One Click Annotation

Ralf Heese, Markus Luczak-Rösch, Radoslaw Oldakowski, Olga Streibel, and Adrian Paschke

Freie Universität Berlin, Institute for Computer Science,
Corporate Semantic Web, D-14195, Germany,
{heese,luczak,oldakowski,paschke}@inf.fu-berlin.de

Abstract. The acceptance and the dissemination of Semantic Web technologies depends on the availability of authoring tools that enable ordinary Web users (i.e. non-experts with respect to semantic technologies) to create and publish semantic content. In this paper, we introduce the *One Click Annotator*, a WYSIWYG Web editor for enriching content with RDFa annotations. An intuitive user interface hides the complexity of creating semantic data. We discuss requirements, challenges, and solutions of the implementation. In large parts the editor is already available as a prototype on the semantic content authoring system *loomp*.

1 Introduction

Gathering information on a topic of interest from the Web is a time-consuming task, because the users are required to discover data sources, to filter, and to integrate information manually. The Web 2.0 movement changed Web into an information interchange platform helping ordinary Web user to ease and partially to automate the discovery process. Along with this development information chunks published online become smaller and are part of larger interconnected data silos or information clouds being accessible by Web 2.0 portals. This is, for example, emphasized by the recent success of social networks and micro blogging.

Semantic technologies, and in the first place linked data, promise further automation by turning the Web of information into a Web of interconnected and machine processable data sources. Although these technologies have reached an acceptable mature state, they are not broadly used in commercial and public Web 2.0 applications. In our opinion this is mainly caused by the lack of user-tailored and easy-to-use tool support for creating and publishing semantic contents.

With the One Click Annotator (OCA) we address the issue of missing tool support for creating semantic content by non-expert users. The OCA is an editor for Web browsers which allows for annotating words and phrases with references to ontology concepts and for creating relationships between annotated phrases. Although you may notice that a user needs at least two clicks to create an annotation, we nevertheless call it *One* Click Annotator, because we want to stress the simplicity of the annotation process. Our main design goal is to realize a tool that non-expert users can easily use to create semantic content: It provides a clear and intuitive user interface, presents only simple and well-described ontologies/vocabularies, and lets the user focus on the task of writing text.

The OCA interacts with a server to process and to store the semantic content as well as to answer queries about resources occurring in the edited content. For this reason, we currently develop the loomp platform [?] which is a lightweight content authoring system for creating, managing, and publishing semantic content. It offers the necessary functionalities for running the OCA.

The paper is structured as follows: In Section 2, we motivate our ideas by a use case in the domain of journalism, because we believe that people working in this area can obtain a great benefit. Afterwards, we focus on the One Click Annotator in Section 3 and discuss requirements, challenges, and solutions. In Section 4, we give an overview of our implementation and the interaction of the OCA and the loomp platform. In Section 6 we conclude and outline future work.

2 A Motivating Use Case

The development of loomp was motivated by a use case in the domain of journalism which is representative for the domain of content intensive work in a heterogeneous environment. Studies such as [?] recognize an increasing importance of online publishing and claim for more professional journalists in online media. Today, journalists research specific topics on demand and access various information sources for this purpose, e.g., websites, articles, and human informants. Personal interviews with journalists and publishing houses revealed that they note the research results using paper and pencil. Few journalists use digital devices for this task and even fewer apply information management systems. To transfer the finished article to the editor at the publishing house the people use free text documents and email communication or they enter their articles directly into editorial managements systems or content management systems.

Journalists can use a combination of OCA and loomp as a personal information management system managing semantically enriched notes, interview logs, references, and articles. They can make use of a semantic search on their own content and the content provided by the Web of linked data, e.g., find all articles about "president Bush" who was in office from 1989 to 1993. Additionally, loomp supports linking resources (e.g., notes to the final article) and, thus, facilitates the reuse of articles or parts of them. On the side of publishing houses content already annotated by journalists allows for offering value-added services to their customers without additional effort. For example, the customer can highlight all phrases being an instance of a certain concept (e.g., all names of rivers). Using loomp editors can add further annotations to articles received from journalists and publish them on different channels, e.g., blog, RSS feed, wiki, or print.

3 One Click Annotator

With the One Click Annotator (OCA) we aim at providing tool support for annotating content semantically. We consider non-experts having little or no knowledge of semantic technologies as the primary target group, because we believe that the success of the Semantic Web depends on reaching a critical mass of users creating and consuming semantic content. Thus, the annotator has to hide the complexity of creating semantic content by providing an intuitive user interface. To realize such an interface, the OCA follows common mindset as the underlying domain model and uses well-known components of existing user interfaces. It is implemented as a component that can be used independently of loomp, e.g., it could be integrated into a wiki. However, a server has to exist which implements the OCA programming interface. Thus, we describe in the remainder the design of the OCA as a client communicating with some server.

3.1 Requirements

The main driver for our requirement analysis is to enable *non-expert users* to annotate their content semantically. "Non-expert" means that a user is able to use a computer, e.g., select some text, click a button, or tag content, but he has no or little knowledge about semantic technologies such as RDF, ontologies, and linked data. In our opinion, the compelling simplicity of Web 2.0 applications should be transferred to the task of creating semantically rich content: light-weight, easy-to-use, and easy-to-understand. In the following list, we focus on design requirements on a semantic annotation tool for non-experts.

Intuitive user interface. The annotator hides the complexity of creating semantic annotations by providing an intuitive user interface. It follows common mindset and reuses procedures of system interaction known from word processors to produce the annotations.

Simple vocabularies. Although Web users know the term URL or Internet address, most of them are not aware of namespaces. Thus, the annotator provides access to vocabularies without going into technical details. Each concept has a meaningful label and is described in simple terms.

Focus on the user's task. Usually, a user wants to perform the task of writing some text and not to annotate content. Thus, the toolbar for creating semantic annotations is seamlessly integrated into a text editor, so that the user can add annotations without distracting himself from his primary task.

Interoperability. To minimize the problems of interoperability the annotator is build on top of standards, e.g. to represent and to access annotated content. Regarding vocabularies, the annotator provides access to widely accepted vocabularies and is able to map concepts of equal or similar meaning.

Comparing these requirements to existing approaches and tools we think that there is still large room for improvement. For example, semantic wikis (e.g., OntoWiki [?] or semantic MediaWiki [?]) rely on a proprietary wiki syntax to represent semantic annotations in the text. As a result, a user has to learn additional syntax and to know the vocabularies. The other extreme consist of annotation services such as OpenCalais [?] that automatically annotate entities in a given text. These allow little or no human influence on the added annotations.

Finally, before we started to develop the OCA we made the following two key design decisions on the implementation: The OCA has 1) to work with modern

Web browsers, and 2) to store the annotated content and the annotations itself in a format that corresponds to widely accepted Web standards. As a consequence the OCA is independent of a certain backend implementation and can easily be integrated into other Web applications. Due to the requirements we were confronted with the challenges as compiled in Table 1 and discussed below.

Challenge	**Solution**
Clear and intuitive UI	Transfer the look&feel of the toolbar for style sheets known from word processors.
Represent annotations as RDF	Selected text is the value of a property assigned to a resource.
Annotate text using existing resources	Use the selected text to guess the resource to link to, but allow manual changes.
Create links between resources	Allow different ways of creating relationships and provide inverse properties.
"Don't repeat yourself."	Support the user with annotation services, but under control of him. Treat tables specially.
Compatibility and ease of integration	Implementation is based on XHTML+RDFa (client) and RDF (server)
Independence of domain ontologies	An RDF schema defines a set of annotations which can refer to well-known ontologies.

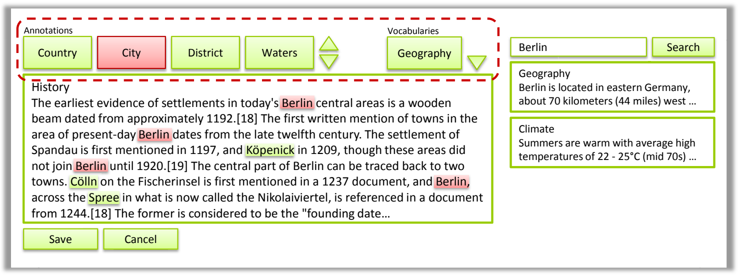
Table 1. Realizing the One Click Annotator: Challenges and solutions

3.2 User Interface

Providing a tool for creating semantic content by non-experts that fulfills the above requirements, we develop the *One Click Annotator* (OCA). In the following, we describe the functionalities and the user interface of the OCA and discuss challenges we were confronted and propose solutions for them.

Clear and intuitive UI. The first challenge addresses the following question: What user interface is suitable to create semantic annotations but is clear and intuitive enough to be understood by non-experts? The central idea is that every computer user is familiar with word processors and is able to select some text and to click on a button to format it italic. Thus, we transferred the concept of assigning style sheets to texts as known from modern word processors to the OCA. In a word processor a user can choose between different sets of style sheets. After having chosen a set he can select some text and assign a style sheet to it, e.g., *heading 1*. In the OCA, a user can similarly choose between different vocabularies, e.g., personal information and geography, and assign an annotation to some selected text, e.g., *foaf:firstName*. The dashed box in Figure 1 shows the ribbon toolbar for selecting an annotation. Please note, that the toolbar only displays human understandable labels but not namespace prefixed labels of properties – in general, we try to avoid to show URIs.

Represent annotations as RDF. The procedure of adding annotations to text leads to another challenge. In our opinion, a non-expert user is hardly aware of the difference between a concept and the label of a concept. For example, if a user reads the phrase *George Bush* in some text, he probably wants to state that



History
The earliest evidence of settlements in today's Berlin central areas is a wooden beam dated from approximately 1192.[18] The first written mention of towns in the area of present-day Berlin dates from the late twelfth century. The settlement of Spandau is first mentioned in 1197, and Köpenick in 1209, though these areas did not join Berlin until 1920.[19] The central part of Berlin can be traced back to two towns. Cölln on the Fischerinsel is first mentioned in a 1237 document, and Berlin, across the Spree in what is now called the Nikolaiviertel, is referenced in a document from 1244.[18] The former is considered to be the "founding date...

Geography
Berlin is located in eastern Germany, about 70 kilometers (44 miles) west ...

Climate
Summers are warm with average high temperatures of 22 - 25°C (mid 70s) ...

Fig. 1. The ribbon toolbar of the One Click Annotator for creating annotations

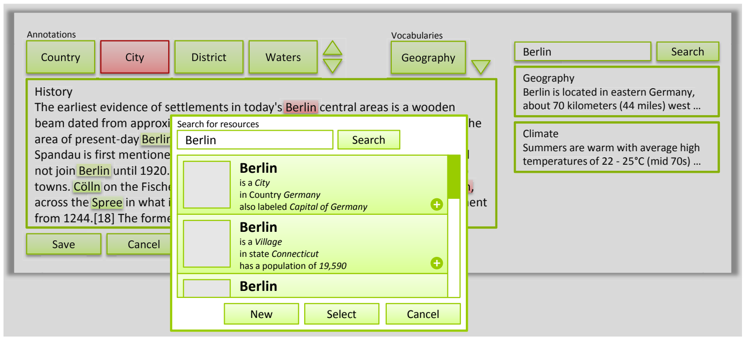
George Bush is a person but not that it is the name of a person. We consider this behavior in the OCA by using the labels of properties in the toolbar (in the following also referred to as annotation) and creating a resource that has a property with the selected text as literal value.

In many cases it will also be possible to derive the type of the generated resource, because a property has a specific domain defined in the schema, e.g., *foaf:Person* is the domain of *foaf:surname*. Moreover, the domain may be derived from the domain of the annotation set. For example, in the context of personal information the property *foaf:name* will probably have the domain *foaf:Person* which is more specific than *owl:Thing* as defined in FOAF.

Annotate text using existing resources. If users annotate the same resource in different texts, it is important to reference the same resource in the generated RDF statements. Otherwise, we get many resources that are not interlinked and the statements in the repository are not very useful and meaningful.

After a user has selected some text and clicked on an annotation in the toolbar, the system presents a list with information of existing resources that are likely to be chosen by him (see Figure 2). The resources may originate from a local or a external repository (e.g., DBpedia). To retrieve appropriate resources from the repositories we use three sources of information: 1) the selected text, 2) the schema information stored in the vocabulary definitions (e.g., domain and range of properties), and 3) annotation services. In our current implementation we use some simple syntactical measures to determine the similarity, e.g., text containment or Levenshtein distance. To enable a user to choose an appropriate resource, the list of resources has not only to contain the label of a resource but also some properties of it. Additionally, a user can view the text fragments that contain occurrences of the resource. If he does not find the resource in the list then he can manually search for a resource or create a new one.

Create links between resources. Assigning an annotation to a selected text is only one facet of creating semantic annotations. The other facet is the creation of relationships between two resources. To simplify the creation of links between resources we offer different approaches: 1) Use a context menu associated with an annotated text to choose a property and a resource to link to or 2) drag an annotated resource, drop it onto another one, and select a property.



Geography
Berlin is located in eastern Germany, about 70 kilometers (44 miles) west ...

Climate
Summers are warm with average high temperatures of 22 - 25°C (mid 70s) ...

History
The earliest evidence of settlements in today's Berlin central areas is a wooden beam dated from approxi
area of present-day Berlin
Spandau is first mentione
not join Berlin until 1920.
towns. Cölln on the Fische
across the Spree in what i
from 1244.[18] The forme

Search for resources
Berlin

Berlin
is a *City*
in Country *Germany*
also labeled *Capital of Germany*

Berlin
is a *Village*
in state *Connecticut*
has a population of *19,590*

Berlin

Fig. 2. User interface for selecting an existing resource

"Don't repeat yourself." Annotating text is a time-consuming task that, in most cases, users do not like to perform even if they only have to add a few metadata about the text. According to the principle "don't repeat yourself" a user can invoke annotation services which automatically analyze the text and propose annotations. Besides external annotation services such as OpenCalais we develop a client-side component that creates annotations while the user is typing. This component keeps track of the annotations created by the user and adds the same annotations to other parts of the text. Independent of the used annotation service, an icon is placed behind any automatically generated annotation allowing to accept, to reject, to choose an alternative annotation, or to disable automatic annotation at all (cf. Figure 3).

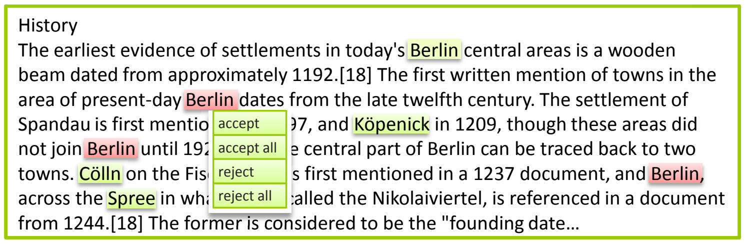

History
The earliest evidence of settlements in today's Berlin central areas is a wooden beam dated from approximately 1192.[18] The first written mention of towns in the area of present-day Berlin dates from the late twelfth century. The settlement of Spandau is first mentio 97, and Köpenick in 1209, though these areas did not join Berlin until 192 e central part of Berlin can be traced back to two towns. Cölln on the Fis s first mentioned in a 1237 document, and Berlin, across the Spree in wh alled the Nikolaiviertel, is referenced in a document from 1244.[18] The former is considered to be the "founding date...

Fig. 3. User interface for accepting or rejecting an annotation

We also develop a dedicated approach for annotating the content of tables. A table should be treated differently, because a table implicitly expresses relationships between the entries of a row (or column). Especially, annotations are likely to repeat in each row, e.g. all entries in a column share the same annotation and the same entry may occur multiple times in a column. To reduce time and effort for annotating tables, we combine the approaches of annotating text, linking resources, and automatic generation of annotations. The user selects a column and assigns an annotation to all entries. Afterwards, he defines the relationships between the columns using the same approach as to link resources. The

OCA transfers the relationship to the entries of each row. Finally, the automatic annotation service takes care of linking repeating entries to the same resource.

3.3 Technologies

The OCA is the client part of a client server architecture for semantic content authoring. It is responsible for displaying the user's content as well as the annotation toolbar and for modifying the content to include semantics. The server is only contacted if the OCA needs information about resources or has to invoke external annotation services. In this section we describe details about processing of the user's content and the creation of semantic annotations by the OCA.

Compatibility and ease of integration. The OCA is implemented as a plug-in of tinyMCE [**?**] which is a WYSIWYG editor for Web browsers. The tinyMCE and, thus, the OCA plug-in is only able to handle HTML content. As a consequence of this and due to our requirements of a high compatibility with and an easy integration into existing software components, the OCA consumes and produces XHTML+RDFa [**?**]. So, XHTML+RDFa is the central format for exchanging content between the client and the server. Servers can then, for example, extract the RDF from the XHTML+RDFa content (e.g., RDFa Distiller [**?**] or ARC [**?**]), link it to RDF data of other sources, and serve it as linked data. As a side effect of using XHTML+RDFa, we are able to provide additional functionality such as faceted viewing which highlights user-selected annotations in a Web browser (e.g., all names of persons).

If the OCA needs to retrieve data about resources it uses an http-request to submit a SPARQL query to the server. In our implementation, the server returns the answer as a simple array of resources and their properties in JSON format. Other formats such as RDF or XML are conceivable.

Convert annotations to RDF(a). While we described the creation of the semantic annotation on a high level in the previous section, we now give some more details on how the content is modified. The procedure of annotating some text is as follows (cf. Figure 4): After a user has selected some text and clicked on the button of an annotation, the OCA queries the server for resources that the user may refer to. He can either select a resource from the list or, if the list does not contain the intended resource, manually search for it or create a new one. New resources are assigned a generated URI based on a hash value with the annotated text as the label. A label is important to ensure that there is at least one human readable property for all resources to display. Then, the OCA inserts a `span` tag containing appropriate RDFa attributes into the XHTML content. For example, Figure 5 shows a snippet of a XHTML+RDFa document stating that there is a resource with the name *George Bush*. When the user saves the content then it is sent to the server for further processing. Finally, if the user saves the content in the editor then the XHTML+RDFa is send to the server which processes it, i.e., extract the RDF, store it in an RDF repository and make it available as linked data.

A drawback of using XHTML+RDFa as representation of annotations is that we are currently not able to create overlapping annotations in a text, e.g., x y

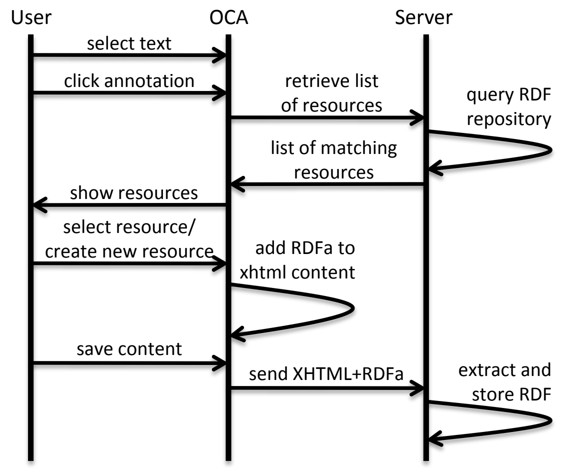


Fig. 4. Sequence diagram illustrating the process of creating an annotation

```
(a)   <span about="http://www.loomp.org/resource/fdff8477..."
            property="http://xmlns.com/foaf/0.1/name">George Bush</span>
(b)   <http://www.loomp.org/resource/fdff8477...>
          rdf:type foaf:Person;
          foaf:name "George Bush".
```

Fig. 5. Example of an RDFa annotation (a) and resulting RDF statements (b)

z where x y and y z are annotated differently. The reason is that each XHTML document is also an XML document. Thus, we have to maintain well-formedness of the document to process it using XML-based tools and libraries. However, we think that this disadvantage is not severe because the case will rarely occur.

Independency of domain ontologies. The OCA is extensible with respect to the ontologies that can be used with it for annotation. A domain is usually described by several ontologies. Since we also want to hide the complexity of choosing an ontology and of mapping different ontologies, we decided to develop a simple RDF schema for describing the elements of the OCA toolbar. Using this schema an annotation is described by the referred property (`loomp:refersTo`) of a standard ontology (e.g., Dublin Core, FOAF, etc.), a label for displaying in the toolbar, a description of the annotation, and examples of its usage (Figure 6).

```
<loomp:AnnotationSet rdf:about=".../pi/0.1/PersonalInformation">
 <rdf:li>
    <loomp:Annotation rdf:about=".../pi/0.1/name">
    <loomp:annotationDomain rdf:resource=".../foaf/0.1/Person"/>
    <loomp:refersTo rdf:resource="http://xmlns.com/foaf/0.1/name"/>
    <rdfs:comment xml:lang="en">Name of a person.</rdfs:comment>
    <rdfs:label xml:lang="en">name</rdfs:label> ...
    </loomp:Annotation>
 </rdf:li> ...
</loomp:AnnotationSet>
```

Fig. 6. Excerpt of an annotation set: Name of a person

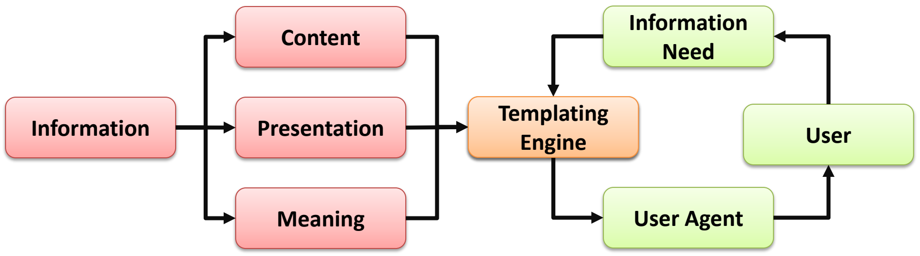


Fig. 7. Separating content from meaning from presentation

Furthermore, the engineer of an annotation set can give hints about the type of resources (`loomp:annoationDomain`) that are created when a user selects an annotation for some text. This is useful to assign a more specific type to a resource than it is defined in the original ontology. For example, the domain of the property `foaf:name` is `owl:Thing` at the moment [?]. In the context of an annotation toolbar about personal information it is more meaningful to create a resource which is of type `foaf:Person` or `foaf:Agent`. The annotation sets to be displayed in the OCA can be configured, so that the OCA shows only vocabularies needed for a given application domain, e.g., a sport journalist will only see annotation sets relevant for his tasks.

4 Semantic Content Management System

The One Click Annotator relies on the functionality of a server managing the semantic content – we refer to such a server as *semantic content management system* (SCMS). A SCMS is a content management system that follows the principle of separating content from meaning from presentation (cf. Figure 7). Thus, a templating engine can make use of the different aspects of an information object to render it according to the specific information needs of a content consumer (e.g., highlight resources of a given type in the content and serve it in different formats). In this section, we give an overview of the programming interface that an SCMS has to provide for supporting OCAs. As a proof of concept and backend of the OCA we currently implement this interface in *loomp*[1].

Figure 8 illustrates the interaction between OCA, SCMS, and RDF store. On the left sides of the arrows we noted the used protocols and on the right side the formats for exchanging data. The server and the OCA exchange content in XHTML+RDFa format and data about resources in JSON format. The communication between SCMS and RDF is based on Semantic Web standards. Using the API of the SCMS an One Click Annotator has access to the semantic content, information

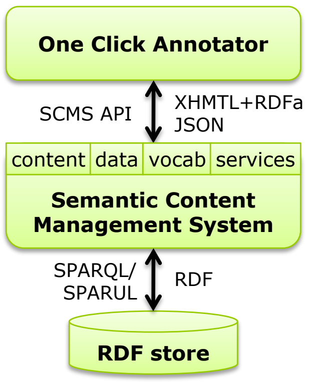


Fig. 8. Architecture

[1] `http://loomp.org`

about resources, vocabularies for annotating content, and additional services. We omitted interfaces for authentication and server administration, because they do not provide any unusual functionality.

The content API comprises functionality that is related to manage content. Its foundation is a simple and easy-to-understand domain model: we distinguish between *content elements* and *documents*. A content element can consist of text, multimedia objects, or embedded SPARQL queries[2]. A document consists of a sequence of content elements. We believe that this domain model is advantageous, because it conforms user's mindset, already used in modern content management systems, and allows an easy reuse of content. The API includes besides basic operations (e.g., to create, to update, and to delete content elements and documents) also operations to search for content semantically. The data API allows for accessing the RDF data directly, e.g., to retrieve all statements of a given resource. In an OCA, the functionality of this API is used to display all known information about a resource or to create links between resources. The vocabulary API provides access to annotation sets. While the OCA uses this interface only for retrieving annotation sets and building the annotation toolbar, a client can use it for creating, updating, and deleting annotation sets.

The server may offer additional services which are useful for annotating and accessing content. For example, the server could provide a service for annotating content automatically by sending the conten to an annotation service, e.g., OpenCalais. The annotated content is then returned to the OCA (Section 3.2).

5 Related Work

The key concept of the One Click Annotator is to support non-expert users w.r.t. semantic technologies to create semantic content. The annotation happens "on-the-fly" while the user is writing text. To our best knowledge we are not aware of any tool that is focused on the target group non-experts as much as the OCA.

However, there are many approaches that support automatic and manual creation of semantic annotations on Web content, e.g., OpenCalais [**?**], Zemanta [**?**], GATE[**?**], and COHSE [**?**]. Automatic approaches based on semantic tagging or text mining are complementary to our approach of one click annotation. Provided as services by a semantic content management system the One Click Annotator can support users in annotating content by invoking these services. Thus, we do not discuss them here in detail.

In literature, we found also tools for annotating manually the content. There is a category of tools that are based on annotation frameworks (Annotea [**?**] and CREAM [**?**]) and enable users to add metadata to content. With the OCA we follow a different approach, because the OCA enables the user the assign a meaning to separate words instead of adding metadata. Some other tools do not seem to be under development for years although they are mentioned in recent articles, e.g., SMORE [**?**] and Mangrove [**?**].

[2] loomp supports only text and embedded SPARQL queries at the moment.

In the next category we put approaches based on social tagging and collaboration. There are many wiki-based systems like semantic wikis: OntoWiki [?], Ikewiki [?], or Semantic MediaWiki [?]. These wikis extend traditional wikis by functionalities that enable users to add annotations to a wiki page and to specify relationships between pages based on ontologies. In our opinion semantic wikis are far from being usable by non-experts. Besides the effort to learn a special syntax to write and to annotate content, a user has to cope with technical terms such as resource, different kinds of relationships, and namespaces.

In contrast, semantic tagging engines such as faviki [?] and the MOAT project [?] exploit the well-known user interaction procedure of tagging to annotate content. Faviki is based on Zemanta [?] and retrieves suggestions for tags, e.g., Wikipedia terms. Similarly, clients based on MOAT use existing ontologies, e.g., DBpedia, to add tags to content. Since they assign a tag to a Web resource these tools work similar as the mentioned annotation frameworks.

In [?], the authors present a JavaScript API for modifying RDFa directly on the client side and synchronizing the changes with the server. While our One Click Annotator is suitable for extensive changes of annotations of a text, this JavaScript library is a useful supplement for smaller changes of annotated texts.

The Tabulator linked data browser [?] allows users to edit data directly on the Web of data. However, since it requires a Firefox plug-in in its current stage of development we see it as a proprietary tool. OpenLink Data Spaces [?] provide a complete platform for creating a presence on the Web of data, e.g., calendar, weblog, or bookmark manager. However, they focus on describing the data entities semantically while we enrich the content itself.

6 Conclusion and Outlook

Up to now, there exist lots of tools wrapping or transforming the data of relational databases or similar content to RDF. However, we think that a key prerequisite in the vision of a Semantic Web is to enable ordinary Web users – non-experts w.r.t. semantic technologies – to create and consume semantic content on the Web easily. Currently, the barrier for them is still too high, because there are no easy-to-use Semantic Web authoring tools available. In this paper, we introduced a simple-to-use "One Click Annotator" (OCA) for enriching text content semantically. The key design goals are a clear and intuitive user interface hiding the complexity of creating semantic data and simple access to vocabularies. To our best knowledge we are not aware of any tool that is focused on the target group non-experts as much as the OCA.

We currently develop a browser-based OCA as a plug-in of the WYSIWYG editor tinyMCE which uses loomp as its backend. Since it is still in the status of a prototype, some concepts are partially implemented and not yet available at the public demonstrator (`http://loomp.org/`).

In the future we focus on automatic generation of links to external resources (e.g., to the Linking Open Data dataset cloud), improved integration of external annotation services such as OpenCalais, and privacy issues.