *window* or *window sill*) can be directly connected with the ontological entities contained in the NLP-populated knowledge. This allows an architect to view relevant content down to the level of an individual construction element using the named entities, while retaining the option to visit the full text through the provided Wiki link.

5. Summary and Conclusions

To support users in the cultural heritage domain, a precise analysis of the different user groups and their particular requirements is essential. In this paper, we present a holistic approach based on a unified system architecture that highlights the many inter-dependencies in supporting different groups with particular features, aimed at different use cases: *Historians* have the support of NLP analysis tools and a user-friendly Web-based access and collaboration tool build around a standard Wiki system. *Laypersons* also benefit from these user-friendly features, while *practitioners*—in our scenario building architects—can additionally use NLP-generated ontology metadata for direct application integration. Finally, our approach also supports computational linguists through corpus construction and querying tools.

The experience from the implemented system using the example of a historical encyclopedia of architecture demonstrates the usefulness of these ideas. Finally, providing a machine-readable knowledge base that integrates textual instances and domain-specific entities is consistent with the vision of the Semantic Web, which has the potential to further enhance knowledge discovery for cultural heritage data.

6. References

Kalina Bontcheva, Valentin Tablan, Diana Maynard, and Hamish Cunningham. 2004. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. of the 40th Anniversary Meeting of the ACL*. http://gate.ac.uk.

Martin Doerr. 2003. The CIDOC Conceptual Reference Module: An Ontological Approach to Semantic Interoperability of Metadata. *AI Mag.*, 24(3):75–92.

Michel Généreux. 2007. Cultural Heritage Digital Resources: From Extraction to Querying. In *Proceedings of the Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2007)*, pages 41–48, Prague, Czech Republic, June. ACL.

Markus Krötzsch, Denny Vrandečić, and Max Völkel. 2006. Semantic MediaWiki. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora Aroyo, editors, *The Semantic Web – ISWC 2006*, volume 4273 of *LNCS*, pages 935–942. Springer.

Bo Leuf and Ward Cunningham. 2001. *The Wiki Way, Quick Collaboration on the Web*. Addison-Wesley.

Peter Lyman and Hal R. Varian. 2003. How Much Information?

I. Mani. 2001. *Automatic Summarization*. John Benjamins B.V.

Efthimios C. Mavrikas, Nicolas Nicoloyannis, and Evangelia Kavakli. 2004. Cultural Heritage Information on the Semantic Web. In Enrico Motta, Nigel Shadbolt, Arthur Stutt, and Nicholas Gibbins, editors, *EKAW*, volume 3257 of *Lecture Notes in Computer Science*, pages 477–478. Springer.

D. Maynard, W. Peters, and Y. Li. 2006. Metrics for Evaluation of Ontology-based Information Extraction. In *Proceedings of the 4th International Workshop on Evaluation of Ontologies on the Web (EON 2006)*, Edingburgh, UK, May.

Praharshana Perera and René Witte. 2005. A Self-Learning Context-Aware Lemmatizer for German. In *Proc. of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, pages 636–643, Vancouver, BC, Canada, October 6–8.

Jeffrey A. Rydberg-Cox. 2002. Cultural Heritage Language Technologies: Building an Infrastructure for Collaborative Digital Libraries in the Humanities. *Ariadne*, 34, December.

Jeffrey A. Rydberg-Cox. 2005. The Cultural Heritage Language Technologies Consortium. *D-Lib Magazine*, 11(5), May.

Sebastian Schaffert. 2006. IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In *WETICE*, pages 388–396.

Patrick Sinclair, Paul Lewis, Kirk Martinez, Matthew Addis, Adrian Pillinger, and Daniel Prideaux. 2005. eCHASE: Exploiting Cultural Heritage using the Semantic Web. In *4th International Semantic Web Conference (ISWC 2005)*, Galway, Ireland, November 6–10.

L. Sweeney. 2001. Information Explosion. In L. Zayatz, P. Doyle, J Theeuwes, and J. Lane, editors, *Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies*. Urban Institute, Washington, DC.

Universität Tübingen – Zentrum für Datenverarbeitung, 2008. *TUSTEP: Handbuch und Referenz*. Version 2008.

René Witte, Petra Gerlach, Markus Joachim, Thomas Kappler, Ralf Krestel, and Praharshana Perera. 2005. Engineering a Semantic Desktop for Building Historians and Architects. In *Proc. of the Semantic Desktop Workshop at the ISWC 2005*, volume 175 of *CEUR*, pages 138–152, Galway, Ireland, November 6.