Annotation Component in KiWi

Marek Schmidt and Pavel Smrž

Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
E-mail: {ischmidt,smrz}@fit.vutbr.cz

Abstract. This paper deals with key functionalities of the KiWi annotation component and shows how it enables seamless combination of informal and formal knowledge and transformation of the former to the latter. It demonstrates how the advanced KiWi features, such as nested content items, reasoning and information extraction, can be used together to make rich semantic annotation easy and useful.

1 Introduction

The original wiki systems employ specific wiki languages to edit content. Such languages can easily be extended to allow semantic annotations, which is the approach taken by various semantic wiki systems, such as Semantic MediaWiki [2]. Other approaches to semantic data editing, as, e.g., in OntoWiki [1], provide a rich interface to edit RDF. However, these annotations are not integrated into wiki text content. In KiWi [4], we combine semantic annotations directly with the text content of the wiki pages and provide advanced user interfaces supporting the annotation process with the help of suggestions coming from information extraction.

2 Knowledge Representation in KiWi

KiWi data model is designed to integrate both formal and informal knowledge [4]. A core entity is a *content item* which may contain an XHTML text content or any other kind of multimedia. A title and a list of tags are associated with content items. Each content item corresponds to exactly one *resource*, which enables adding arbitrary RDF statements about the content item. The 1:1 relationship between a content item and a resource reflects a usual practice in semantic wikis. However, having only this linking mechanism is limiting as it is then not possible to represent formal statements about other entities than the current page. Some wikis, such as Semperwiki [3], allow defining an *about* entity, independent of the page, to describe other entities than the current page.

The KiWi nested content items and fragments allow for a more granular and more natural annotation. Fragments enable annotating arbitrary segments of text with arbitrary tags, comments and RDF metadata, which is akin to

annotating a paper with a marker, enhanced with semantics. Nested content items are used for annotating whole sections of text with arbitrary metadata. While no explicit *about* resource as in Semperwiki is supported in KiWi itself, such behaviour can be implemented in KiWi using the native KiWi reasoning support by creating rules. It is thus possible to define an 'about' rule, such that nested item would act as a proxy for a different resource, and any RDF triple assigned to the nested item could automatically be inferred on the referenced resource.

3 User Interface for Information Extraction

The information extraction service in the KiWi system uses natural language processing and machine learning algorithms to provide suggestions for annotations [5]. There are two ways users can interact with the information extraction services in KiWi.

3.1 ASIDE – Annotate Single Document Efficiently

Users can create and edit all kinds of annotations supported by the KiWi system mentioned in the previous section. The information extraction component supports the user by displaying suggestions.

Suggestions can be applied at various stages of the annotation process. Some suggestions can be shown directly in the text, so that the user can select the piece of text just by clicking on the suggestion.

When the user makes a selection of the piece of the text, all the suggestions relevant to that piece of text are displayed. This may include more suggestions than the previous step, as an additional information extraction step is taken at this time which employs apriori information on selecting the particular piece of text.

To support emerging knowledge, it is also crucial to support partially specified annotations (such as a link to an entity of which only type is known, but no entity to link to exists yet), or annotations that conflict with the current ontology (such as an object predicate linking to an entity of a wrong type). The user interface shows the partial annotations in yellow and erroneous annotations in red.

Some suggestions can be ambiguous, such as a suggestion for a link to a user page based on the user names. The annotation can be directly created from these kinds of suggestions, but it will be marked as partially specified, so the user sees that additional action is necessary to make this annotation into a 'green' correct one.

The suggestions can also be displayed in a list sorted by type. The suggestion list includes properties which are defined for the current content item type, but for which no suggestions have been found in the document. A user can thus see if there are some of the required annotations missing. Then, she can annotate just by dragging a selected piece of text and dropping it to a particular type box

to create an annotation of this type. List of types for the current content item is generated from the underlying ontology.

3.2 AMUSE – Annotate Multiple Documents Simultaneously (and Efficiently)

Especially when dealing with a new task, it is often the case that one needs to semantically enrich many documents of the same type, e.g., a bunch of minutes from a series of previous meetings. The use of the ASIDE tool introduced in the previous subsection on each individual wiki page would mean a tedious work. In these situations, it is preferable to focus on a specific type of annotations and process all the documents in one run.

AMUSE is a kind of discovery tool intended to identify all instances of the given type in all the documents available. This tool is also used to configure the information extraction services and to 'tune' it with respect to the particular type of annotation being extracted.

Machine learning algorithms are employed to classify potential instances. AMUSE takes advantage of existing annotations found in the initial training data and retrains the classifiers on the user feedback (accepting or rejecting suggestions).

The behaviour of the tool depends on the type of entity it is used on:

- *Types.* Identify all the pages of the given type, based on document classification. In addition to document features, contextual features derived from the links to pages of the given type are used for classification.
- *Tags.* Same as for types, but additionally also discover all the text fragments that should have this tag.
- *Datatype properties*, such as 'foaf:birthday'. Classify all the fragments of the particular type. A specific extractor can be assigned to each of these kinds of extractions (such as a date extractor for recognizing date information from text, money extractor to recognize amounts of money in a listed currencies, etc.)
- *Object properties*, such as 'foaf:currentProject'. Discovers links to entities and their roles. It works in combination with the type classifier to recognize roles of the potential links.
- *Other entities.* Discover links to this entity from other pages. This may involve disambiguating titles shared by several pages.

After the initialization of this tool for a specific entity (type, tag or property), AMUSE displays a ranked list of suggestions coming from various content items. Users can immediately accept or reject the suggestions, thus annotating the content items and improving the system by providing the training examples at the same time.

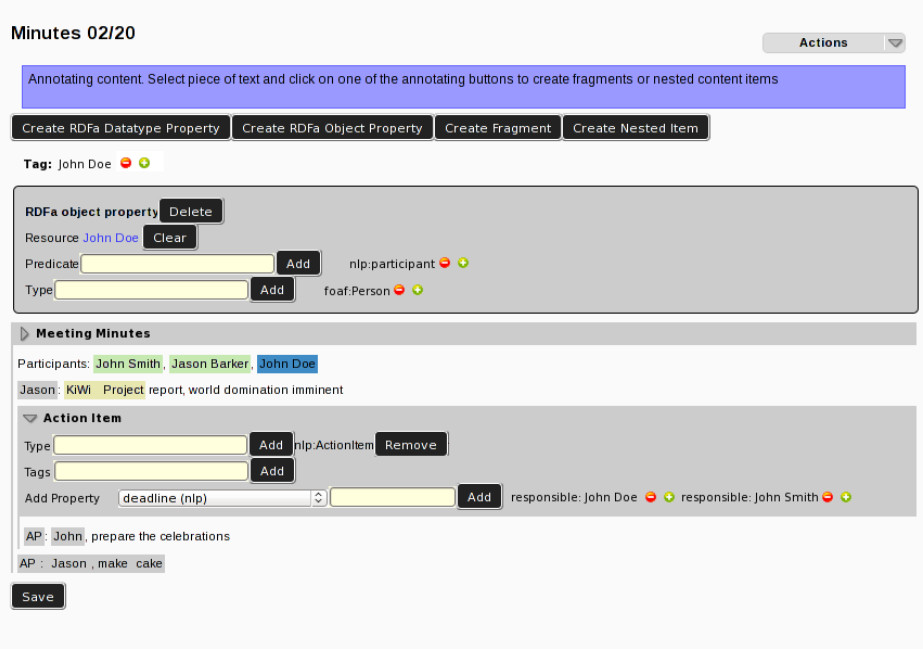

Fig. 1. The annotation tool, annotating a meeting minutes document. Currently editing a link to an entity named 'John Doe'.

4 Use-case Scenario

The scenario discussed in this subsection corresponds to an enterprise setting. A semantic wiki is used to facilitate the knowledge formalisation process in project management tasks. Various kinds of information need to be formally represented in the knowledge base, such as information about projects, customers, people, resources, meetings and tasks. This data can then appear in simple queries ('who attended the meetings where project Foo was discussed'), better task management (tasks can be formally defined directly in the meeting minutes document and automatically appear in the responsible person's 'todo' lists and calendars). This scenario assumes that an ontology describing the entities and their relations already exists in the system.

As demonstrated by Figure 1, meeting minutes are produced in the KiWi system. The annotation tool is opened. The system immediately offers suggestion regarding the type of the document. Selecting the proper type leads to more relevant suggestions. The information extraction component recognizes some of the names of people present and correctly offers the role 'participants'. One of the names could not be identified, because this user was not mentioned yet in the system. It is still recognized that the string corresponds probably to a name of a person, though, so a suggestion to create a new entity of the type 'foaf:Person' is

displayed. Accepting the suggestion creates a new entity in the knowledge base. It will be recognized in all further documents.

Some of the other recognized entities are irrelevant for the current context (such as matching general terms in the ontology), so the user rejects these suggestions. The provided feedback instructs the system not to offer these suggestions in similar contexts in future steps. The user accepts one other suggestion triggered by a label of one of the projects. The meeting is now formally associated with the project.

The user can create another annotation, such as selecting a piece of text around an action item and clicking the Nested Content Item button and selecting the type ActionItem. The ActionItem class specifies several properties, such as deadline and responsible persons. The user can fill the responsible person property just by dragging-and-dropping one of the person annotations created earlier. The task will automatically appear in the task list of the responsible person after the automatic application of the appropriate reasoning rule.

5 Conclusions and Future Directions

The annotation component introduced in this paper has become an integral part of the KiWi system. It enables formal semantic annotation of any kind of existing content. Information extraction supports the annotation by providing context-dependent suggestions which are naturally integrated into the user interface. The discussed use-case scenario shows advantages of the implemented tools in realistic conditions.

Our future work will focus on merging the annotation tool and the KiWi editor and on displaying the suggestions at real time while editing the content. We will also continue to collect real use data to quantify the actual improvement in the annotation process given by the suggestions.

Acknowledgement The research has received funding from the EC's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 211932.

References

1. Auer, S., Dietzold, S., and Riechert, T. Ontowiki-A tool for social, semantic collaboration. *Lecture notes in computer science 4273* (2006), 736.
2. Krötzsch, M., Vrandecic, D., and Völkel, M. Semantic mediawiki. In *ISWC* (2006), vol. 6, Springer, pp. 935–942.
3. Renaud, E. O., Delbru, R., Möller, K., and Völkel, M. Annotation and navigation in semantic wikis. In *SemWiki* (2006), p. 29.
4. Schaffert, S., Eder, J., Grünwald, S., Kurz, T., Radulescu, M., Sint, R., and Stroka, S. KiWi–a platform for semantic social software. In *Proceedings of the 4th Workshop on Semantic Wikis, European Semantic Web Conference* (2009).
5. Smrz, P., and Schmidt, M. Information Extraction in Semantic Wikis. In *Proceedings of the 4th Workshop on Semantic Wikis, European Semantic Web Conference* (2009).