An Annotation Framework for the Semantic Web

Steffen Staab, Alexander Maedche and Siegfried Handschuh
Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
{sst, ama, sha}@aifb.uni-karlsruhe.de
http://www.aifb.uni-karlsruhe.de/WBS

Abstract

Creating metadata by annotating documents is one of the major techniques for putting machine understandable data on the Web. Though there exist many tools for annotating web pages, few of them fully support the creation of semantically interlinked metadata, such as necessary for a truely *Semantic Web*. In this paper, we present an ontology-based annotation environment, *OntoAnnotate*, which offers comprehensive support for the creation of semantically interlinked metadata by human annotators.

1 Introduction

With the upswing of metadata on the Semantic Web for means like semantic web portals (Staab et al., 2000a), there comes the urgent need for adding semantic metadata to existing web pages such that they are digestible for humans and machines. Though there exists a wide range of sophisticated, even professional, annotation tools (cf. Section 4 on related work), none of the ones that we know of has yet fully exploited the new wealth of possibilities that come with RDF (Lassila & Swick, 1999) and RDF-Schema (Brickley & Guha, 1999) as metadata formats. In particular, semantic annotation has so far mostly restricted to describing documents or items in documents *in isolation of each other*. In light of the Semantic Web, what intelligent agents crave for are web pages and items on web pages that are not only described in isolation from each other, but that are also *semantically interlinked*.

We have used semantically interlinked information for gathering knowledge relevant in a particular community of users (Staab et al., 2000a). The underlying idea was that for that domain a group of users would provide semantic metadata *about the content* of relevant web pages. Thus, our Community Web Portal could present all this knowledge, taking great advantage of semantic structures: personalization by semantic bookmarks ("Fred is interested in RDF research"), conceptual browsing, or the derivation of implicit knowledge (e.g., if John works in a project, which is about XML then he knows something about XML), have been some of the features that thrived by having semantically interlinked information. Similarly, we envision that intelligent agents may profit from semantically interlinked information on the Web in the future.[1]

Building the Community Web Portal we found that there were a number of tricky issues with providing semantic annotation in this manner: First of all, the semantic annotation task does not adhere to a strict template structure, such as Dublin Core to name one of the more sophisticated ones in use. Rather it needs to follow the structure given by schema definitions that may vary with, e.g., domain and purpose. In fact, our intelligent agents rely on *domain ontologies*. Semantic annotations need to be congruent with ontology definitions in order to allow for the advantages we have indicated above. Secondly, semantically interlinked metadata is labor-intensive to produce and, hence, expensive. Therefore duplicate annotation must be avoided. Because semantic annotation is a continuous process in a distributed setting there are several sources for duplication. There is knowledge generated by other annotators. In order to allow for the reuse of their annotations it is important that one does not start from scratch when

[1] Similar projects like WebKB (Martin & Eklund, 1999), SHOE (Heflin & Hendler, 2000), and, more recently, DAML (`http://www.daml.org`) point in the same direction.

annotating sources, but that one builds on others efforts (in particular their creation of IDs). Then, there is a multitude of schema descriptions (ontologies) that also change over time to reflect changes in the world. Because manual re-annotation of old web pages seems practically infeasible, one needs an annotation framework that allows to handle ontology creation, mappings and versioning. Thirdly, purely manual annotation is very expensive. Therefore, only very valuable information will be annotated and it is necessary to help the human annotator with his task. What is needed is support for automatic — or at the least, semi-automatic — semantic annotation of web pages. Finally, there is a lack of experience in creating semantically interlinked metadata for web pages. It is not clear how human annotators perform overall and, hence, it is unclear what can be assumed as a baseline for the machine agent. Though there are corresponding investigations for only *indexing* documents, e.g. in library science (Leonard, 1977), a corresponding richer assignment of interlinked metadata that takes advantage of the object structures of RDF is lacking.

In this paper, we deal with the first three of the above mentioned issues. Regarding the fourth problem on evaluating semantic annotation we refer the interested reader to a companion paper (Staab et al., 2000b).

We first (Section 2) present our basic tool for ontology-based semantic annotation and, then, consider the issue of semantic annotation as an ongoing process. In particular, interlinkage between objects and evolving metadata schema need to be managed to avoid redundant annotations and re-annotating, respectively. Section 3 describes the different layers for semi-automatic semantic annotation and introduces the techniques that we use in our tool. Before concluding, we discuss related work in the areas of evolving schemata and crawling, semi-automatic semantic annotation — prerequisite experiences and techniques for useful, semantically interlinked metadata on the Semantic Web.

2 Ontology-based Semantic Annotation

An *ontology* is commonly defined as an explicit, formal specification of a shared conceptualization of an domain of interest. This means that an ontology describes some application-relevant part of the world in a machine-understandable way. The concepts and concept definitions that are part of the ontology have been agreed upon by a community of people who have an interest in the corresponding ontology. The core “ingredients” of an ontology are its set of concepts, its set of properties, and the relationships between the elements of these two sets.

Ontological structures may give additional value to semantic annotations. They allow for additional possibilities on the resulting semantic annotations, such as inferencing or conceptual navigation that we have mentioned before. But also the reference to a commonly agreed set of concepts by itself constitutes an additional value through its normative function. Furthermore, an ontology directs the attention of the annotator to a predefined choice of semantic structures and, hence, gives some guidance about what and how items residing in the documents may be annotated.

Besides of these advantages that ontology-based semantic annotation yields in comparison to “free text metadata generation”, the extended set of capabilities also entails some new problems that need to be solved. In particular, semantic interlinkage between document items incurs the difficulty to adequately manage these interlinkages. Essentially, this means that an ontology-based annotation tool must address the issue of *object identity* and its management across many documents. Also, ontologies may have elaborate definitions of concepts. When their meaning changes, when old concepts need to be erased, or when new concepts come up, the *ontology changes*. Because updating previous annotations is generally too expensive, one must deal with change management of ontologies in relation to their corresponding annotations. Finally, one must prevent redundant annotation which stem from duplicate pages on the web or annotation work done by fellow annotators. Hence, we provide two basic mechanisms for recognizing *document identity*. In the remainder of this section we embed these requirements into a coherent framework.

2.1 OntoAnnotate — The Core Tool

While most annotation tools implicitly subscribe to a particular ontology (e.g., Dublin Core), our tool, OntoAnnotate, makes the relationship between particular ontologies and their parts, i.e. concepts and properties, explicit. OntoAnnotate, presents to the user an interface that dynamically adapts to the given ontology. It has been developed based on our earlier experiences with manual ontology-based semantic annotation that have been described in (Erdmann et al., 2000).

As principal language for semantic annotations and ontologies, OntoAnnotate relies on RDF and RDF Schema. RDF Schema can be seen as a language for lightweight ontology descriptions, allowing to define the interlinkage between different concepts (called "classes" in RDFS), properties, and objects (i.e "class members", also called "instances"). To name but a few other possible formats, WebKB uses Conceptual Graphs (Martin & Eklund, 1999), SHOE employs horn logic rules (Heflin & Hendler, 2000), and we have formerly exploited F-Logic (Staab et al., 2000a). RDF and RDF Schema, however, provide completely web compatible common denominator that everyone agrees on now. Therefore we have replaced proprietory formats we have used originally.

OntoAnnotate allows for the easy annotation of HTML documents. One may create objects with URIs and relate them to text passages, which are then highlighted. The semantic meaning of the objects and the text passages is given by four semantic categories:

1. **Object identification**: New objects are created by asserting the existence of an object with a unique identifier. The annotation tool supports the creation of object identifiers from text passages.

   This is a mostly syntactic operation, the only semantic consequence is that the set of existing objects is augmented by one.

2. **Object–class relationships**: Each object is assigned to a class of objects by the human annotator. In general, objects may be asserted to belong to multiple classes. To keep the user interface and the evaluation simpler, OntoAnnotate only allows single classification.

3. **Object–attribute relationships**: Each object may be related to attribute values by an attribute. Each attribute value is either a text passage chosen by highlighting or a string typed in by the annotator. For a given object the annotator can only create object–attribute relationships if the object's class definition allows its creation, i.e. if the class definition includes a corresponding attribute.

   An attribute is a property the domain of which is a literal.

4. **Object–object relationships**: Each object may be related to all existing objects (including itself) via an (object) relation. For a given object the annotator can only create object–object relationships if the object's class definition allows its creation, i.e. if the class definition includes a corresponding (object) relation.

   An relation is a property the domain of which is a resource.

Figure 1 shows the screen for navigating the ontology and creating annotations in OntoAnnotate. The left pane displays the document and the right panes show the ontological structures contained in the ontology, namely classes, attributes and relations. In addition, the right pane shows the current *semantic annotation knowledge base*, i.e. existing objects, their classification, object–attribute relationships and object–object relationships created during the semantic annotation.

To illustrate the annotation process with OntoAnnotate, we sketch a small annotation scenario using our tool: Annotation typically starts with identifying a new object. The user provides a new object identifier and selects the appropriate class of this object from a tree view. In our example, the object identifier `RStuder` is typed in and the class FULLPROFESSOR is selected from the ontology. Upon categorization of a new object into a class C, OntoAnnotate shows the possible attributes of C (cf. the attributes ADDRESS, NAME, PHONE, etc. of FULLPROFESSOR in the right upper pane of

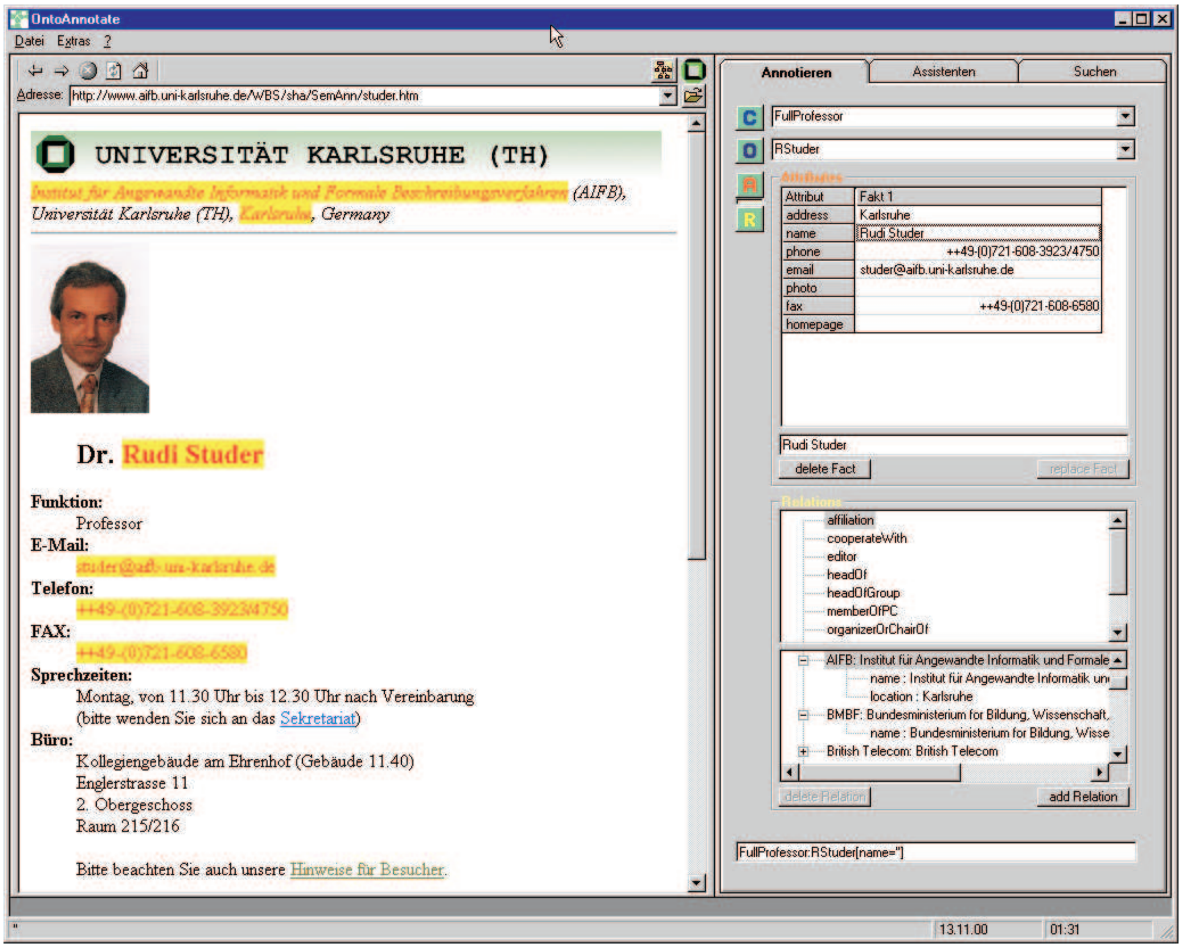

Figure 1: Screenshot of the OntoAnnotate GUI.

Figure 1) and the actual attributes of the chosen object (cf. `Karlsruhe`, `Rudi Studer`, etc. in right upper pane of Figure 1). In addition, one may look at the object relations of C (cf. `affiliation`, `cooperateWith`, etc. in the right lower pane of Figure 1) and the actual relations of the chosen object. In order to dynamically display the properties of classes and their instances, OntoAnnotate queries the *annotation inference server*. The annotator continues with marking text passages and drags them into empty fields of the attribute table, thereby creating new attributes relationships between the currently chosen object and the currently marked text passage (e.g., between `RStuder` and `studer@aifb.uni-karlsruhe.de` in Figure 1). The annotator may create metadata describing new object–object relationships by choosing an object relation and, then, either creating a new object on the fly or by choosing one of the objects, pre-selected by OntoAnnotate according to the range restriction of the chosen relation. For instance, the AFFILIATION of a PERSON must be an ORGANIZATION. Therefore, only organizations are offered as potential fillers for the affiliation relation of `RStuder`.

2.2 Object Identity

The first version of OntoAnnotate already relied on ontology structures to guide annotation, but it did not consider annotation as being a process carried through in a complex environment. The general problem stems from the fact that without corresponding tool support, annotators would too often create new objects rather than reuse existing ones. Therefore new properties were not attached to existing objects, but to new entities. In case studies, like the Community Web Portal (Staab et al., 2000a) annotators came up with many different object identifiers for single persons, which made it impossible to combine all the data about these persons.

Considering semantic annotation as a continuous process, we came up with two new requirements:

1. The annotation inferencing server needs to maintain object identifiers during the annotation process.

2. A crawler needs to gather relevant object identifiers for the start of the annotation.

The first requirement is solved by the annotation inference server, by adding objects to and querying objects from the server during actual annotation as described in the previous subsection.

The second requirement has been solved by allowing the annotator to start a focused crawl of RDF facts — covering the document and annotation server, but also relevant parts of the Web — which provides the annotation inference server with an initial set of object identifiers, categories, attributes and relations. Thus, the metadata provided by other annotaters may be used as the starting point that one may contribute additional data to.

Currently, RDF data is comparatively weakly interlinked. Hence, it is sufficient to restrict the focus of the crawl by web server restrictions and depth of the crawl. With more metadata on the Web, one needs to employ more sophisticated techniques in the future.

2.3 Ontology Changes

There exists a tight interlinkage between evolving ontologies and the semantic document annotation. In any realistic application scenario, incoming information that is to be annotated does not only require some more annotating, but also continuous adaptation to new semantic terminology and relationships.

Heflin and Hendler (Heflin & Hendler, 2000) have elaborated in great detail on how ontology revisioning may influence semantic annotations. Therefore, we here only sketch one example revision and its effects:

When an existing class definition is refined, the maintainer of the semantic annotations may explore the objects that belong to this class. He may decide individually or for all objects

- that the objects stay in the class and, hence, the semantic meaning of the annotations is extended by additional semantic constraints;

- that the objects are categorized to belong only to the superclasses of the re-defined class and, hence the semantic meaning of the annotations is reduced by cutting away semantic constraints;

- that the objects are moved to another class.

Along similar lines, other cases of ontology revisions are treated.

The annotation maintainer may explore all the possibilities in the ontology engineering tool, OntoEdit (Staab & Maedche, 2000) and may define mapping rules to bridge between different ontology revisions. Later on, querying may take advantage of these mappings to also retrieve "old" annotations.

2.4 Document Identity

In order to avoid duplicate annotation, existing semantic annotations of documents should be recognized. Because interesting semantic annotations will eventually refer to external web pages that change, the annotator needs some hints when he encounters a document that has been annotated before, but that may have slightly changed since. Finally, the annotator also needs to recognize that this may be a duplication of another document seen before (e.g. on a mirror site).

For these recognition tasks we provide the following mechanisms: In our local setting we have a document management system where annotated documents and their metadata are stored. OntoAnnotate uses the URI to detect the re-encounter of previously annotated documents and highlights annotations in the old document for the user. Then the user may decide to ignore or even delete the old annotations and create new metadata, he may augment existing data, or he may just be satisfied with what has been annotated before.

In order to recognize that a document has been annotated before, but now appears under a different URI, OntoAnnotate searches in the document management system computing similarity with existing documents by document vector models. If there appear documents the similarity which to the currently viewed document is near 1, then these are indicated to the annotator such that he may check for congruency.

These two techniques for recognizing document identity are very basic, but effective for maintaining document identity in OntoAnnotate, given a dynamic environment such as the Web.

2.5 OntoAnnotate — The Semantic Annotation Environment

The overall annotation environment as outlined in this section is depicted in Figure 2: The core OntoAnnotate is used for viewing web pages and actually providing annotations. It also stores annotated documents in the document management system and adds new metadata to the annotation inference server. The latter is also queried for providing conceptual restrictions given by the ontology. Thus, the annotator's view is restricted to conceptual structures that are congruent with the given ontology.

The annotation process is started either with an annotation inference server without objects, or the server process is fed with metadata crawled from the Web and the document server. The annotation inference server supports multiple ontologies. Annotations refer to the classes and properties that were used for their creation by namespaces. F-Logic rules are finally used to map between different namespaces, thus allowing to keep track of semantic annnotations (at least to some degree) even when the currently used ontology is replaced by an update.

The user additionally has the possibility to use semi-automatic means for recognizing class instances and properties between them. In the subsequent section we will further describe the text analysis component that supports semi-automatic semantic text annotation.

3 Semi-Automatic Annotation

Based on our experiences and the existing annotation tool for supporting ontology-based semantic annotation of texts, we now approach semi-automatic annotation of documents. In general, one may distinguish between different kinds of semi-automatic annotation mechanisms, that have already researched in existing work:

- **Wrapper Generation:** Especially in the case of annotating web pages that mainly consist of HTML tables, one may annotate the first row of the table and automatically enumerate over the residual rows of the table.
- **Pattern Matching:** Regularity of word expressions may be captured by regular expression based patterns. For example given the pattern {word}*{GmbH} yields for the german language to generic pattern for company names, and, thus, successfully recognize instances of the class COMPANY of the ontology. Patterns are stored with the concepts of the domain ontology.
- **Information Extraction:** The most complex mechanism for semi-automatic annotation is full fledged ontology-based information extraction based on a shallow text processing strategy.

Depending in the structure given in the documents one may apply one of the methods listed above. We here only shortly describe the mechanisms that we currently use in our tool OntoAnnotate. In real-world documents typically all three methods (and more) will have to be applied in combination. In our future work we will analyze the structures contained in the documents to derive a suitable processing strategy for the documents and document parts.

Wrapper Generation. Recently, several approaches have been proposed for wrapping semi-structured documents, such as HTML documents. Wrapper factories (*cf.* Sahuguet et al. (Sahuguet & Azavant, 2000)) and wrapper induction (*cf.* Kushmerick (Kushmerick, 2000)) have considerably facilitated the task of wrapper construction. In order to wrap directly into *OntoAnnotate* we have developed our own wrapper approach that directly alignes regularities in semi-structured documents with their corresponding ontological meaning.

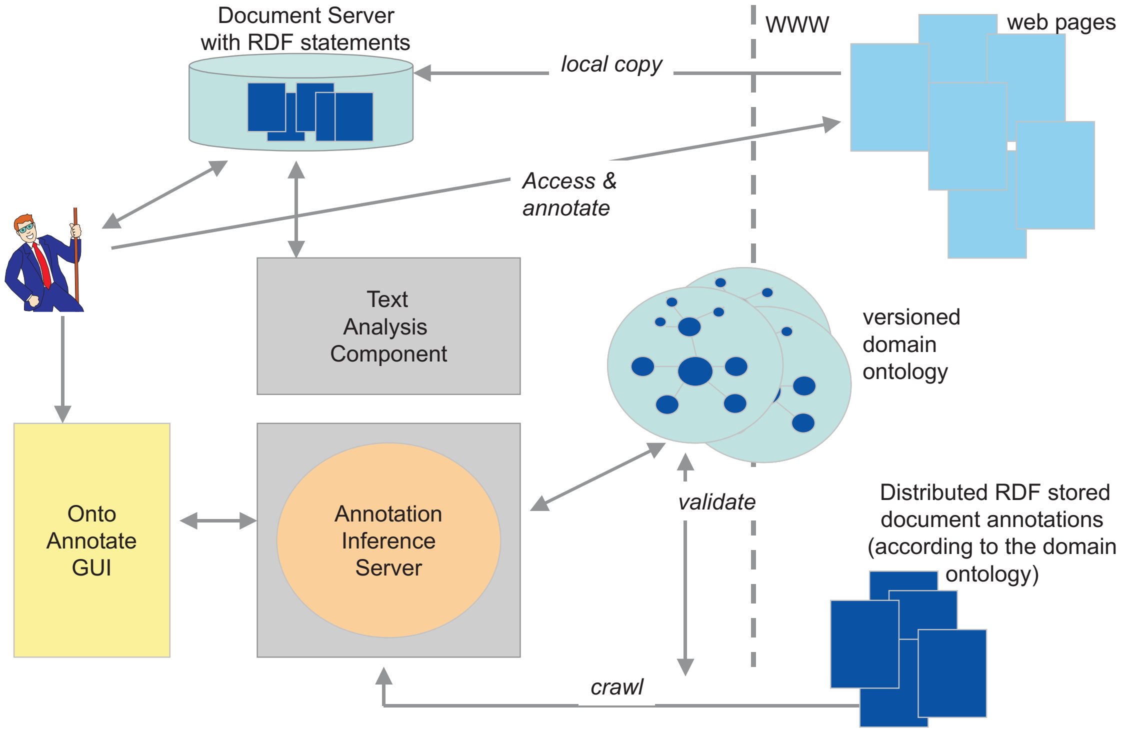

Figure 2: OntoAnnotate — The Semantic Annotation Environment.

Pattern Matching. We use a very simple mechanism for recognizing patterns in HTML documents. We use OroMatcher[2] based on Perl 5.003 regular expressions. Patterns are developed and tested in our regular expression workbench.

Information Extraction. At the highest level of processing we conceive an information extraction-based approach for semi-automatic annotation, which has been implemented on top of SMES (Saarbrücken Message Extraction System), a shallow text processor for German (cf. (Neumann et al., 1997)). This is a generic component that adheres to several principles that are crucial for our objectives. *(i)*, it is fast and robust, *(ii)*, it realizes a mapping from terms to ontological concepts, *(iii)* it yields dependency relations between terms, and, *(iv)*, it is easily adaptable to new domains.[3]

[2]OroMatcher 1.1 is freely available at http://www.savarese.org/oro/software/OROMatcher1.1.html

[3]The interlinkage between the information extraction system SMES and domain ontologies is described in further detail in (Staab et al., 1999).

We here give a short survey on SMES in order to provide the reader with a comprehensive picture of what underlies our system. The architecture of SMES comprises a *tokenizer* based on regular expressions, a *lexical analysis* component including a *word and a domain lexicon*, and a *chunk parser*. The tokenizer scans the text in order to identify boundaries of words and complex expressions like “$20.00” or “Mecklenburg-Vorpommern”[4], and to expand abbreviations. The lexicon contains more than 120,000 stem entries and more than 12,000 subcategorization frames describing information used for lexical analysis and chunk parsing. Furthermore, the domain-specific part of the lexicon associates word stems with concepts that are available in the concept taxonomy. *Lexical Analysis* uses the lexicon to perform, *(1)*, morphological analysis, *i.e.*, the identification of the canonical common stem of a set of related word forms and the analysis of compounds, *(2)*, recognition of name entities, *(3)*, retrieval of domain-specific information, and, *(4)*, part-of-speech tagging. While the

[4]Mecklenburg-Vorpommern is a region in the north east of Germany.

steps (1),(2) and (4) can be a viewed as standard for information extraction approaches (cf. (Appelt et al., 1993; Neumann et al., 1997)), the step (3) is of specific interest for our annotation task. This step associates single words or complex expressions with a concept from the ontology if a corresponding entry in the domain-specific part of the lexicon exists. E.g., the expression "Hotel Schwarzer Adler" is associated with the concept HOTEL.

SMES includes a *chunk parser* based on weighted finite state transducers to efficiently process phrasal and sentential patterns. The parser works on the phrasal level, before it analyzes the overall sentence. Grammatical functions (such as subject, direct-object) are determined for each dependency-based sentential structure on the basis of subcategorizations frames in the lexicon. Our primary output derived from SMES consists of *dependency relations* (Hudson, 1990) found through lexical analysis (compound processing) and through parsing at the phrase and sentential level. Thereby, the grammatical dependency relation need not even hold directly between two conceptually meaningful entities. For instance, in the sentence "The `Hotel Schwarzer Adler` in `Rostock` celebrates Christmas.", "Hotel Schwarzer Adler" and "Rostock", the concepts of which appear in the ontology as HOTEL and CITY, respectively, are not directly connected by a dependency relation. However, the preposition "in" acts as a mediator that incurs the conceptual pairing of HOTEL with CITY.

4 Related Work

This paper is motivated by the urgent need for adding metadata to existing web pages in an efficient and flexible manner that takes advantage of the rich possibilities offered by RDF (Lassila & Swick, 1999) and RDF-Schema (Brickley & Guha, 1999). Tools and practices so far have not reflected the new possibilities.

There are only a few tools that support adding metadata to existing web pages. We will present related work in this area and show how our approach and our implemented tool described in section 2 compares to the existing work. Additionally, our paper introduces semantic annotation as a continuous process. We therefore shortly review existing work in this area.

Related Work on Annotation Tools. Koivunen et. al. (Koivunen et al., 2000) introduce a framework for categorizing annotation tools distinguishing between a proxy–based and a browser–based approach. The proxy-based approach stores and merges the annotation and therefore preprocesses the annotated documents to be viewable for a standard web-browser. Within the browser–based approach the browser is modified to merge the document with the annotation data just prior to presenting the content to the user.

Many of the annotation tools rely on specialized browsers to offer a better user interface. One of them is *Amaya*. Amaya (Guetari et al., 1998; Vatton, 2000) is a web-browser that acts both as an editor and as a browser. It has been designed at W3C with the primary purpose of being a testbed for experimenting and demonstrating new languages, protocols and formats for the Web. It includes a WYSIWYG editor for HTML and XML. It can publish documents remotely, through the HTTP protocol. It handles Cascading Style Sheets (CSS) and the new MathML language, for representing mathematical expressions. An experiment for including vector graphics into Web documents is also described. Amaya is the primary browser /editor for the annotation approach in (Koivunen et al., 2000). The annotation data itself is exchanged in RDF/XML form to provide other clients access to the annotation database. Currently, however, it does not provide comprehensive support with annotation inference server and crawling.

ComMentor (Roescheisen et al., 1994) is another browser-based tool as part of the Stanford Integrated Digital Library Project. It manages the meta-information independently of the documents on separate meta-information servers. The research prototype implementation was completed in 1994, the code of the tool is no longer maintained.

ThirdVoice[5] is a commercial product that uses plug-ins to enhance web browsers. This enhancement allows the access to the annotation stored at the ThirdVoice database located on a centralized server from the company. The an-

[5]http://www.thirdvoice.com

notated text parts will appear in the browser as underlined links. These links point to the information on the database that will be presented on the user request in a separate viewer. Most of the annotation stored there seems to be links to further information, so that ThirdVoice is mainly used as a kind of an extended link-list. Along the same lines, *JotBot* (Vasudevan & Palmer, 1999) follows a browser–based approach that uses Java applets to modify the browsers behavior.

Yawas (Denoue & Vignollet, 2000) is an annotation tool that is based on the Document Object Model (DOM) and Dynamic HTML. It codes the annotations into an extended URL format and uses local files similar to bookmark files to store and retrieve the annotations. A modified browser can then transform the URL format into DOM events. Locally stored annotation files can be sent to other users.

The *CritLink* (Yee, 1998) annotation tool follows the proxy approach. This approach has the advantage that it works with any existing browser. The system is simply used by prefixing the URL with http://crit.org e.g. to see the annotated version of semanticweb.org someone can access the system with the URL http://crit.org/http://semanticweb.org.

The approach closest to *OntoAnnotate* is the *SHOE Knowledge Annotator*. The Knowledge Annotator is a Java program that allows users to mark-up web pages with the SHOE ontology. The SHOE system (Luke et al., 1997) defines additional tags that can be embedded in the body of HTML pages. In SHOE there is no direct relationship between the new tags and the original text of the page, i.e. SHOE tags are not annotations in a strict sense.

According to the above mentioned classification OntoAnnotate follows the browser-based approach with the exception that it is not developed as an web-browser extension. OntoAnnotate can be regarded as a workbench for semantic annotation of documents using domain-specific ontologies and this enriching HTML pages with semantics that an software agent is capable to automatically process the content of the page and reason about it.

Related Work on Semantic Annotation as a Continuous Process. There is only little research that considers the maintenance of ontologies or more general the maintenance of knowledge bases. In (Menzies, 1998) an overview over knowledge maintenance is given. Menzies reviews systems that contribute to different types of knowledge maintenance. The paper analyzes the AI and software engineering literature according to 35 different knowledge maintenance tasks. It concludes that there is no overall strategy that covers all 35 tasks.

The phenomenon of dynamic ontologies has nicely been described in (Heflin & Hendler, 2000). In their work they discuss the problems associated with managing ontologies in distributed environments such as the web. The underlying representation language is SHOE, a web-based representation language that supports multiple versions of ontologies. Foo (Foo, 1995) has published some initial, theoretical thoughts on ontology revision. Foo outlines the main ideas on the topic of ontology revision and constitutes ontology change as a frontier of knowledge systems research.

Related Work on Semi-Automatic Annotation. Pustejovsky et al. (Pustejovsky et al., 1997) describe their approach for semantic indexing and typed hyperlinking. As in our approach finite state technologies support lexical acquisition as well as semantic tagging. The goal of the overall process is the generation of so called *lexical webs* that can be utilized to enable automatic and semi-automatic construction of web-based texts.

In (Bod et al., 1997) approaches for learning syntactic structures from syntactically tagged corpus has been transferred to the semantic level. In order to tag a text corpus with typelogical formulae, they created tool environment called SEMTAGS for semi-automatically enriching trees with semantic annotations. SEMTAGS incrementally creates a first order markov model based on existing annotations and proposes a semantic annotation of new syntactic trees. The authors report promising results: After the first 100 sentences of the corpus had been annotated, SEMTAGS already produced the correct annotations for 80% of the nodes for the immediately subsequent sentences.

5 Conclusion

This paper presents an approach for creating meta data by (semi-automatic) annotating web pages. Starting from our ontology-based annotation environment OntoAnnotate, we have collected experiences in an actual evaluation study.

Future work will have to start on current studies that have looked at the feasibility of automatic building of knowledge bases from the web (cf. (Craven et al., pear)). In our future work, we want to integrate such methods into an even more comprehensive annotation environment — including e.g. the learning of ontologies from web documents (Maedche & Staab, 2000) and (semi-)automatic ontology-based semantic annotation. The general task of knowledge maintenance, including evolving ontologies and semantic annotation knowledge bases, remains a topic for much further research in the near future.

Acknowledgements. We thank Stefan Decker for initiating the first version of an annotation tool. We thank our students Mika Maier-Collin and Jochen Klotzbuecher for building the Annotation Tool; Dirk Wenke for the Ontology Engineering Environment OntoEdit; and our colleague Kalvis Apsitis for the RDF Crawler. We gratefully acknowledge the dedication of our students who annotated web pages for our experiments. This work was partially supported by DARPA in the project DAML/OntoAgents and by Ontoprise GmbH.

References

Appelt, D., Hobbs, J., Bear, J., Israel, D., & Tyson, M. (1993). FASTUS: A finite state processor for information extraction from real world text. In *Proceedings of IJCAI-93*, Chambery, France.

Bod, R., Bonnema, R., & Scha, R. (1997). Data-oriented semantic interpretation. In *In Proceedings of the Second International Workshop on Computational Semantics (IWCS), Tilburg, 1997.*

Brickley, D. & Guha, R. (1999). Resource description framework (RDF) schema specification. Technical report, W3C. W3C Proposed Recommendation. http://www.w3.org/TR/PR-rdf-schema/.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (to appear). Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*.

Denoue, L. & Vignollet, L. (2000). An annotation tool for web browsers and its applications to information retrieval. In *In Proceedings of RIAO2000*, Paris.

Erdmann, M., Maedche, A., Schnurr, H.-P., & Staab, S. (2000). From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In *P. Buitelaar & K. Hasida (eds). Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content*, Luxembourg.

Foo, N. (1995). Ontology Revision. In *Proceedings of the 3rd International Conference on Conceptual Structures. Springer Lecture Notes in Artificial Intelligence*. Springer.

Guetari, R., Quint, V., & Vatton, I. (1998). Amaya: an Authoring Tool for the Web. In *Maghrebian Conference on Software Engineering and Artificial Intelligence. Tunis, Tunsia*.

Heflin, J. & Hendler, J. (2000). Dynamic Ontologies on the Web. In *Proceedings of American Association for Artificial Intelligence Conference (AAAI-2000). Menlo Park, California, AAAI Press*.

Hudson, R. (1990). *English Word Grammar*. Basil Blackwell, Oxford.

Koivunen, M.-R., Brickley, D., Kahan, J., Hommeaux, E. P., & Swick, R. R. (2000). The W3C Collaborative Web Annotation Project ... or how to have fun while building an RDF infrastructure.

Kushmerick, N. (2000). Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence*, 118(1).

Lassila, O. & Swick, R. (1999). Resource description framework (RDF) model and syntax specification. Technical report, W3C. W3C Recommendation. http://www.w3.org/TR/REC-rdf-syntax.

Leonard, L. (1977). Inter-Indexer Consistence Studies, 1954-1975: A Review of the Literature and Summary of the Study Results. Graduate School of Library Science, University of Illinois. Occasional Papers No.131.

Luke, S., Spector, L., Rager, D., & Hendler, J. (1997). Ontology-based Web agents. In Johnson, W. L. (Ed.), *Proceedings of the 1st International Conference on Autonomous Agents*, pages 59–66. ACM.

Maedche, A. & Staab, S. (2000). Discovering conceptual relations from text. In *ECAI-2000 - European Conference on Artificial Intelligence. Proceedings of the 13th European Conference on Artificial Intelligence*. IOS Press, Amsterdam.

Martin, P. & Eklund, P. (1999). Embedding Knowledge in Web Documents. In *Proceedings of the 8th Int. World Wide Web Conf. (WWW'8), Toronto, May 1999*, pages 1403–1419. Elsevier Science B.V.

Menzies, T. (1998). Knowledge maintenance: The state of the art. *The Knowledge Engineering Review*, 10(2).

Neumann, G., Backofen, R., Baur, J., Becker, M., & Braun, C. (1997). An information extraction core system for real world german text processing. In *In Proceedings of ANLP-97*, pages 208–215, Washington, USA.

Pustejovsky, J., Boguraev, B., Verhagen, M., Buitelaar, P., & Johnston, M. (1997). Semantic indexing and typed hyperlinking. In *Proceedings of AAAI Spring Symposium, NLP for WWW*.

Roescheisen, M., Mogensen, C., & Winograd, T. (1994). Shared Web Annotations as a Platform for Third-Party Value-Added Information Providers: Architecture, Protocols, and Usage Examples. Technical report stan-cs-tr-97-1582, Computer Science Dept., Stanford University.

Sahuguet, A. & Azavant, F. (2000). Building Intelligent Web Applications Using Lightweight Wrappers. *to appear in: Data and Knowledge Engineering*.

Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Studer, R., & Sure, Y. (2000a). Semantic Community Web Portals. In *Proceedings of the 9th World Wide Web Conference (WWW-9), Amsterdam, Netherlands*.

Staab, S., Braun, C., Düsterhöft, A., Heuer, A., Klettke, M., Melzig, S., Neumann, G., Prager, B., Pretzel, J., Schnurr, H.-P., Studer, R., Uszkoreit, H., & Wrenger, B. (1999). GETESS — searching the web exploiting german texts. In *Proceedings of the 3rd Workshop on Cooperative Information Agents*, LNCS, Berlin. Springer.

Staab, S. & Maedche, A. (2000). Ontology engineering beyond the modeling of concepts and relations. In Benjamins, V., Gomez-Perez, A., & Guarino, N. (Eds.), *Proceedings of the ECAI-2000 Workshop on Ontologies and Problem-Solving Methods. Berlin, August 21-22, 2000*.

Staab, S., Maedche, A., & Handschuh, S. (2000b). Creating Metadata for the Semantic Web - A Annotation Environment and its Evaluation. Technical Report 412, Institute AIFB, Karlsruhe University.

Vasudevan, V. & Palmer, M. (1999). On Web Annotations: Promises and Pittfalls of Current Web Infrastucture. In *Proceedings of HICSS'99*, pages 5–8, Maui, Hawaii.

Vatton, I. (2000). W3C's Amaya 4.0 Editor/Browser. Technical report, W3C. http://w3c1.inria.fr/Amaya/.

Yee, K.-P. (1998). CritLink: Better Hyperlinks for the WWW. http://crit.org/ ping/ht98.html.