Text2Onto

A Framework for Ontology Learning and Data-driven Change Discovery

Philipp Cimiano, Johanna Völker

Institute AIFB, University of Karlsruhe
{pci,jvo}@aifb.uni-karlsruhe.de

Abstract. In this paper we present Text2Onto, a framework for ontology learning from textual resources. Three main features distinguish Text2Onto from our earlier framework TextToOnto as well as other state-of-the-art ontology learning frameworks. First, by representing the learned knowledge at a meta-level in the form of instantiated modeling primitives within a so called Probabilistic Ontology Model (POM), we remain independent of a concrete target language while being able to translate the instantiated primitives into any (reasonably expressive) knowledge representation formalism. Second, user interaction is a core aspect of Text2Onto and the fact that the system calculates a confidence for each learned object allows to design sophisticated visualizations of the POM. Third, by incorporating strategies for data-driven change discovery, we avoid processing the whole corpus from scratch each time it changes, only selectively updating the POM according to the corpus changes instead. Besides increasing efficiency in this way, it also allows a user to trace the evolution of the ontology with respect to the changes in the underlying corpus.

1 Introduction

Since ontologies provide a shared understanding of a domain of interest, they have become a key technology for semantics-driven modeling, especially for the ever-increasing need for knowledge interchange and integration. Semantic annotation of data with respect to a certain ontology makes it machine-processable and allows for exchanging this data between different applications. Therefore, ontologies are frequently used for the explicit representation of knowledge which is implicitly given by various kinds of data. Since building an ontology for a huge amount of data is a difficult and time consuming task a number of tools such as TextToOnto[1] [17], the ASIUM system [8], the Mo'k Workbench [2], OntoLearn [21] or OntoLT [3] have been developed in order to support the user in constructing ontologies from a given set of (textual) data. However, all these tools suffer from several shortcomings.

First of all, they all depend either on very specific or proprietary ontology models which can not always be translated to other formalisms in a straightforward

[1] http://sourceforge.net/projects/texttoonto/

way. This is certainly undesirable as ontology learning tools should be independent from a certain ontology model in order to be widely applicable and used. This is especially important in a context such as the Semantic Web in which different ontology models coexist next to each other. In Text2Onto[2] we overcome this problem by representing the learned ontological structures at a meta-level in form of so called modeling primitives rather than in a concrete knowledge representation language. As in [11], a collection of instantiated modeling primitives can then be translated into any target language. In this way we are able to handle the most prevalent representation languages currently used within the Semantic Web: RDFS, OWL and F-Logic.

Second, the interaction with end-users, in contrast to linguists or machine-learning specialists, has been largely neglected within such systems. As users are typically the ones who are most familiar with the domain, user interaction should be a central part of the system architecture. And third, most of these tools lack a certain robustness with respect to changes made to the data set. In fact, most state-of-the-art systems need to relearn the complete ontology once the underlying corpus has changed.

Text2Onto is a complete re-design and re-engineering of our system TextToOnto, a tool suite for ontology learning from textual data [17]. Text2Onto targets all these problems by introducing two new paradigms for ontology learning: (i) Probabilistic Ontology Models (POMs) which represent the results of the system by attaching a probability to them and (ii) data-driven change discovery which is responsible for detecting changes in the corpus, calculating POM deltas with respect to the changes and accordingly modifying the POM without recalculating it for the whole document collection. The benefits of these key design choices are various.

By assigning probabilities to the learned structures, the interaction with the user can be made more efficient by presenting him the learned structures ranked according to the certainty of the system or only presenting him the results above a certain confidence threshold. Moreover, in Text2Onto we store a pointer for each object in the POM to those parts of the document collection from which it was derived, allowing the user to understand why a certain concept, instance or relation was created and thus increasing the POM's traceability. And finally, the POM allows to maintain even inconsistent alternatives in parallel thus relegating the task of creating a consistent ontology to the user.

The benefits of data-driven change discovery are even more obvious. First, there is no need of processing the whole document collection when it changes thus leading to increased efficiency. Second, the user can explicitly track the changes to the ontology since the last change in the document collection thus being able to trace the evolution of the ontology with respect to changes in the underlying document collection.

This paper describes the framework and architecture of Text2Onto. It does not focus on the evaluation of the single ontology-learning algorithms, which will

[2] http://ontoware.org/projects/text2onto/

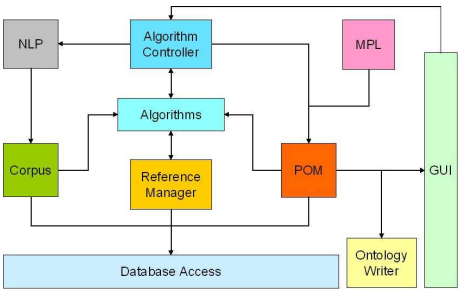

Fig. 1. Architecture of Text2Onto

be presented elsewhere. Nevertheless, we also briefly describe the algorithms implemented in the framework so far.

2 Architecture

The architecture of Text2Onto (cf. figure 1) is centered around the Probabilistic Ontology Model (see Section 2.1) which stores the results of the different ontology learning algorithms (cf. section 2.4). The algorithms are initialized by a controller, the purpose of which is (i) to trigger the linguistic preprocessing of the data, (ii) to execute the ontology learning algorithms in the appropriate order and (iii) to apply the algorithms' change requests to the POM. The fact that none of the algorithms has the permission of directly manipulating the POM guarantees maximum transparency and allows for the flexible composition of arbitrarily complex algorithms as described below.

The execution of each algorithm consists of three phases: First, in the *notification phase*, the algorithm learns about recent changes to the corpus. Second, in the *computation phase*, these changes are mapped to changes with respect to the reference repository, which stores all kinds of knowledge about the relationship between the ontology and the data (e.g. pointers to all occurrences of a concept). And finally, in the *result generation phase*, requests for POM changes are generated from the updated content of the reference repository.
The algorithms provided by the Text2Onto framework can be classified according to two different aspects: *task*, i.e. the kind of modeling primitives (see section 2.1) they produce, and *type*, that means the method which is employed in order to extract instances of the regarding primitives from the text. Each algorithm producing a certain kind of modeling primitive can be configured to apply several algorithms of different types and to combine their requests for POM changes in order to obtain a more reliable probability for each instantiated primitive (cf. [5]). Various types of pre-defined strategies allow for specifying the way the individual probabilities are combined.

2.1 The Probabilistic Ontology Model

A *Probabilistic Ontology Model* (POM) as used by Text2Onto is a collection of instantiated modeling primitives which are independent of a concrete ontology

representation language. In fact, Text2Onto includes a *Modeling Primitive Library* (MPL) which defines these primitives in a declarative fashion. The obvious benefits of defining primitives in such a declarative way are twofold. On the one hand, adding new primitives does not imply changing the underlying framework thus making it flexible and extensible. On the other hand, the instantiated primitives can be translated into any knowledge representation language given that the expressivity of the primitives does not exceed the expressivity of this target language. Thus, the POMs learned by Text2Onto can be translated into various ontology representation languages such as RDFS[3], OWL [4] and F-Logic [14]. In fact we follow a similar approach to knowledge representation as advocated in [11] and [19]. Gruber as well as Staab et al. adopt a translation approach to knowledge engineering in which knowledge is modeled at a meta-level rather than in a particular knowledge representation language and is then translated into different target languages. In Text2Onto we follow this translation-based approach to knowledge engineering and define the relevant modeling primitives in the MPL. So called *ontology writers* are then responsible for translating instantiated modeling primitives into a specific target knowledge representation language. The modeling primitives we use in Text2Onto are given below. The name of the corresponding primitive of Gruber's Frame Ontology is shown in parenthesis where applicable:

- concepts (CLASS)
- concept inheritance (SUBCLASS-OF)
- concept instantiation (INSTANCE-OF)
- properties/relations (RELATION)
- domain and range restrictions (DOMAIN/RANGE)
- mereological relations
- equivalence

It is important to mention that the above list is in no way exhaustive and could be extended whenever it is necessary. The motivation for considering exactly these relations is the fact that the algorithms integrated in the framework are currently only able to learn is-a, instance-of, part-whole as well as equivalence relations and restrictions on the domain and range of relations.
The POM is not probabilistic in a mathematical sense, but because every instantiated modeling primitive gets assigned a value indicating how certain the algorithm in question is about the existence of the corresponding instance. The purpose of these 'probabilities' is to facilitate the user interaction by allowing her to filter the POM and thereby select only a number of relevant instances of modeling primitives to be translated into a target language of her choice.

2.2 Data-driven Change Discovery

In order to define the task of data-driven change discovery we first distinguish between *change capturing* and *change discovery*.

[3] http://www.w3.org/TR/rdf-schema/
[4] http://www.w3.org/TR/owl-features/

Change capturing can be defined as the generation of ontology changes from explicit and implicit requirements. Explicit requirements are generated, for example, by ontology engineers who want to adapt the ontology to new requirements or by the end-users who provide the explicit feedback about the usability of ontology entities. The changes resulting from this kind of requirements are called top-down changes. Implicit requirements leading to so-called bottom-up changes are reflected in the behavior of the system and can be induced by applying change discovery methods.
Change discovery aims at generating implicit requirements by inducing ontology changes from existing data. [20] defines three types of change discovery: (i) structure-driven, (ii) usage-driven and (iii) data-driven. Whereas structure-driven changes can be deduced from the ontology structure itself, usage-driven changes result from the usage patterns created over a period of time. Data-driven changes are generated by modifications to the underlying data, such as text documents or a database, representing the knowledge modeled by an ontology. Therefore, *data-driven change discovery* provides methods for automatic or semi-automatic adaption of an ontology according to modifications being applied to the underlying data set.
The benefits of data-driven change discovery are twofold. First, an elaborated change management system enables the user to explicitly track the changes to the ontology since the last change in the document collection thus being able to trace the evolution of the ontology with respect to changes in the underlying document collection. Second and even more important, there is no longer the need of processing the whole document collection when it changes thus leading to increased efficiency.
Independently from a particular application scenario some requirements have to be met by any application which is designed to support data-driven change discovery. The most important one is, of course, the need to keep track of all changes to the data. Each change must be represented in a way which allows for associating with it various kinds of information, such as its type, the source it has been created from and its target object (e.g. a text document). In order to make the whole system as transparent as possible not only changes to the data set, but also changes to the ontology should be logged. If ontological changes are caused by changes to the underlying data, the former should be associated with information about the corresponding modification to the data. Moreover, the system should allow for defining various *change strategies*, which specify the degree of influence changes to the data have with respect to the ontology or the POM respectively. This permits to take into account the confidence the user has in different data sources or the fact that documents might become out-dated after a while.
It is quite obvious that each algorithm in Text2Onto supporting automatic or semi-automatic data-driven change discovery requires a formal, explicit representation of two kinds of knowledge: First, knowledge about which concepts, instances and relations are affected by certain changes to the data and second, knowledge about how to react to these changes in an appropriate way, i.e. how

to update the POM in response to these changes. Consequently, the concrete knowledge to be stored by an ontology extraction system depends on the way these algorithms are implemented. A concept extraction algorithm, for example, might need to store the text references and term frequencies associated with each concept, whereas a pattern-based concept classification algorithm might have to remember the occurrences of all patterns matched in the text. Therefore, in Text2Onto each type of algorithm is provided with a suitable reference store (see section 2). It is among the algorithm controller's tasks to set up a suitable store each time a new algorithm is added.

2.3 Natural Language Processing

Many existing ontology learning environments focus either on pure machine learning techniques [2] or rely on linguistic analysis [3, 21] in order to extract ontologies from natural language text. Text2Onto combines machine learning approaches with basic linguistic processing such as tokenization or lemmatizing and shallow parsing. Since it is based on the GATE framework [7] it is very flexible with respect to the set of linguistic algorithms used, i.e. the underlying GATE application can be freely configured by replacing existing algorithms or adding new ones such as a deep parser if required. Another benefit of using GATE is the seamless integration of JAPE which provides finite state transduction over annotations based on regular expressions.
Linguistic preprocessing in Text2Onto starts by tokenization and sentence splitting. The resulting annotation set serves as an input for a POS tagger which in the following assigns appropriate syntactic categories to all tokens. Finally, lemmatizing or stemming (depending on the availability of the regarding processing components for the current language) is done by a morphological analyzer and a stemmer respectively.
After the basic linguistic preprocessing is done, a JAPE transducer is run over the annotated corpus in order to match a set of particular patterns required by the ontology learning algorithms. Whereas the left hand side of each JAPE pattern defines a regular expression over existing annotations, the right hand side describes the new annotations to be created (see listing 1.1). For Text2Onto we developed JAPE patterns for both shallow parsing and the identification of modeling primitives, i.e. concepts, instances and different types of relations (c.f. [13]).

Listing 1.1. JAPE pattern: Hearst

```
 (NounPhrase1):superconcept
 (
  {Token.kind == punctuation}
 )?
 {SpaceToken.kind == space}
 {Token.string == "such"}
 {SpaceToken.kind == space}
 {Token.string == "as"}
 {SpaceToken.kind == space}

 (NounPhrasesAlternatives):subconcept
):hearst1
-->
:hearst1.SubclassOfRelation = { rule = "Hearst1" },
:subconcept.Domain = { rule = "Hearst1" },
:superconcept.Range = { rule = "Hearst1" }
```

Since obviously, both types of patterns are language specific, different sets of patterns for shallow parsing and ontology extraction have to be defined for each language. Because of this and due to the fact that particular processing components for GATE have to be available for the regarding language, Text2Onto currently only supports ontology learning from English texts. Fortunately, thanks to recent research efforts made in the SEKT project[5] GATE components for the linguistic analysis of various languages such as German and Spanish have been made available recently. Since we want to provide full support for all of these languages in future releases of Text2Onto, we have already integrated some of these components, and we are currently working on the development of appropriate patterns for Spanish and German.

2.4 Algorithms

This section briefly describes for each modeling primitive the algorithms used to learn corresponding instances thereof. In particular we describe the way the probability for an instantiated modeling primitive is calculated.

Concepts In Text2Onto we have implemented several measures to assess the relevance of a certain term with respect to the corpus in question. In particular, we implemented different algorithms calculating the following measures: Relative Term Frequency (RTF), TFIDF (Term Frequency Inverted Document Frequency), Entropy and the C-value/NC-value method in [10]. For each term, the values of these measures are normalized into the interval [0..1] and used as corresponding probability in the POM.

Subclass-of Relations In order to learn subclass-of relations, in Text2Onto we have implemented various algorithms using different kinds of sources and techniques following the approach in [5]. In particular we implemented algorithms exploiting the hypernym structure of WordNet [9], matching Hearst patterns [13] in the corpus as well as in the World Wide Web and applying linguistic heuristics mentioned in [21]. The results of the different algorithms are then combined through combination strategies as described in [5]. This approach has been evaluated with respect to a collection of tourism-related texts by comparing the results with a reference taxonomy for this domain. The best result obtained was an F-Measure of 21.81%, a precision of 17.38% and a recall of 29.95%. As the algorithm already indicates the confidence in its prediction with a value between 0 and 1, the probability given in the POM can be set accordingly.

Mereological Relations For the purpose of discovering mereological (part-of) relations in the corpus, we developed JAPE expressions matching the patterns described in [4] and implemented an algorithm counting the occurrences of patterns indicating a part-of relation between two terms t_1 and t_2, i.e. part-of(t_1,t_2).

[5] www.sekt-project.com

The probability is then calculated by dividing by the sum of occurrences of patterns in which t_1 appears as a part. Further, as in the algorithm described above we also consult WordNet for mereological relations and combine the elementary probabilities with a certain combination strategy.

General Relations In order to learn general relations, Text2Onto employs a shallow parsing strategy to extract subcategorization frames enriched with information about the frequency of the terms appearing as arguments. In particular, it extracts the following syntactic frames:

- transitive, e.g. love(subj,obj)
- intransitive + PP-complement, e.g. walk(subj,pp(to))
- transitive + PP-complement, e.g. hit(subj,obj,pp(with))

and maps this subcategorization frames to ontological relations. For example, given the following enriched subcategorization frame

hit(subj:*person*,obj:*thing*,with:*object*)

the system would update the POM with these relations:

hit(domain:*person*,range:*thing*)
hit_with(domain:*person*,range:*object*)

The probability of the relation is then estimated on the basis of the frequency of the subcategorization frame as well as of the frequency with which a certain term appears at the argument position in question.

Instance-of Relations In order to assign instances or named entities appearing in the corpus to their correct concept in the ontology, Text2Onto relies on a similarity-based approach extracting context vectors for instances and concepts from the text collection and assigning instances to the concept corresponding to the vector with the highest similarity with respect to their own vector as in [1]. As similarity measure we use the Skewed divergence presented in [15] as it was found to perform best in our experiments. Using this similarity measure as well as further heuristics, we achieved an F-Measure of 32.6% when classifying instances with respect to an ontology comprising 682 concepts [6]. Alternatively, we also implemented a pattern-matching algorithm similar to the one used for discovering part-of relations (see above).

Equivalence Following the assumption that terms or concepts are equivalent to the extent to which they share similar syntactic contexts, we implemented algorithms calculating the similarity between terms on the basis of contextual features extracted from the corpus, whereby the context of a terms varies from simple word windows to linguistic features extracted with a shallow parser. This corpus-based similarity is then taken as the probability for the equivalence of the concepts in question.

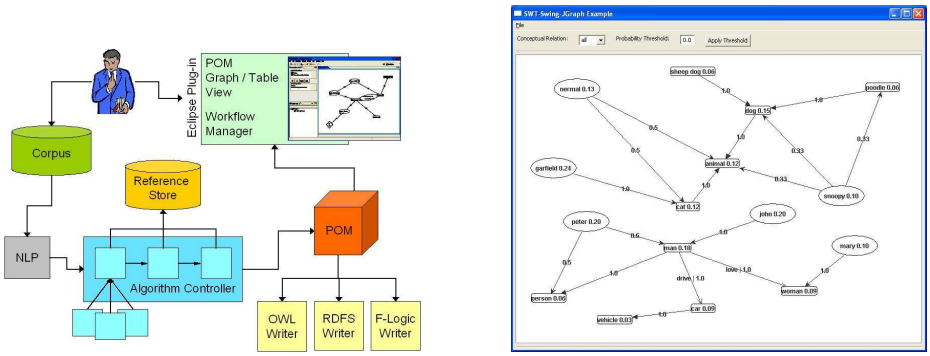

Fig. 2. Usage Scenario (left) and POM visualization (right)

3 Graphical User Interface

In addition to the core functionality of Text2Onto described above we developed a graphical user interface featuring a corpus management component, a workflow editor, configuration dialogues for the algorithms as well as tabular and graph-based POM visualizations. It will be available as an Eclipse[6] plug-in which could facilitate a smooth integration into ontology editors at a later development stage.

A typical usage scenario for Text2Onto is depicted by figure 2 (left). The user specifies a corpus, i.e. a collection of text, HTML or PDF documents, and starts the graphical workflow editor. The editor provides her with a list of algorithms which are available for the different ontology learning tasks, and assists her in setting up an appropriate workflow for the kind of ontology she wants to learn as well as to customize the individual ontology learning algorithms to be applied. Once the ontology learning process is started, the corpus gets preprocessed by the natural language processing component described in section 2.3, before it is passed to the algorithm controller. In the following, depending on the configuration of the previously specified workflow, a sequence of ontology learning algorithms is applied to the corpus. Each algorithm starts by detecting changes in the corpus and updating the reference store accordingly. Finally, it returns a set of requests for POM changes to its caller, which could be the algorithm controller, but also a more complex algorithm (cf. section 2). After the process of ontology extraction is finished, the POM is presented to the user.

Since the POM unlike any concrete ontology is able to maintain thousands of conflicting modeling alternatives in parallel, an appropriate and concise visualization of the POM is of crucial importance for not overwhelming the user with too much information. Although several pre-defined filters such as a probability threshold will be available for *pruning* the POM, some user interaction might still be needed for transforming the POM into a high-quality ontology. Currently, two different visualization types are available: a tabular view showing a number of sorted lists for all kinds of modeling primitives and a graph-based representation which is depicted by figure 2 (right). After having finished her interaction with the POM, i.e. after adding or removing concepts, instances or

[6] http://www.eclipse.org

relations, the user can select among various ontology writers, which are provided for translating the POM into different ontology representation languages.

4 Related Work

Several ontology learning frameworks have been designed and implemented in the last decade. The Mo'K workbench [2], for instance, basically relies on unsupervised machine learning methods to induce concept hierarchies from text collections. In particular, the framework focuses on agglomerative clustering techniques and allows ontology engineers to easily experiment with different parameters. OntoLT [3] is an ontology learning plug-in for the Protégé ontology editor. It is targeted more at end users and heavily relies on linguistic analysis. It basically makes use of the internal structure of noun phrases to derive ontological knowledge from texts.
The framework by Velardi et al., OntoLearn [21], mainly focuses on the problem of word sense disambiguation, i.e. of finding the correct sense of a word with respect to a general ontology or lexical database. In particular, they present a novel algorithm called SSI relying on the structure of the general ontology for this purpose. Furthermore, they include an explanation component for users consisting in a gloss generation component which generates definitions for concepts which were found relevant in a certain domain.
TextToOnto [17] is a framework implementing a variety of algorithms for diverse ontology learning subtasks. In particular, it implements diverse relevance measures for term extraction, different algorithms for taxonomy construction as well as techniques for learning relations between concepts [16]. The focus of TextToOnto has been so far on the algorithmic backbone with the result that the combination of different algorithms as well as the interaction with the user had been neglected so far. The successor Text2Onto targets exactly these issues by introducing the POM as a container for the results of different algorithms as well as adding probabilities to the learned structures to facilitate the interaction with the user.
Common to all the above mentioned frameworks is some sort of natural language processing to derive features on the basis of which to learn ontological structures. However, all these tools neglect the fact that the document collection can change and that it is unfeasible to start the whole learning process from scratch. Text2Onto overcomes this shortening by storing current results in stores and calculating POM deltas caused by the addition or deletion of documents. Very related is also the approach of [12] in which a qualitative calculus is presented which is able to reason on the results of different algorithms, resolving inconsistencies and exploiting synergies. Interesting is the dynamic aspect of the approach, in which the addition of more textual material leads to a reduction in the number of hypothesis maintained in parallel by the system.
Furthermore, as argued in the introduction, previously developed ontology learning frameworks all lack an explanation component helping the user to understand why something has changed in the underlying POM. In addition, most tools do

not indicate how certain a learned object actually is, thus making it more difficult for the user to select only the most reliable findings of the system.

5 Conclusion and Outlook

We have presented our framework Text2Onto with the aim of learning ontologies from textual data. Its novel aspects as compared to similar frameworks are: (i) the independence of the actual ontology model or knowledge representation language, (ii) the introduction of probabilistic ontology models allowing more sophisticated models of user interaction and (iii) the integration of data-driven change discovery strategies increasing the efficiency of the system as well as the traceability of the learned ontology with respect to changes in the corpus, thus making the whole process more transparent.
In future versions of Text2Onto a graphical workflow engine will provide support for the automatic or semi-automatic composition of complex ontology learning workflows. For transforming the POM into a consistent OWL or RDF ontology we aim at a tight integration with the KAON evolution framework [20] which will allow to detect and resolve inconsistencies in the generated POMs. The development of the explanation component will be carried on with particular regard to the DILIGENT methodology [18]. By generating machine readable explanations we will make a major step in the direction of making Text2Onto part of the DILIGENT process. We are currently preparing an evaluation setting for comparing the results of the newly developed ontology learning algorithms with previous implementations provided by TextToOnto and other ontology learning tools. Moreover, a detailed user evaluation will offer valuable clues to the usability of the graphical user interface and the benefits gained from the availability of an explanation component.

6 Acknowledgments

Research reported in this paper has been partially financed by the EU in the IST-2003-506826 project SEKT (http://www.sekt-project.com) and the IST-2001-34038 project Dot.Kom (http://www.dot-kom.org). We would like to thank our students Simon Sparn, Stephan Oehlert, Günter Ladwig and Matthias Hartung for their assistance in implementing the system and all our colleagues for fruitful discussions.

References

1. E. Alfonseca and S. Manandhar. Extending a lexical ontology by a combination of distributional semantics signatures. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, 2002.
2. G. Bisson, C. Nedellec, and L. Canamero. Designing clustering methods for ontology building - The Mo'K workbench. In *Proceedings of the ECAI Ontology Learning Workshop*, pages 13–19, 2000.

3. P. Buitelaar, D. Olejnik, and M. Sintek. OntoLT: A protégé plug-in for ontology extraction from text. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2003.
4. E. Charniak and M. Berland. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 57–64, 1999.
5. P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. Learning taxonomic relations from heterogeneous sources. In *Proceedings of the ECAI 2004 Ontology Learning and Population Workshop*, 2004.
6. P. Cimiano and J. Völker. Towards large-scale, unsupervised and ontology-based named entity recognition. Technical Report. AIFB, University of Karlsruhe, 2004.
7. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Annual Meeting of the ACL*, 2002.
8. D. Faure and C. Nedellec. A corpus-based conceptual clustering method for verb frames and ontology. In *Proceedings of the LREC Workshop on Adapting lexical and corpus resources to sublanguages and applications*, 1998.
9. C. Fellbaum. *WordNet, an electronic lexical database*. MIT Press, 1998.
10. K. Frantzi, S. Ananiadou, and J. Tsuji. The c-value/nc-value method of automatic recognition for multi -word terms. In *Proceedings of the ECDL*, pages 585–604, 1998.
11. T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
12. U. Hahn and K. Schnattinger. Towards text knowledge engineering. In *AAAI/IAAI*, pages 524–531, 1998.
13. M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.
14. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, 1995.
15. L. Lee. Measures of distributional similarity. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 25–32, 1999.
16. A. Maedche and S. Staab. Discovering conceptual relations from text. In W. Horn, editor, *Proceedings of the 14th European Conference on Artificial Intellignece (ECAI'2000)*, 2000.
17. A. Maedche and S. Staab. Ontology learning. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 173–189. Springer, 2004.
18. H. S. Pinto, C. Tempich, and S. Staab. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engingeering of ontologies. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, 2004.
19. S. Staab, E. Erdmann, and A. Maedche. Engineering ontologies using semantic patterns. In *Proceedings of the IJCAI'01 Workshop on E-Business and Intelligent Web*, 2001.
20. L. Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, University of Karlsruhe, 2004.
21. P. Velardi, R. Navigli, A. Cuchiarelli, and F. Neri. Evaluation of ontolearn, a methodology for automatic population of domain ontologies. In P. Buitelaar, P. Cimiano, and B. Magnini, editors, *Ontology Learning from Text: Methods, Applications and Evaluation*. IOS Press, 2005. to appear.