An Extensible Semantic Wiki Architecture

Jochen Reutelshoefer, Fabian Haupt, Florian Lemmerich, Joachim Baumeister

Institute for Computer Science, University of Würzburg, Germany
email: {reutelshoefer, fhaupt, lemmerich, baumeister}@informatik.uni-wuerzburg.de

Abstract. Wikis are prominent for successfully supporting the quick and simple creation, sharing and management of content on the web. Semantic wikis improve this by semantically enriched content. Currently, notable advances in different fields of semantic technology like (paraconsistent) reasoning, expressive knowledge (e.g., rules), and ontology learning can be observed. By making use of these technologies, semantic wikis should not only allow for the agile change of its content but also the fast and easy integration of emerging semantic technologies into the system. Following this idea, the paper introduces an extensible semantic wiki architecture.

1 Introduction

Semantic wikis have become an attractive solution for collaborative knowledge formalization and sharing. One major challenge at that task is the fact that knowledge can be represented on many different levels of formality and in a wide variety of formalisms [1]. Further, the requirements for a semantic wiki application strongly depend on the domain and the targeted community. In consequence, a wide range of diverse semantic wiki approaches has evolved employing different techniques, e.g. different types of formalized content or reasoning capabilities. While, for example, Semantic Media Wiki [2] focuses on efficient reasoning on large data, IkeWiki [3] allows for easy ontology editing with rich expressiveness. SweetWiki [4] provides an elaborated *Wiki Object Model*. The OntoWiki system [5] supports the combination and visualization of multimedia data. AceWiki [6] follows a different knowledge acquisition strategy using a controlled language. Further wikis work with mathematical, prolog-based or classification knowledge [7–9]. All these systems were created having various distinct application scenarios in mind. Accordingly, they are utilizing a wide range of strategies and technologies, e.g., efficient reasoners, specialized reasoners, controlled languages, various markups, different content browsers and visualizations. Each of these systems is fitting well to its particular intended purposes but if one intends to support a specific (novel) semantic wiki application each of the systems reveals advantages *and* disadvantages. Without a suitable extensible semantic wiki one has to choose a suboptimal solution or implement an entire new semantic wiki system from scratch. We claim that the optimal solution to support a semantic wiki application needs to be adapted precisely to its requirements, considering the domain, the targeted user community and the envisioned

use-cases thoroughly. This analysis reveals the appropriate formalisms, reasoning support, visualization and querying capabilities needed. But in order to serve a wider range of these possible requirements and to be able to optimize each semantic wiki application to its domain and community it is beneficial to have an extensible semantic wiki architecture with a basic toolkit for knowledge formalization, knowledge visualization and reasoning. We envision that this kind of extensible semantic wiki architecture enables the customization of a semantic wiki system to a particular application at low (software) engineering costs offering various reusable and extensible components. Providing optimal support to any (non-technical) domain and community will raise the general acceptance and spread of this technology. Beside selection and integration of existing techniques into a semantic wiki also the agile integration of novel technologies can improve the general semantic wiki functionality and customization to specific applications. We presume, that especially advanced reasoners and NLP-techniques employed for semi-automated knowledge formalization can bring considerable benefit when being combined with the existing wiki technology.

In this paper we describe the concept of an extensible semantic wiki architecture and motivate the emerging possibilities. With the system KnowWE we present a prototype of an extensible semantic wiki architecture and show its current extensions.

2 Challenges and Dimensions of extending Semantic Wikis

We briefly outline a conceptual view on semantic wikis in general followed by the discussion of the possibilities and challenges extending semantic wikis. Figure 1 shows the three components of what we call the "knowledge pipeline" in semantic wikis. It shows the flow of the formalized knowledge from the contributing user role to the consuming user role through the *Knowledge Formalization Component*, the *Reasoning Component*, and the *Knowledge Presentation Component*.

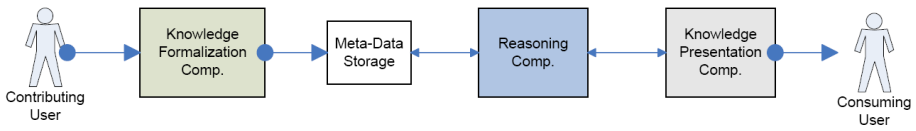

Fig. 1. Sketch of the "knowledge pipeline" of a semantic wiki.

– **Knowledge Formalization Component:** The knowledge formalization component allows the user to formalize parts of the (textual) knowledge. This usually is done by markups or (semantic) forms. These markups are extracted and transformed into a target representation which is commonly

stored explicitly (e.g., in RDF) to allow for efficient reasoning. This transformation implicitly defines the semantics of the formalized knowledge having the target reasoner and the ontology in mind.
- **Reasoning Component:** A reasoning component uses the formalized knowledge created by the knowledge formalization component and is able to deduce higher-level information from it. While most semantic wikis employ an RDF-reasoner, there are several other reasoning approaches present, that are beneficial in particular application scenarios.
- **Knowledge Presentation Component:** This component describes the method how the additional functionality provided by meta-data and reasoner is used to supply the user with the right (high-level) information in a suitable form. This includes as an important aspect the visualization of the results or functionality. The reasoning capabilities can be used to provide semantic navigation, querying, rendering fact sheets, meta-data browser, and more.

These components together provide the additional value of a semantic wiki. In the following the possibilities of extending each of these components are discussed in more detail.

2.1 Dimensions of Semantic Wiki Extensions

As mentioned in Section 1 the possible extensions of semantic wikis are manifold and cannot be foreseen in general. In order to allow for many kinds of extensions we discuss each of the three components separately:

1. **Formalization extension:** Given any methods (e.g., markup) to insert atomic formal relations, in a technical point of view any knowledge base can be created. However, the widespread employment of semantic technology is hindered by the formalization task being not simple and efficient enough (*Knowledge Acquisition Bottleneck* [10]). One way to counteract is lowering the barriers of knowledge formalization. The development of (domain specific) high-level markup languages with comfortable editing support can help to make knowledge definition compact, transparent and efficient. The use of controlled language is one approach in this direction [11]. Another possibility for reducing the workload of the domain specialists is the integration of (preconfigured) text mining methods, that propose formalizations based on the informal text content. Thus, the users only have to decide whether to confirm or dismiss a formalization proposition.
2. **Reasoning extension:** Although basic reasoning engines are currently available there are still challenges with respect to scalability and expressiveness [12] to be addressed. Further, there is ongoing research to cope with inconsistent knowledge, incompleteness and uncertainty [13–15]. For some applications it will be valuable to replace or enhance the basic reasoning engine by an early prototype result from such research work.
3. **Presentation extension:** The challenge of these kinds of extensions is to present the user the right high-level information in the right form at the

right time (without overflowing him). These extensions must be specified according to the use-cases addressed by the intended application. One frequent application might be precompiled (possibly parameterized) use-case specific queries decorated by a GUI component for execution and having a visualization component attached for result presentation (e.g., table-based, graph-based, highlighted).

When designing an extensible semantic wiki architecture these three levels of extension need to be considered as shown in Figure 2.

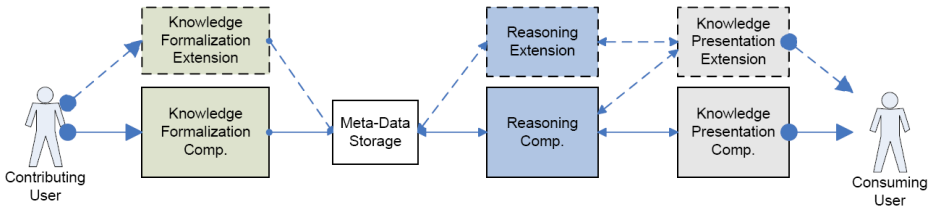

Fig. 2. Sketch of the "knowledge pipeline" of a semantic wiki with extensions.

Examining the possibilities of the extensions of each level it becomes evident, that an extension on one component can extend the entire functionality of a semantic wiki system. Thus, the three components can be extended separately or combined. This results in the semantic wiki extension space sketched in Figure 3. Assuming that the core semantic wiki system itself already provides some functionality in each component/dimension extensions on the three dimensions can separately or combined contribute to the total functionality of the semantic wiki. Hence, an extensible semantic wiki architecture should allow for (independent) extension of these three dimensions. If the core functionality of the extensible semantic wiki nearly fits the requirements single dimensions can be extended denoting "refining" extensions. Heavy-weight extensions along all three dimensions might have their own language, reasoning and presentation functionality. To clarify the threefold distinction in a more practical context we present three extensions along three, two and one dimensions respectively in Section 3.

2.2 Decorating Semantic Wikis

As already mentioned in the introduction we claim, that a semantic wiki system should be precisely tailored to a semantic wiki application considering the domain, community, and use-cases. This method is in compliance with the ideas presented recently by Yaron Koren at the semantic wiki mini series[1]. One can assume that there are many domains where semantic wiki technology could be employed beneficially. One must not assume that every possible user in any domain is able *and* willing to get used to concepts like *Semantic Wiki*, *ontology*,

[1] http://ontolog.cim3.net/cgi-bin/wiki.pl?ConferenceCall_2008_12_11

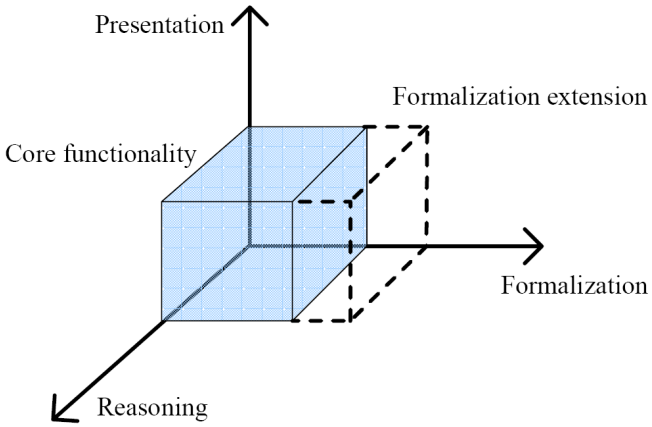

Fig. 3. The semantic wiki extension space.

RDF-triple, *SPARQL* or *DL-Reasoning*. Nonetheless it is possible to create semantic wikis that allow for efficient knowledge sharing and use for these user groups at the cost of some customization. The intended use cases and the user's mental model of the knowledge need to be identified in advance. Then, the ontology capturing the knowledge that is necessary to support the use cases can be modeled. Further, a method for knowledge formalization (e.g., user and domain specific markup) is designed that fits the mental model of the users. This markup or editing component implicitly creates (possibly complex) RDF-structures. At last an extension of the knowledge presentation component is implemented. This extension exactly supports the use cases revealed be the requirements analysis. It executes the necessary calls of the reasoning engine and presents the result in a visualization matching to the mental model of the user. One example might be some buttons with underlying predefined SPARQL queries with result sets rendered in some (domain specific) visualization. Any technical details (e.g., property names, creation of triples, SPARQL queries) are completely hidden from the user. The typical extension pattern for decorated semantic wikis is shown in Figure 4.

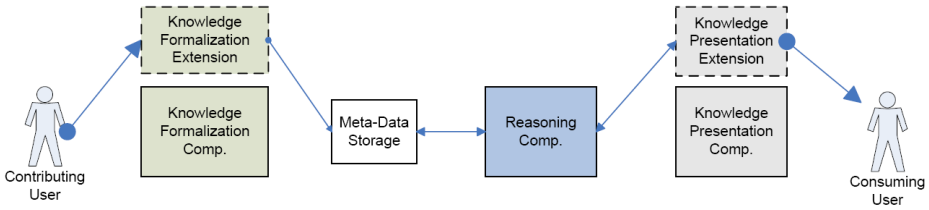

Fig. 4. Extension pattern of a "decorated" semantic wiki application.

The core functionality for knowledge formalization and knowledge presentation should be hidden from the untrained user to reduce confusion. This approach has been implemented on the HermesWiki project that is described in more detail in Section 3.

2.3 Challenges towards an Extensible Semantic Wiki

In the following we discuss the three most important aspects when designing an extensible semantic wiki:

1. **Basic functionality:** In order to allow powerful extensions with advanced features with little implementation costs, it is necessary to decide what semantic core-functionality comes along with the basic semantic wiki architecture. Enabling applications that have to deal with large data sets and high user activity, requires a slim and scalable text-processing and reasoning engine. Reasoners that focus on processing of more expressive or inconsistent knowledge are often consuming more computational power and need to be included by an extension if necessary.
2. **Usability:** One of the most important reasons for the wide acceptance and success of wikis is their high usability and the low training costs for new users. Semantic wikis are bringing new possibilities to wikis and thus are inevitably adding some complexity to its usage. Adding these new functionalities to the Wiki interface with the least mental overload is a critical issue in semantic wiki design. The core strength of the Wiki approach being simple otherwise can easily vanish. In every case it is sensible to enable 'non-semantic' users to work like in a 'non-semantic' wiki with no adaptation allowing them to discover single additional functionalities step by step. We propose the idea of a 'shadow'-semantic wiki hiding all of its advanced functionality at the beginning. Advanced features (e.g., fact sheets, meta-data browsers) can be added to pages using tags or configuration settings by more experienced users when necessary.
3. **Extension mechanism:** Various extension mechanisms on the software engineering level are applicable to create feature-rich and flexible software. However, there are several challenges specific to the context of semantic wiki functionality. The mechanism should be able to support light-weight "refining" extensions in a very simple way and at the same time still allow for complex extensions (e.g., with own markup, meta-data representation, reasoners and result visualization). In general, an unpleasant issue in modular software engineering in general is the (programming-)language barrier. For technical reasons combinations of software components written in different programming languages are often insufficiently manageable and inefficient. Unfortunately, this poses some kind of barrier for employing various implementations of semantic technologies to an extensible semantic wiki architecture.

The best solutions to all these aspects cannot be stated straight forward containing several trade-offs that might have to be revised after future experiences.

3 The Extensible Semantic Wiki KnowWE

We have designed and implemented the extensible semantic wiki *KnowWE* (Knowledge Wiki Environment) aiming to the concept of extensibility as stated in Section 2. As many software components on NLP and reasoning engines are implemented in Java, it appears helpful having our system also implemented in this language. We especially focused on the simple definition of new markups and tools allowing for easy text-based refactoring of these. In this chapter we introduce the core functionality of KnowWE. Additionally, three existing extensions are presented to demonstrate our approach.

Currently, KnowWE runs on top of JSPWiki[2], but it can be easily used with any (java-based) wiki engine having an extension mechanism similar to JSPWiki's *PageFilter* concept. Only the wiki connector providing page- and user-management has to be reimplemented accordingly.

3.1 The KnowWE Core

In KnowWE each article content is held in a tree-based data-structure. This tree can be used for refactoring, rendering (e.g., personalization, syntax highlighting), navigation, external editors and also allows the mapping between the textual content and the extracted meta-data. Arbitrary semantic extensions along any dimensions can be created without touching the core components of KnowWE. For any declared formalization extension (e.g., custom markups) the parse-tree is generated and hang up into the article content tree.

To provide a flexible library of reusable components, we integrated tools that are valuable for a wide range of possible scenarios managing formalized knowledge in a Wiki:

- A semantic renaming tool and an annotation browser which is demonstrated Section 3.4
- Table editing functionality allowing for visual editing of structured data.
- A flexible include mechanism for reusing arbitrary page snipplets across wiki pages for modular content and knowledge management.
- Parsing components for table-based, line-based, xml-based and regex-based markups to simplify the declarative definition of markup without or low efforts on parsing functionality.

Without any extensions, KnowWE comes with a set of basic functionality comprising the general features of a semantic wiki. To include formalized knowledge the KnowWE core version provides simple markup and the possibility to import knowledge from external sources. New properties can also be introduced ad-hoc to the system by the use of property definition sections on every wiki page. The properties defined within these tags are not created as standard OWL properties but rather as N-ary relations[3]. This allows us to automatically add

[2] http://www.jspwiki.org
[3] http://www.w3.org/TR/swbp-n-aryRelations/

supplementary information to the created knowledge. One of the automatically added nodes is a *TextOrigin* node that contains a reference to the textual position within the wiki text where the annotation was made, as well as revision information of the annotation's creation. There are several predefined properties which can be used to create annotations. Those key properties like `subClassOf`,`type` and `subPropertyOf` are treated separately from the user created properties. They are imported into the wiki-knowledge as their RDFS counterparts, allowing the reasoner to work on the generated knowledge without further translations.

An annotation is created by a simple link-like markup syntax. In its basic form an annotation is similar to the markup used in Semantic MediaWiki [2]. The following example is taken from a consumer domain describing digital cameras and shows the annotation of the concept associated with the local page by the property *hasBrand* with the value *Canon*.

```
Canon released the new 50D [hasBrand::Canon] a while ago.
```

In this form the subject of the annotation's property is the default concept of the page. Beside the double square brackets the differences to a Semantic MediaWiki annotation is that the annotation in this form has no explicit textual representation in the page view. Thus there is no Link to *Canon* created in the text but the formal knowledge is attached to the preceding word in this case. The markups can be extended by optional components like a subject different from the current page concept. Another way of modifying an annotation is by including a specific piece of text to be annotated. The following example shows the annotation of the text phrase "new camera" by a formal relation connecting the camera instance *Canon EOS 50D* and the brand instance *Canon* by the *hasBrand* relation.

```
Canon released the [new camera <=> Canon EOS 50D hasBrand::Canon]
a while ago.
```

This additional syntax provides more flexibility compared to the standard annotation. It allows to define relations outside the wiki page representing the instance. Although recommended in KnowWE, it is technically not necessary that every concept has its own wiki page. Further, it allows for precise attachment of a formal relation to a text phrase for documentation. This also enables queries to find all text phrases that for example were annotated with a *hasBrand* relation. KnowWE also provides a basic fact sheet markup to define multiple relations in a compact manner.

Our wiki uses the Sesame RDF storage[4] for saving and querying the created knowledge. The reasoning capabilities of our system are provided by the OWLIM engine (version 3.0b9)[5]. The default setting on the reasoning level is owl-max providing full RDFS reasoning as well as OWL-Lite semantics.

[4] http://www.openrdf.org
[5] see http://www.ontotext.com/owlim/

The reasoning and the N-ary representation of properties provide the background to answering SPARQL queries on the wiki ontology and imported ontologies. A SPARQL query is embedded into a wiki page using the simple XML-tag *sparql*. The query itself is being processed by the sesame query engine and the results are rendered in a table on the wikipage. For example, the following rather simple query produces a list of all concepts, in this case all cameras which are produced by `Canon`.

```
partial product list described in this wiki:
<sparql render="links">
SELECT ?cam
WHERE {
?t rdf:subject    ?cam  .
?t rdf:predicate  lns:hasBrand .
?t rdf:object     lns:Canon .
} LIMIT 5
</sparql>
```

The abbreviation `lns` is replaced by the local namespace which is constructed from the url of the wiki installation. The results are rendered as links to the pages where they are defined as shown in Figure 5. Omitting the `render` attribute creates a simple table of the Canon products.

Fig. 5. The result of a simple query for Canon products.

3.2 The d3web Extension

The d3web extension is a KnowWE extension to enable the definition and use of classification knowledge. We outline this extension only briefly since it has previously been described in detail [9].

– **Knowledge formalization extensions:** Different markup languages have been developed to capture various types of classification knowledge, e.g., covering models, rules, decision trees [16]. Here, the KnowWE architecture

allows for the integration of context sensitive editing support and syntax-highlighting. The textual markup is compiled into two different representations. First it is transformed into the proprietary object structure that is used by the integrated d3web[6] reasoning engine. To exploit querying capabilities, it additionally is compiled into an OWL-ontology using an upper ontology for classification which is explained more detailed in [17].
- **Reasoning extensions:** The d3web project summarizes a set of reasoners for diagnostic problem-solving. The reasoning engine is employed in this extension to work on explicit knowledge created by the use of the markup languages described above.
- **Knowledge presentation extensions:** Beside visualization and browsing mechanisms for the formalized problem-solving knowledge several possibilities for the execution of knowledge bases are included. A user can initiate a problem-solving session either by starting a structured interview or by freely answering questions that are rendered in the wiki pages. After each entered answer the currently most appropriate solutions (calculated by the d3web reasoning engine with the knowledge base) are displayed. Such a problem-solving session can be considered as an incremental personalized query to a classification system.

The d3web extension represents a heavy-weight form of a semantic wiki extension coming along with libraries containing reasoners, parsers and dialog components. However, this extension is especially intended for small to medium sized data sets and small user communities.

3.3 The HermesWiki Extension

The HermesWiki is a KnowWE-based Wiki in the historical domain developed in German language. It is built in cooperation with historians from the University of Würzburg. The main purpose of the HermesWiki is to provide an overview on ancient Greek history for teaching purposes of (undergraduate) students. Additionally, the Wiki provides direct links from the descriptions of historical events to translations of their (historical) sources. The Wiki consists of three parts: A collection of about twenty essays giving a comprehensive domain walkthrough, translations of the describing ancient sources, and an extensive glossary. Entries in the glossary are semantically tagged, e.g., as "politician" or "poet". The project, started in summer 2008, currently features more than 500 wiki articles, often illustrated with maps and pictures.
Technically, the wiki implements extensions along the dimension of formalization and presentation. The most important formalization extension of the HermesWiki is a specialized markup to explicitly define historical events in the main essay articles. This markup was developed in cooperation with the historians to allow for maximum usability in this community. Each historical event is defined in its own text block, structured as the following example:

[6] www.d3web.de

```
<<Lamian War (2)
323b-322b

After Alexanders death the Greeks revolted against Macedonian rule
under the lead of the Athenians.
[...]

SOURCE: Paus.::1.25.3-6
SOURCE: Diod.::18,8-18
>>
```

Each event is enclosed with double angle brackets. In the first line the title of the event ("Lamian War") is given, followed by a single number in parentheses, which describes the importance of the event. For example, events with an importance rating of '1' are considered essential while events with a rating of '3' are categorized as "additional knowledge". In the second line of the markup the point or interval in time when the event occurred is noted in a compressed form, e.g., the string "323b-322b" points out, that the event occurred from 323 BC until 322 BC. Further annotations can reflect more precise dating as well as uncertainty. After one empty line a free text description of the event follows. At the end of each event block ancient sources of the event are mentioned, explicitly marked by the keyword "SOURCE:" as the first word of a new line. This markup for historical events is currently used around 600 times in the HermesWiki. A time event will be modeled in OWL using a small ontology containing a class for timeline events with the properties as *hasImportance*, *hasTimeDesignation*, *hasDescription* and *hasSource*. Having this information extracted to OWL allows for several forms of exploitation in the presentation dimension. HermesWiki can generate different views on the timeline events by filtering them on constraints regarding the time, in which an event occurred, on event importance, and on the article, where an event was defined or on sources occurring. Figure 6 shows an exemplary generated view on the timeline in the HermesWiki featuring parts of the conquests of *Alexander the Great* (in German language). The importance of the events is color coded. One (domain-specific) use case supported by this extension is the (semantic) navigation "through the time" using the links provided by the ordered time line views. This Wiki project demonstrates how a rather light-weight domain specific extension along the dimensions of formalization and presentation can yield a "decorated" wiki as described in Section 1 that is usable by domain specialists which are not familiar with semantic techniques in general.

3.4 A POS-Tagger Extension

To demonstrate the possibilities achieved with low efforts extending KnowWE we have implemented a small toy-extension to enable Part-Of-Speech-Tagging. Further, it serves as a tutorial for the KnowWE extension mechanism. To point out the simple integration of NLP-tools and libraries we employed the Stanford

Fig. 6. A generated timeline in the HermesWiki

POS-tagger[7] to analyze the wiki content. Figure 7 shows the resulting wiki view with this extension activated.

In this example configuration the extension calls the POS-tagger and marks all verbs found as *VerbType* nodes in the content tree. To show the results in the wiki a yellow highlighting-renderer is attached to the node type *VerbType*.

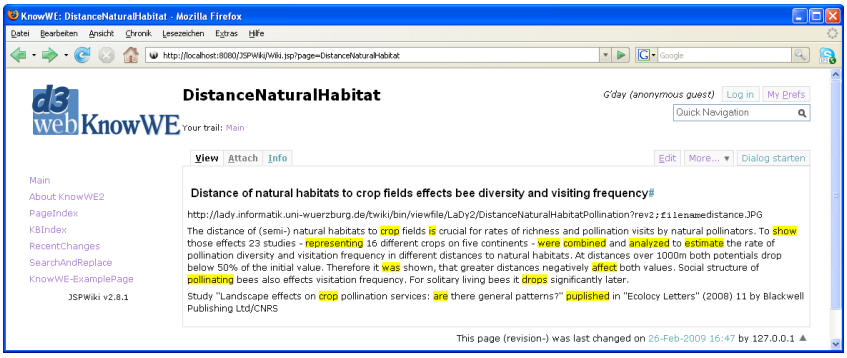

Fig. 7. The view of a wiki article with the POS-Tagging-Demo extension enabled.

This extension, consisting of a few lines of code calling a POS-Tagger library, enables to use the following tools coming along with KnowWE:

[7] http://nlp.stanford.edu/software/tagger.shtml

The annotation browser This tool shown in Figure 8 allows for browsing the wiki by annotation-types. For each annotation a link to the occurrence in the wiki page is provided. The column *ancestors* shows the types on the path of the content tree depicting the semantic context of the finding, which is of course *TaggingDemo* in this example. In this verb-tagging scenario this could be used to review the POS-tagging results manually. The browser can also be extended to support a two-step semi-automated workflow, where users can confirm or dismiss annotations to create verified meta-data. In general, this tool can serve as a statistical overview on formalized content.

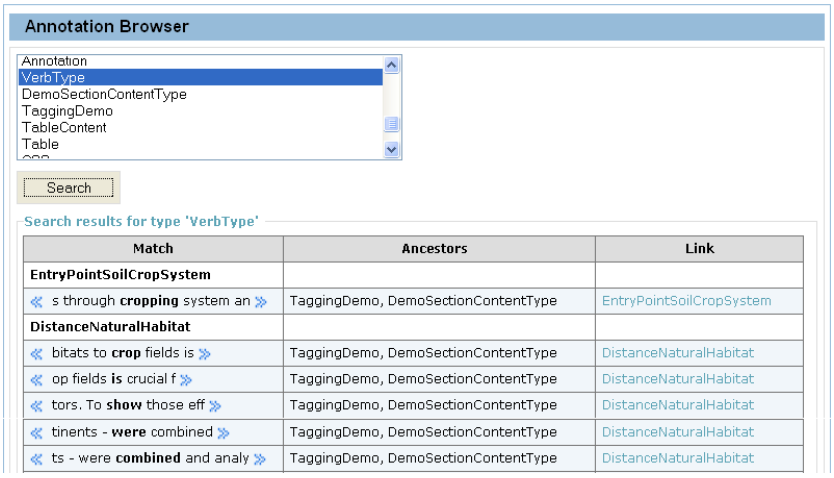

Fig. 8. Annotation browser listing all existing annotations in the wiki for a selected type

The semantic renaming tool The semantic annotation given to words by the POS-Tagger can be used for text refactoring. The semantic renaming tool is basically a standard global search and replace tool but additionally enabling semantic type-filtering. Thus, we can globally rename a string, but only if the string has a certain "role" in a particular markup, that is *VerbType* in this example. One use case which is enabled by this simple single-dimensional extension (Formalization) is that one could start some disambiguation efforts on the given content (e.g., for further processing by other NLP-techniques). Since this article is about crop fields we might like to rename the verb "crop" to a synonym like "cut" or "truncate" ensuring no confusion between the verb "crop" and the noun "crop" can arise. The renaming tool allows for sorting and filtering the replacements of the searched string by annotation-type and by articles. One can search for the string "crop" and all occurrences in the wiki are listed with annotation-context. Thus, one can select to replace all occurrences that are annotated as

VerbType in a subset of Wiki articles. In this way all nouns of "crop" stay untouched - this of course only works assuming that the POS-Tagger successfully separated the verbs from the nouns.

This semantic renaming tool working on all installed formalization extensions of a KnowWE system forms a basic refactoring tool for knowledge at different degrees of formality tagged by various formalization techniques.

4 Discussion

In this paper we motivated and discussed the concept of an extensible semantic wiki architecture. As an example implementation we introduced our extensible semantic wiki KnowWE and presented some of its current extensions. In our work we focus on supporting complex markups and the interconnection of formalized and textual content. The support of refactoring methods on the semi-formalized contents is the subject of our current research. The probably most popular semantic wiki Semantic MediaWiki shows numerous extensions on formalization and presentation. It is employing the "Decoration" pattern in different applications by the use of extensions like *semantic templates*, *semantic forms* and *semantic queries*. We think, that project-oriented customization of the tools towards the needs of the targeted domain and user community will become a more and more important challenge in the future. To demonstrate and evaluate our approach on this task we presented some case studies. Beside the HermesWiki project we are also applying this approach to the *BIOLOG* [8] project in order to optimally support the management of biological knowledge collected there. In this project we plan to use KnowWE to develop methods for semi-automated knowledge formalization using the agile employment of ontology learning methods. Considered from the other side, for researchers developing such ontology learning methods semantic wikis are attractive as an evaluation platform if simple integration of these methods is possible. We hope, that extensibility in general will reduce the setup-costs of semantic wiki solutions and therefore help to further establish this technology.

We will drive the KnowWE implementation towards even more flexibility and stability. For the latest news we refer to the project page of KnowWE on sourceforge[9].

References

1. Schaffert, S., Gruber, A., Westenthaler, R.: A semantic wiki for collaborative knowledge formation. In: Proceedings of SEMANTICS 2005 Conference, Trauner Verlag (2006)
2. Krötzsch, M., Vrandecić, D., Völkel, M.: Semantic MediaWiki. In: ISWC'06: Proceedings of the 5th International Semantic Web Conference, LNAI 4273, Berlin, Springer (2006) 935–942

[8] www.biolog-europe.org

[9] http://sourceforge.net/projects/knowwe/

3. Schaffert, S.: IkeWiki: A semantic wiki for collaborative knowledge management. In: STICA'06: 1st International Workshop on Semantic Technologies in Collaborative Applications, Manchester, UK (2006)
4. Buffa, M., Gandon, F., Ereteo, G., Sander, P., Faron, C.: : A semantic wiki. Web Semantics **8**(1) (2008) 84–97
5. Auer, S., Dietzold, S., Riechert, T.: OntoWiki – A Tool for Social, Semantic Collaboration. In: ISWC'06: Proceedings of the 5th International Semantic Web Conference, Berlin, Springer (2006) 736–749
6. Kuhn, T.: Combining Semantic Wikis and Controlled Natural Language. In Bizer, C., Joshi, A., eds.: Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC2008). Volume 401., CEUR Workshop Proceedings (2008)
7. Lange, C., Kohlhase, M.: A semantic wiki for mathematical knowledge management. In Völkel, M., Schaffert, S., eds.: Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics. Workshop on Semantic Wikis, ESWC2006 (June 2006)
8. Nalepa, G.J., Wojnicki, I.: Proposal of a prolog-based knowledge wiki. In Nalepa, G.J., Baumeister, J., eds.: KESE. Volume 425 of CEUR Workshop Proceedings., CEUR-WS.org (2008)
9. Baumeister, J., Puppe, F.: Web-based Knowledge Engineering using Knowledge Wikis. In: Proceedings of Symbiotic Relationships between Semantic Web and Knowledge Engineering (AAAI 2008 Spring Symposium). (2008)
10. Wagner, C.: Breaking the knowledge acquisition bottleneck through conversational knowledge management. Information Resources Management Journal **19**(1) (2006) 70–83
11. Kaljurand, K.: ACE View — an ontology and rule editor based on Attempto Controlled English. In: 5th OWL Experiences and Directions Workshop (OWLED 2008), Karlsruhe, Germany (26–27 October 2008)
12. Krötzsch, M., Schaffert, S., Vrandecic, D.: Reasoning in semantic wikis. In: Reasoning Web. (2007) 310–329
13. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05), Edinburgh, Scotland (August 2005)
14. Ma, Y., Hitzler, P., Lin, Z.: Algorithms for paraconsistent reasoning with owl. In Franconi, E., Kifer, M., May, W., eds.: The Semantic Web: Research and Applications. Proceedings of the 4th European Semantic Web Conference, ESWC2007, Innsbruck, Austria, June 2007. Volume 4519 of Lecture Notes in Computer Science., Springer (JUN 2007) 399–413
15. Klinov, P., Parsia, B.: Pronto: Probabilistic ontological modeling in the semantic web. In: Proceedings of the Poster and Demonstration Session at the 5th European Semantic Web Conference (ESWC2008). Volume 401 of CEUR Workshop Proceedings., CEUR-WS.org (2008)
16. Baumeister, J., Reutelshoefer, J., Puppe, F.: Markups for Knowledge Wikis. In: SAAKM'07: Proceedings of the Semantic Authoring, Annotation and Knowledge Markup Workshop, Whistler, Canada (2007) 7–14
17. Reutelshoefer, J., Baumeister, J., Puppe, F.: Ad-hoc knowledge engineering with semantic knowledge wikis. In: SemWiki'08: Proceedings of 3rd Semantic Wiki workshop - The Wiki Way of Semantics (CEUR Proceedings 360). (2008)