A Semantic Memory for Incremental Ontology Population

Berenike Loos* and Lasse Schwarten**

*German National Library, Information Technology, Frankfurt am Main, Germany
**University of Bremen, Computer Science, Bremen, Germany
b.loos@d-nb.de, lasse@tzi.de

Abstract

Generally, ontology learning and population is applied as a semi-automatic approach to knowledge acquisition in natural language understanding systems. That means, after the ontology is created or populated, an expert of the domain can still change or refine the newly acquired knowledge. In an incremental ontology learning and population framework (as e.g. applied in open-domain dialog systems) this approach is not sufficient as knowledge about the real world is dynamic and, therefore, has to be acquired and updated constantly. In this paper we propose the storing of newly acquired instances of an ontological concept in a separate database instead of integrating them directly into the system's knowledge base. The advantage is that incorrect knowledge is not part of the internal knowledge representation but stored aside. Furthermore, information about the confidence of the learned instances can be displayed and used for a final revision by an expert as well as for a further fully automatic acquisition.

1. Introduction

In an open-domain natural language understanding system the automatic learning of ontological concepts and corresponding relations between them is essential, as a complete manual modeling of them is not feasible as the real world is not static, but is continously extended by new objects, models, processes and their corresponding denotations.

Therefore, this information has to be extracted and converted into knowledge during a user's inquiry containing data not known to the system so far. As the reliability of the newly acquired instance cannot be ensured by an expert of the domain as e.g. in off-line ontology learning and population (from now on referred to as OLP; (Cimiano, 2006; Maedche, 2002)), it cannot be integrated completely into the system's knowledge base. Instead, this instance is stored in a database linked with the ontology, including confidence values for the existent data.

The advantage of the external storage is manifold. First of all, newly acquired knowledge can be incorrect or only partly correct. Therefore, it cannot be accessed by other components of the system, which are not in need of this particular piece of information for processing. Another advantage is the provision of a confidence value which indicates parts which are in need of a manual change by an expert. As it is likely that newly retrieved information is also requested by other users, the semantic memory additionally serves as a semantic cache, which provides a more performant access than the whole knowledge base.

In Section 2. the motivation of the external storage of knowledge is presented. Section 3. describes a scenario in which the semantic memory is of particularly interest. Thereafter, in Section 4. the model of the semantic memory is introduced. Section 5. gives an overview of related work in the field of ontology learning with a focus on the inclusion of a semantic memory and cache. Finally, Section 6. gives an outlook on additional work on this topic we are planning in the future.

2. Motivation

The ontology learning and population process can be divided into three main tasks as described in Loos (2006) and shown in Figure 1. The first task is the extraction of relevant information from texts. *Relevant information* can either mean all terms constituting a domain or information about a single word, which should be represented in the ontology. E.g. the ontology needs to be extended by all concepts which can be served at a restaurant. With the help of machine learning or e.g. pattern-matching methods this information can be extracted. The second task is the mapping of the newly found information to corresponding ontological concepts and instances. For this task a word-to-concept lexicon as well as a synonym lexicon and word sense disambiguation techniques can be advantageous to find the correct location in the ontology. The third task is the subsequent integration of the transferred information into the ontology. In Figure 1 this would be the insertion of *Restaurant* as a concept underneath *Building*. In this paper the focus lays on the final subtask and here especially on the population of the ontology. The term *ontology population* refers to the adding of new instances to ontological concepts.

As OLP is generally done in a semi-automatic manner, a final revision by an expert is undertaken. In incremental OLP this is not feasible, as the knowledge, which is acquired through information extraction techniques cannot be validated during run-time. Therefore, a kind of memory separated from the ontology has to be created so that the integrity of the ontology can still be warranted.

On the one hand, this is done to separate the not completely reliable knowledge from the validated system's ontology and, on the other hand, it is done to keep track of changes and a confidence value given by the system about the encoded information.

In Section 3. a scenario is presented to better illustrate the motivation for this work.

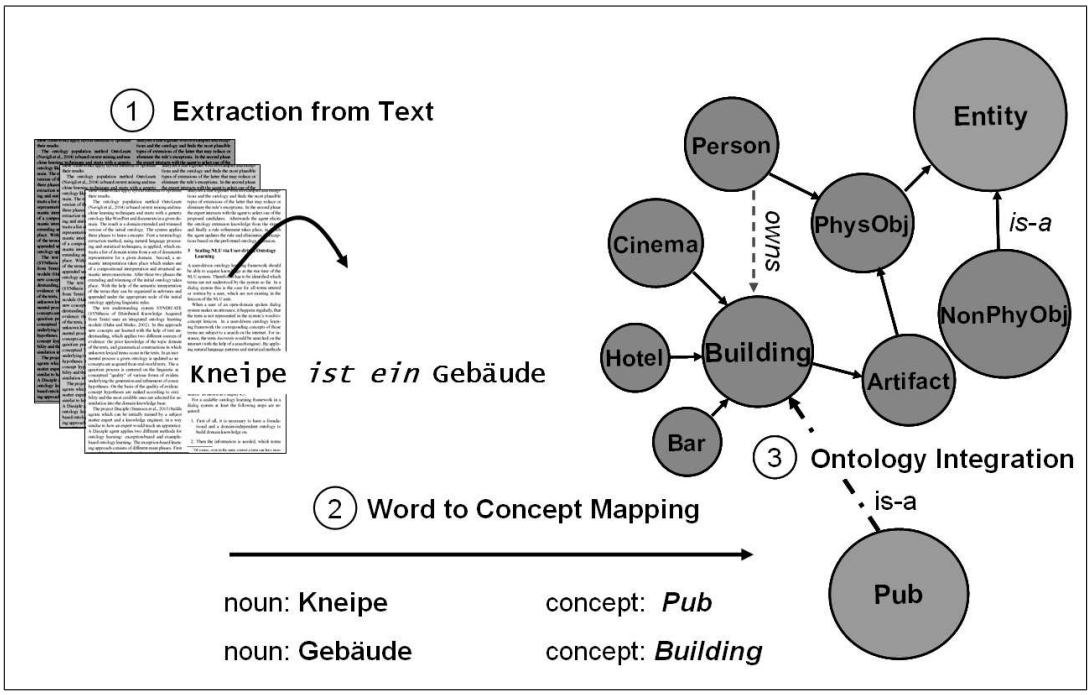

Figure 1: Three subtasks of ontology learning according to (Loos, 2006). In a first step information is extracted from text. In a next step, the information is mapped to ontological concepts. The last step constitutes the integration of the newly acquired knowledge in the ontology.

3. The Scenario

The user of e.g. a dialog system (such as Smartweb[1]) or an information portal (such as Heidelberg Mobil[2]) tries to get information about a place which is not included in the knowledge base of the system so far: E.g. he asks "How do I get to the i-Punkt?". The unknown term is the instance "i-Punkt", therefore, the appropriate concept in the ontology might be *Bar*[3]. This concept inherits a *has-street-name* property (as well as many others) from a superclass *building*. With the help of this hint and a contextual specification with respect to the location of the user, the corresponding street name of the place can be retrieved on the Internet with information extraction methods as described in Loos & Biemann (2007).

Now the newly acquired knowledge is given a value to express its confidence about the extracted information and the mapping of the found hypernym[4] into an ontological class, to which the instance can be added by an *instance-of* relation. Furthermore, the automatically acquired knowledge can be reviewed by an expert in an orderly manner as he sees the confidence values of the system and can change the database entries without having complete knowledge about the ontology (see Figure 2).

[1]http://www.smartweb-project.de (last access: 3rd March 2008).

[2]http://www.heidelberg-mobil.de (last access: 3rd March 2008).

[3]In the context of a specific location.

[4]According to Lyons (1977) hyponymy is the relation which holds between a more specific lexeme (i.e. a hyponym) and a more general one (i.e. a hypernym). E.g. animal is a hypernym of cat.

The following section will present the procedure involving the semantic memory.

4. The Semantic Memory

Whenever the system encounters an unknown term during a user's inquiry it tries to request the semantic memory by giving the term and the context location (as e.g. *Heidelberg* or *Bremen*) in which it appears. If it is successful the system can utilize the retrieved information and continue. In this case the semantic memory acts as a cache which increases the overall performance of the system.

However, in case the query fails, the system starts its information extraction component to come up with a meaning for the unknown term. After a hypernym of a named entity such as *i-Punkt* has been extracted from the Internet and mapped to a concept of the ontology, the system can store the result as a new instance in the semantic memory.

Using the knowledge that has been gained during the mapping process, the system can further identify semantic properties, which are then filled with additional data (e.g. *has-phone-number* etc.). For each semantic property the system tries to extract pieces of information. Apparently, the reliability of the extracted results varies, which is reflected in the semantic memory by an estimated confidence value.

The semantic memory, as seen in Figure 3, has been structured in a way that for each instance an arbitrary number of properties can be held. This allows the system to introduce new properties and relations at run-time.

Instances as well as semantic properties are assigned a confidence value to mark how confident the system in this particular result is. Furthermore, for each record in the *instance_learning_table* a unique identifier is stored. This

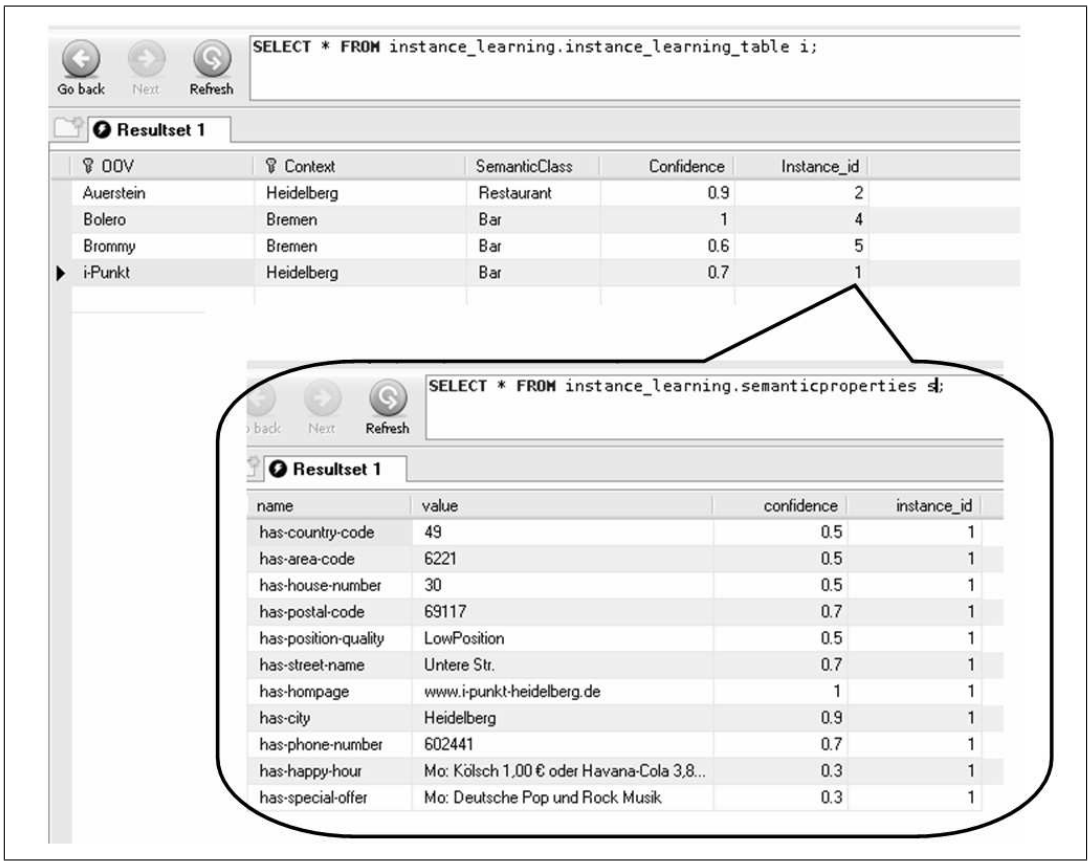

Figure 2: The semantic memory. The automatically acquired knowledge can be reviewed by an expert in an orderly manner as he sees the confidence values of the system and can change the database entries without having complete knowledge about the ontology.

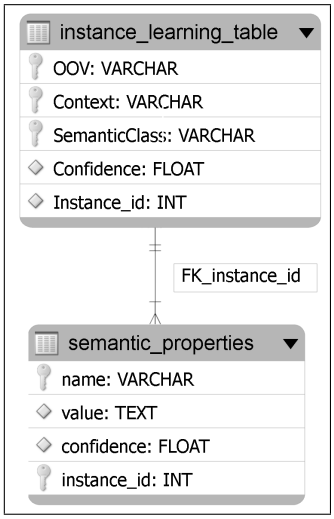

Figure 3: The database structure. For each instance an arbitrary number of properties can be held. Therefore, the system can introduce new properties and relations at runtime.

identifier is needed to link properties to each instance and can be used in future versions to allow duplicate entries with different confidence values and property sets.

In Figure 2 an example of the semantic memory entries is given. If the system queries the memory using the term *i-Punkt* and the context *Heidelberg*, it would retrieve the information that this term in this context has been mapped to *Bar* with a confidence of *0.7*. Along with this, the system gets a list with all the attached properties, their data and confidence values.

5. Related Work

The idea of acquiring knowledge exactly at the time it is needed is relatively new and became extremely useful with the emergence of open-domain dialog systems. Before that, more or less complete ontologies could be modeled for the few domains covered by a natural language understanding system. Nonetheless, many ontology learning frameworks exist, which alleviate the work of an ontology engineer to construct an ontology manually. Most of these components also include a memory for data which needs to be reviewed by an expert after the automatic learning process.

A framework which dates back before the rise of the term *ontology learning* is ASIUM (Faure and Nedellec, 1999), which helps a user in acquiring knowledge from technical text using subcategorization frames from a syntactic analysis as well as concept clustering for the extraction. The process of ontology integration itself is assessed and refined by an expert. Similarly, OntoLearn (Navigli et al., 2004; Missikoff et al., 2002) uses specialized Web site texts as a corpus to extract terminology. The resulting terms are filtered

by statistical techniques and then used to automatically enrich WordNet with the help of a semantic interpretation and the detection of taxonomic and similarity relations. The main application of the system is word sense disambiguation, which applies the learned ontology as a source of contextual knowledge.

In contrast to ASIUM, Web->KB (Craven et al., 2000) aims at creating a knowledge base for an existing ontology and takes two inputs. The first one is an ontology denoting the classes and relations of interest when creating the knowledge base. The second one is a set of training data, which consists of labeled regions of hypertext that represent instances of these classes and relations. With the help of these inputs, the system learns to extract information from other pages and hyperlinks on the Web.

Similar to the Web->KB framework, Abraxas (Iria et al., 2006) makes use of input knowledge as an indicator of the anticipated domain. Furthermore, Abraxas utilizes the notion of incrementality for creating and gradually extending existing language resources in terms of one another, with optional and minimal supervision by an expert. The initial input can be either, an ontology, a corpus, patterns or combinations thereof and it can serve as a specification of the domain of interest or as seed data for a bootstrapping cycle. Incrementality, plays an important role in the process and it is likely that the framework will be adjusted for the on-line extension of an ontology in the future.

Semantic caching as a solution for high performance access to ontologies is e.g. described in Hong et al. (2002) and Karnstedt (2003). Even though the advantage of performance gain is also an advantage in our work, the main focus of the construction of a database interface to the ontology is to separate newly and completely automatically acquired knowledge from the manually crafted ontology. Furthermore, information about the confidence of the learned instances can be displayed and used for a final revision as well as a further automatic acquisition in the future.

6. Conclusion and Future Work

In this paper we described the establishment of a database interface which acts as a semantic memory between our incremental OLP tool and the system's knowledge base.

A next step in our work will be to allow the information extraction algorithms to produce more than only one result. For each instance and/or property not only the first but also the second and third best hypotheses delivered by the information extraction module will be stored in the semantic memory. This will allow an expert to choose from more than one option when he is reviewing the newly acquired knowledge. Further, we are working on a component that can allow the user to give feedback about the quality of the results, which could then be used to learn in which context which instance occurs to be the right one.

Acknowledgments

The authors both thank the Klaus Tschira Foundation for financial support.

7. References

Philipp Cimiano. 2006. *Ontology Learning and Population from Text*. Springer, Heidelberg, Germany.

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. 2000. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 118(1/2):69–113.

David Faure and Claire Nedellec. 1999. Knowledge acquisition of predicate argument structures from technical texts using machine learning: The system asium. In *EKAW '99: Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling and Management*, London, UK. Springer-Verlag.

Zheng Hong, Lu Ruqian, Jin Zhi, and Hu Sikang. 2002. Ontology-based semantic cache in aokb. *J. Comput. Sci. Technol.*, 17(5):657–664.

José Iria, Christopher Brewster, Fabio Ciravegna, and Yorick Wilks. 2006. An incremental tri-partite approach to ontology learning. In *Proceedings of the Language Resources and Evaluation Conference (LREC-06), Genoa, Italy 22-28 May*.

Marcel Karnstedt, Kai-Uwe Sattler, Ingolf Geist, and Hagen Hoepfner. 2003. Semantic caching in ontology-based mediator systems. *Berliner XML Tage*, pages 155–169.

Berenike Loos and Chris Biemann. 2007. Supporting web-based address extraction with unsupervised tagging. In *Studies in Classification, Data Analysis, and Knowledge Organization*. Springer.

Berenike Loos. 2006. Scaling natural language understanding via user-driven ontology learning. In *Proceedings of the Third Workshop on Scalable Natural Language Understanding*, pages 33–40, New York City, New York, June. Association for Computational Linguistics.

John Lyons. 1977. *Semantics*. University Press, Cambridge, MA.

Alexander Maedche. 2002. *Ontology Learning for the Semantic Web*. Kluwer Academic Publishers, USA.

Michele Missikoff, Roberto Navigli, and Paola Velardi. 2002. Integrated approach to web ontology learning and engineering. In *IEEE Computer - November*.

Roberto Navigli, Paola Velardi, Alessandro Cucchiarelli, and Francesca Neri. 2004. Extending and Enriching WordNet with OntoLearn. pages 279–284, Brno, Czech Republic, December. Masaryk University Brno, Czech Republic.