Integrating Wiki Systems, Natural Language Processing, and Semantic Technologies for Cultural Heritage Data Management

René Witte, Thomas Kappler, Ralf Krestel, and Peter C. Lockemann

Abstract Modern documents can easily be structured and augmented to have the characteristics of a semantic knowledge base. Many older documents may also hold a trove of knowledge that would deserve to be organized as such a knowledge base. In this chapter, we show that modern semantic technologies offer the means to make these heritage documents accessible by transforming them into a semantic knowledge base. Using techniques from natural language processing and Semantic Computing, we automatically populate an ontology. Additionally, all content is made accessible in a user-friendly Wiki interface, combining original text with NLP-derived metadata and adding annotation capabilities for collaborative use. All these functions are combined into a single, cohesive system architecture that addresses the different requirements from end users, software engineering aspects, and knowledge discovery paradigms. The ideas were implemented and tested with a volume from the historic Encyclopedia of Architecture and a number of different user groups.

1 Introduction

Modern documents can be turned into veritable knowledge bases by linking the text or parts thereof to a multitude of supportive data such as explication of semantics, user notes, discussion panels, background information, current news, quotes and

René Witte
Concordia University, Montréal, Canada, e-mail: witte@semanticsoftware.info

Thomas Kappler
Swiss Institute of Bioinformatics, Geneva, Switzerland, e-mail: tkappler@googlemail.com

Ralf Krestel
L3S Research Center, Hannover, Germany, e-mail: krestel@l3s.de

Peter C. Lockemann
Karlsruhe Institute of Technology, Germany, e-mail: Lockemann@kit.edu

citations, illustrative material, or more detailed presentations. Older documents such as books dealing with our cultural heritage often deserve the same kind of support. However, they exist only in analog form, and one has to search for and find related material by inspecting an often huge number of textual sources. If such documents are to be repeatedly accessed by a group of peers it might be worthwhile to structure them along the lines of modern documents.

Since doing so by hand is a cumbersome and lengthy affair, one should find ways to build an initial structure by automatically extracting relevant information from the analog document. Our thesis is that extraction should be based on an understanding of the semantics contained in the document with its text, tables, figures, etc. We studied the issue in the context of a project for developing enhanced semantic support for users of textual cultural heritage data, more specifically on the historic Encyclopedia of Architecture, written in German between 1880–1943. Our aim was to apply modern semantic technologies to make these heritage documents more flexibly accessible by transforming them into a semantic knowledge base. More specifically, by using techniques from natural language processing and Semantic Computing, we automatically populate an ontology that allow building historians to navigate and query the encyclopedia, while architects can directly integrate it into contemporary construction tools. Additionally, all content is made accessible in a user-friendly Wiki interface, combining original text with NLP-derived metadata and adding annotation capabilities for collaborative use.

A particular result 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 one volume of the historic encyclopedia of architecture and a number of different user groups, including building historians, architects, and NLP system developers.

We discuss the user groups and their requirements in Section 2, examine in Section 3 the related work, and then develop our solution in Section 4. Section 5 closes the chapter.

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 approach, we consider the requirements from four different user groups; each of them having a different background and expectations concerning the management of historical textual data.

(1) Historians: Within this group, we target users that deal with historical 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 historical 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 historical 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: Historical 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 historical 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 explicit requirements a system needs to fulfill, based on the user groups defined above:

Web Interface. To make the historical 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 historical data from a scientific point of view—in particular group (1)—often need to comment, add, and collaborate on the historical data. This should be supported within the same interface as the primary

(historical) 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 historical 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 historical 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 (historical) 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 historical 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 (open linked data) and OWL (Semantic Web).

Application Integration. As pointed out in the last requirement, external applications should be provided with automated access to the historical 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 [13, 14] 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 analyses. This metadata can also contain hyperlinks to connect related texts with each other. The importance of correct morphological analysis is stressed as a baseline technol-

ogy for users in the humanities, a statement which is also reflected in our work by integrating a self-learning lemmatizer for the German language [12] 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 by Mavrikas et al. [10], 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 cultural heritage data from different sources in the Internet and processes the data afterwards. An ontology [3] 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. Fujisawa [4] proposes to facilitate the access to images of cultural heritage by extracting metadata from the accompanying natural language text. Paraphrasing of the descriptions and the metadata based on the knowledge and experience of the user is proposed as a second step.

Another approach based on the CIDOC-CRM[1] ontology is presented by Genereux [5]. The system described there consists of two parts, one for extracting cultural heritage 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.

Sinclair et al. [17] present a system 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) [7] or IkeWiki [15]. 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 capa-

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

bilities), semantic Wikis like SMW have explicit notational and internal semantic capabilities. The next version of our Wiki/NLP integration architecture, currently under development, will support the SMW extension for storing results of NLP pipelines.

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, 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 19^{th} 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 [8]. 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. Fig. 1 shows how these and the remaining components are systematically assembled to form the overall *Semantic Assistants* architecture of our system [23].

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

[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.

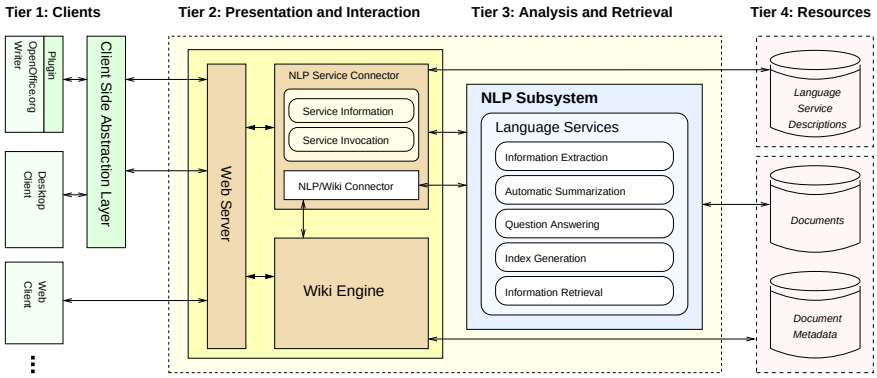

Fig. 1 System architecture overview

application suite, can also be extended to be integrated with our architecture [6]. 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 conversion 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 a SOAP 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 [22].

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. The latest version of the architecture, as well as some example NLP services, is available as open source software.[5]

[4] Web Services Architecture, http://www.w3.org/TR/ws-arch/

[5] Semantic Assistants, http://www.semanticsoftware.info/semantic-assistants-project

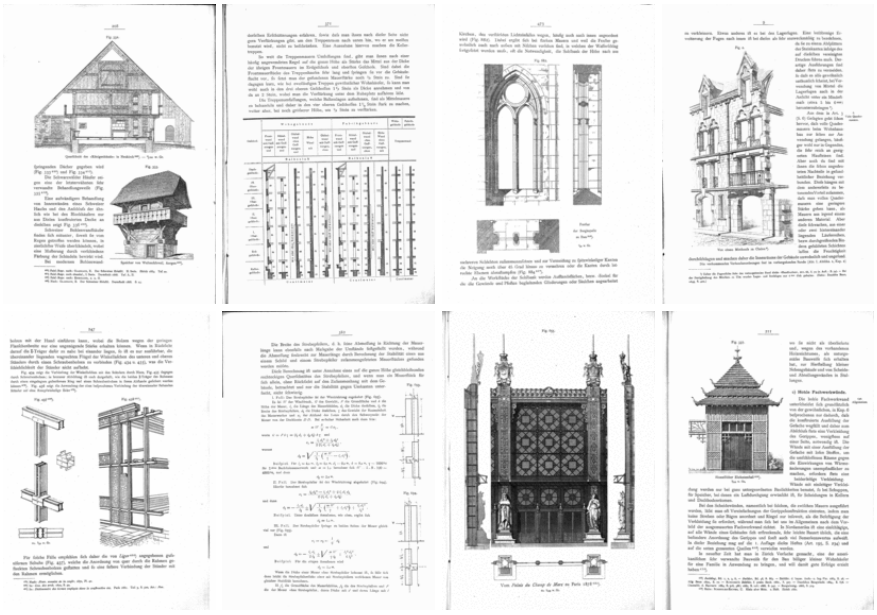

Fig. 2 Source material examples: Scanned pages from *Handbuch der Architektur* (1900)

4.2 Source Material

We implemented and evaluated the ideas described here for a particular set of historical documents: the German *Handbuch der Architektur*, a comprehensive multivolume encyclopedia of architecture.[6] The full encyclopedia was written between the late 19^{th} and early 20^{th} 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 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.

[6] Edited by Joseph Durm (⋆14.2.1837 Karlsruhe, Germany, †3.4.1919 ibidem) and three other architects since 1881.

Due to funding limitations, we only dealt with a single volume[7] within the project described in this chapter. However, the concepts and technologies have been designed with the complete dataset in mind.

4.3 Digitization and Error Correction

The source material was 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 needed 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 Fig. 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. It is freely available online under an open content license.[8]

4.4 Format Transformation and Wiki Upload

The digitized content was delivered in the *TUSTEP*[9] format. This content was first converted to XML, and finally to Wiki markup. In the following, we briefly describe the conversion process.

---

[7] 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.

[8] Durm Corpus, http://www.semanticsoftware.info/durm-corpus

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

4.4.1 TUSTEP Format

TUSTEP is a toolkit for the "scientific work with textual data" [18], 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[10] representation provides for easy and flexible data transformation, e.g., changing an element node of the document tree such as `<page no="12">` to a text node containing the appropriate Wiki markup in the next step. The resulting XML format can be directly used for NLP corpus generation, which is then loaded into an NLP framework, such as GATE [2]. This XML corpus is also freely available online.[11]

4.4.3 Wiki Markup

To make the historical 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

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

[11] Durm Corpus, http://www.semanticsoftware.info/durm-corpus

Fig. 3 The Wiki interface integrating digitized text, scanned originals, and separate "Discussion" pages

an encyclopedic original, we chose the *MediaWiki*[12] system, which is best known for its use within the *Wikipedia*[13] projects. MediaWiki stores the textual content in a MySQL database, the image files are stored as plain files on the server. It provides a PHP-based dynamic web interface for browsing, searching, and manual editing of the content.

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.

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

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

MediaWiki uses the markup language *Wikitext*, which was designed as a "simplified alternative to HTML,"[14] 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 the format `<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 Fig. 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]]
```

4.4.5 Wiki Data Upload

The workflow between the Wiki and the NLP subsystems is shown in Fig. 4. The individual sub-components are loosely coupled through XML-based data exchange. Basically, three steps are necessary to populate the Wiki with both the encyclopedia text and the additional data generated by the NLP subsystem. Firstly (Step 1 in Fig. 4), the original Tustep markup of the digitized version of the encyclopedia is converted to XML as describe above. In Step 2, the XML data is converted to the text markup used by MediaWiki. And finally (Step 3), the created Wiki markup is added to the MediaWiki system using parts of the Python Wikipedia Robot Framework,[15]

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

[15] Python Wikipedia Robot Framework, http://pywikipediabot.sf.net

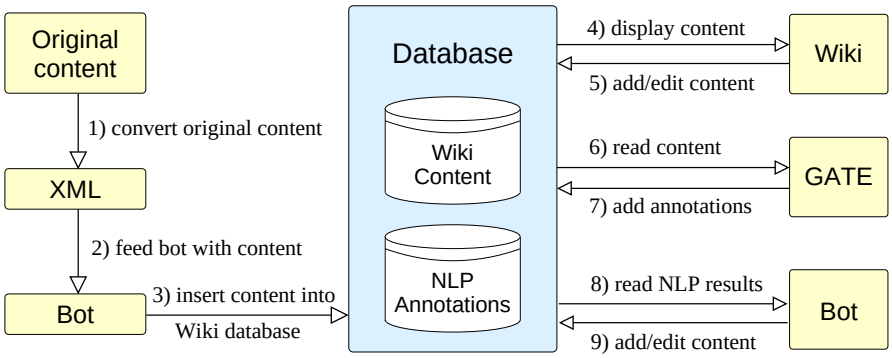

Fig. 4 Workflow between document storage, retrieval, and NLP analysis

a library offering routines for tasks such as adding, deleting, and modifying pages of a Wiki or changing the time stamps of pages.

4.5 Integrating Natural Language Processing

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, parameterization, and execution of these NLP services is handled in our framework (see Fig. 1, Tier 3, "NLP Subsystem") by GATE, the *General Architecture for Text Engineering* [2]. 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 historical 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 ontology population.

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 a user

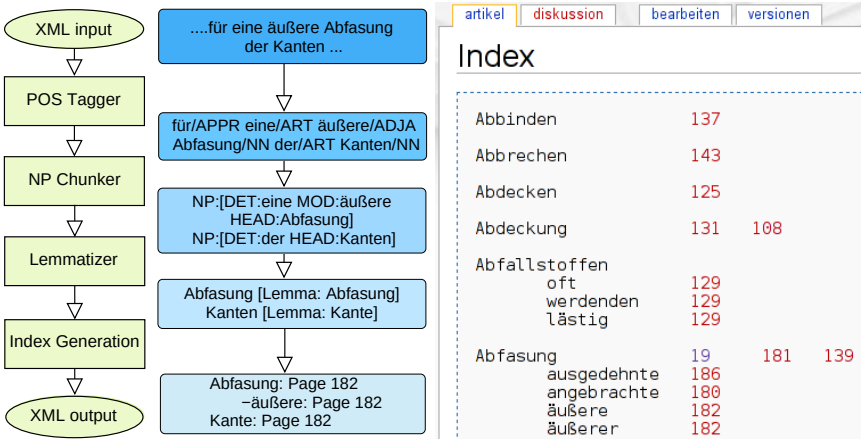

Fig. 5 NLP-generated full text index, integrated into the Wiki interface (page numbers are hyperlinks to Wiki pages)

finding useful information he was not expecting, which is especially important for historical documents, which often contain terminology no longer in use.

For our automatic index generation, shown in Fig. 5, we first determine the part-of-speech for each word (noun, verb, adjective, etc.) using the TreeTagger [16].[16] Based on this information, the open source chunker MuNPEx[17] groups words into *noun phrases* (NPs), which consist of a head noun, a (possibly empty) list of adjectives, and an optional determiner. 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 historical terminology no longer in use, we developed a self-learning lemmatizer for German [12], which is freely available online.[18] 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 Fig. 5. The generated index is then uploaded from the NLP subsystem into the Wiki through a connector ("NLP/Wiki Connector" in Fig. 1).

4.5.2 Automatic Summarization

Large text corpora make it impossible for single users to deal with the whole document set. The sheer amount of information encoded in natural language in huge text collections poses a non-trivial challenge to information systems in order to adequately support the user. To find certain information, to get an overview of a docu-

[16] TreeTagger, http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

[17] Multi-Lingual Noun Phrase Extractor (MuNPEx), http://www.semanticsoftware.info/munpex

[18] Durm German Lemmatizer, http://www.semanticsoftware.info/durm-german-lemmatizer

"Welche Art von Putz bietet Schutz vor Witterung?"
Ist das Dichten der Fugen für die Erhaltung der Mauerwerke, namentlich an den der Witterung ausgesetzten Stellen, von Wichtigkeit, so ist es nicht minder die Beschaffenheit der Steine selbst. Bei der früher allgemein üblichen Art der gleichzeitigen Ausführung von Verblendung und Hintermauerung war allerdings mannigfach Gelegenheit zur Beschmutzung und Beschädigung der Verblendsteine geboten. Will man einen dauerhaften Putz erzielen, so gilt für alle Arten von Mauerwerk die Regel, da die zu putzenden Flächen frei von Staub sein mssen, da dieser trennend zwischen Mauer und Putz wirken und das feste Anhaften des letzteren verhindern würde. ...

Fig. 6 Excerpt from a focused summary generated based on a question (shown on top)

ment, or just to browse a text collection, automatic *summarization* [9] offers various methods of condensing texts.[19]

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. *Contrastive Summaries* [20] can support a user in highlighting differences across a number of articles (or article versions) on the same topic, thereby showing both commonalities and differences. In our project, we contrasted modern building standards (DIN/SIN) with content from the historic encyclopedia.

Focused summaries [19] 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. An example for such a summary is shown in Fig. 6: This summary provides a series of relevant sentences in answer to the user's question, *"Welche Art von Putz bietet Schutz vor Witterung?"* (Which kind of plaster would be suitable to protect brickwork against weather influences?). In [21], we further discuss the usefulness of focused summaries for a particular architectural scenario.

4.5.3 Integrating further NLP Web 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,

[19] See, e.g., the *Text Analysis Conference* (TAC), http://www.nist.gov/tac

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 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 historical data, e.g., through desktop tools targeted at user group (2). In our example scenario, the architects need to integrate the historical 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 format. In our architecture, this is provided through the NLP-driven population of formal (OWL-DL) 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),[20] 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). DLs describe domains in terms of TBox (also known as concepts or classes), roles (also known as relationships or properties) and ABox (also known as individuals or instances). 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, triplestores, 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 historical 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.

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

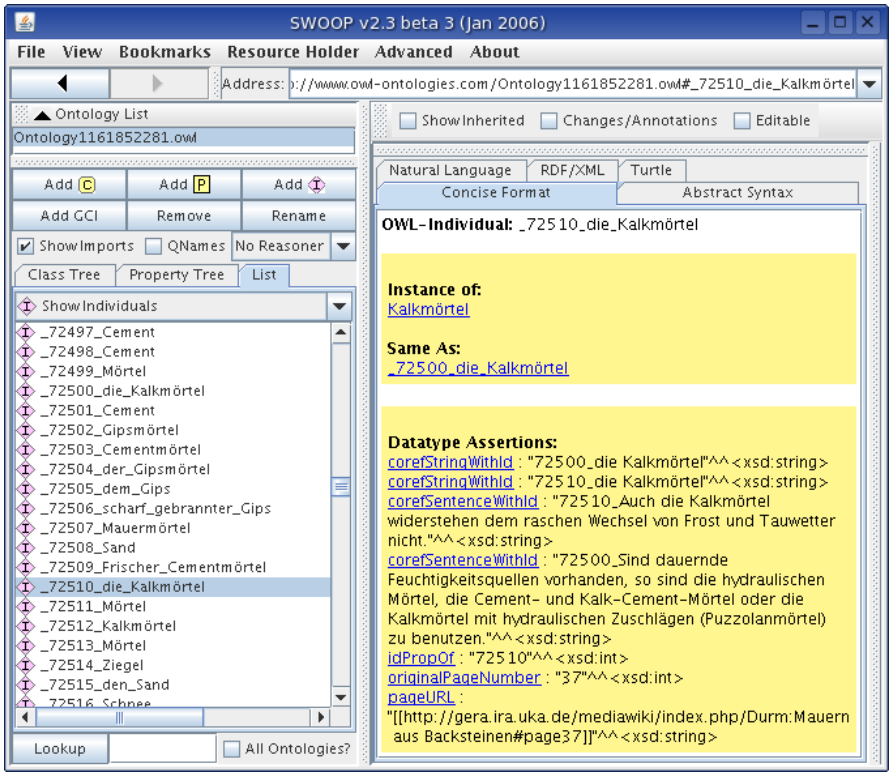

Fig. 7 An ontology instance created through NLP

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

erence 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 Fig. 7) for the ontology described above. An overview of the workflow is shown in Fig. 8.

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* [1] labels each token in the text with all ontology classes it can belong to. And secondly, ontology-aware grammar rules written in the JAPE[21] language are used to find named entities (NEs). Evaluation of the correctness of the generated instances can be conducted using precision and recall measures [11].

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 Fig. 7. This ontology population process is facilitated by an application-independent GATE component, the *OwlExporter* [24], which we made available as open source software.[22]

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,

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

[22] OwlExporter, http://www.semanticsoftware.info/owlexporter

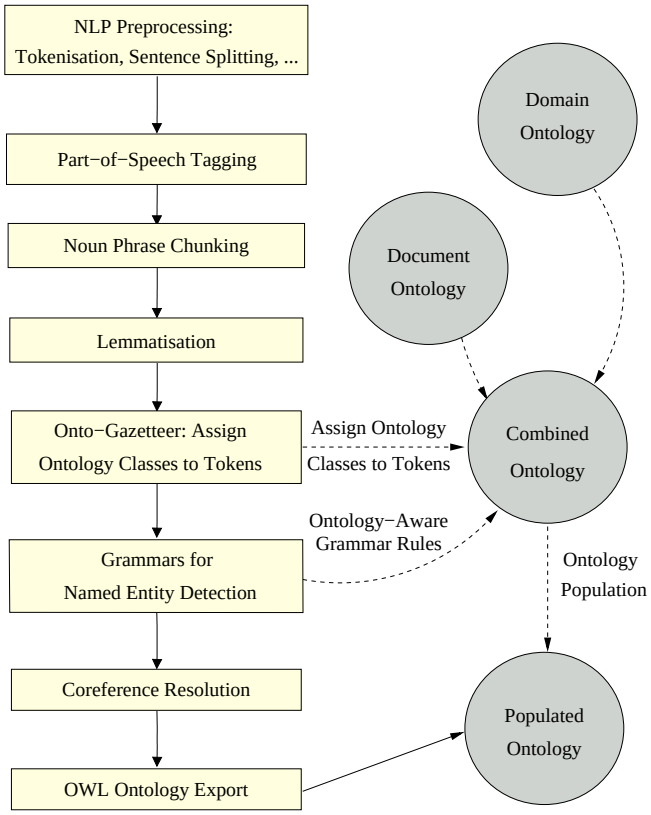

Fig. 8 NLP pipeline for ontology population

such as SPARQL.[23] Queries are a much more expressive paradigm for analyzing text mining results than simple information retrieval (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 Fig. 9. 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[24]), 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,

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

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

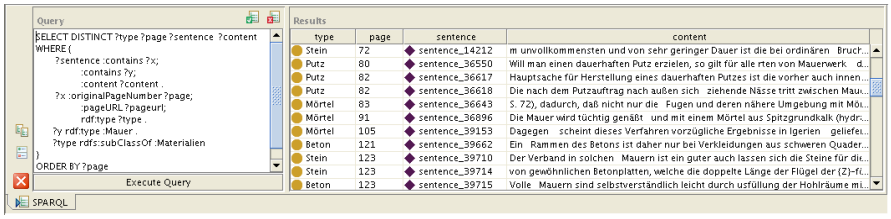

Fig. 9 Posing a question to the historical knowledge base through a SPARQL query against the NLP-populated ontology

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 mean any end-user accessible system that wants to integrate the historical 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

The thesis underlying our work was that by understanding the semantics contained in a document one can transform older documents into an initial semantic knowledge base. We demonstrated for an encyclopedia from the cultural heritage domain that this can indeed be done. We developed a methodology for organizing the transformation process, and we identified the necessary tools for implementing the methodology. To support users in the cultural heritage domain, a precise analysis of the different user groups and their particular requirements is essential. The challenge was to find 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. Indeed, providing a machine-readable knowledge base that integrates textual instances and domain-specific entities is consistent with the vision of the Semantic Web. The data for the encyclopedia, as well as a number of tools, are publicly accessible under open source licenses.

We believe that our methodology and tools are general enough to be applied to other knowledge domains, and hence have the potential to further enhance knowledge discovery for cultural heritage data.

Acknowledgements Praharshana Perera contributed to the automatic index generation and the Durm lemmatizer. Qiangqiang Li contributed to the ontology population pipeline. Thomas Gitzinger contributed to the index generation.

References

1. Kalina Bontcheva, Valentin Tablan, Diana Maynard, and Hamish Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 2004.
2. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. of the 40th Anniversary Meeting of the ACL*, 2002. http://gate.ac.uk.
3. Martin Doerr. The CIDOC Conceptual Reference Module: An Ontological Approach to Semantic Interoperability of Metadata. *AI Mag.*, 24(3):75–92, 2003.
4. Satoko Fujisawa. Automatic creation and enhancement of metadata for cultural heritage. *Bulletin of the IEEE Technical Committee on Digital Libraries (TCDL)*, 3(3), 2007.
5. Michel Généreux. 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 2007. ACL.
6. Thomas Gitzinger and René Witte. Enhancing the OpenOffice.org Word Processor with Natural Language Processing Capabilities. In *Natural Language Processing resources, algorithms and tools for authoring aids*, Marrakech, Morocco, June 1 2008.
7. Markus Krötzsch, Denny Vrandečić, and Max Völkel. 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, 2006.
8. Bo Leuf and Ward Cunningham. *The Wiki Way, Quick Collaboration on the Web*. Addison-Wesley, 2001.
9. I. Mani. *Automatic Summarization*. John Benjamins B.V., 2001.
10. Efthimios C. Mavrikas, Nicolas Nicoloyannis, and Evangelia Kavakli. 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, 2004.
11. D. Maynard, W. Peters, and Y. Li. 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 2006.
12. Praharshana Perera and René Witte. 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 2005.
13. Jeffrey A. Rydberg-Cox. Cultural Heritage Language Technologies: Building an Infrastructure for Collaborative Digital Libraries in the Humanities. *Ariadne*, 34, December 2002.
14. Jeffrey A. Rydberg-Cox. The Cultural Heritage Language Technologies Consortium. *D-Lib Magazine*, 11(5), May 2005.
15. Sebastian Schaffert. IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In *WETICE*, pages 388–396, 2006.
16. H. Schmid. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*, 1995.
17. Patrick Sinclair, Paul Lewis, Kirk Martinez, Matthew Addis, Adrian Pillinger, and Daniel Prideaux. eCHASE: Exploiting Cultural Heritage using the Semantic Web. In *4th International Semantic Web Conference (ISWC 2005)*, Galway, Ireland, November 6–10 2005.
18. Universität Tübingen – Zentrum für Datenverarbeitung. *TUSTEP: Handbuch und Referenz*, 2008. Version 2008.
19. René Witte and Sabine Bergler. Fuzzy Clustering for Topic Analysis and Summarization of Document Collections. In Z. Kobti and D. Wu, editors, *Proc. of the 20th Canadian Conference on Artificial Intelligence (Canadian A.I. 2007)*, LNAI 4509, pages 476–488, Montréal, Québec, Canada, May 28–30 2007. Springer.
20. René Witte and Sabine Bergler. Next-Generation Summarization: Contrastive, Focused, and Update Summaries. In *International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*, Borovets, Bulgaria, September 27–29 2007.
21. René Witte, Petra Gerlach, Markus Joachim, Thomas Kappler, Ralf Krestel, and Praharshana Perera. 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 2005.
22. René Witte and Thomas Gitzinger. Connecting Wikis and Natural Language Processing Systems. In *WikiSym '07: Proceedings of the 2007 International Symposium on Wikis*, pages 165–176, New York, NY, USA, 2007. ACM.
23. René Witte and Thomas Gitzinger. Semantic Assistants – User-Centric Natural Language Processing Services for Desktop Clients. In *3rd Asian Semantic Web Conference (ASWC 2008)*, volume 5367 of *LNCS*, pages 360–374, Bangkok, Thailand, February 2–5 2009. Springer.
24. René Witte, Ninus Khamis, and Juergen Rilling. Flexible Ontology Population from Text: The OwlExporter. In *The Seventh International Conference on Language Resources and Evaluation (LREC 2010)*, pages 3845–3850, Valletta, Malta, May 19–21 2010. ELRA.