The Ontology Extraction & Maintenance Framework Text-To-Onto

Alexander Maedche[1] and Raphael Volz[2]
[1]Forschungszentrum Informatik FZI, 76131 Karlsruhe,
maedche@fzi.de, http://www.fzi.de/wim
[2]Institute AIFB, University of Karlsruhe, 76128 Karlsruhe,
volz@aifb.uni-karlsruhe.de, http://www.aifb.uni-karlsruhe.de/WBS

Abstract

Ontologies play an increasingly important role in Knowledge Management. One of the main problems associated with ontologies is that they need to be constructed and maintained. Manual construction of larger ontologies is usually not feasible within companies because of the effort and costs required. Therefore, a semi-automatic approach to ontology construction and maintenance is what everybody is wishing for. The paper presents a framework for semi-automatically learning ontologies from domain-specific texts by applying machine learning techniques. The TEXT-TO-ONTO framework integrates manual engineering facilities to follow a *balanced cooperative modelling* paradigm.

1 Introduction

In recent years Knowledge Management (KM) has become an important success factor for enterprises. Increasing product complexity, globalization, virtual organizations and customer orientation demand a more thorough and systematic management of knowledge — within an enterprise and between several cooperating enterprises. IT-supported KM solutions are often built around an organizational memory that integrates informal, semi-formal and formal knowledge to facilitate the access, sharing and reuse of knowledge by members of the organization(s) to solve their individual or collective tasks. In such a context, knowledge has to be modelled, appropriately structured and interlinked to support flexible integration and its personalized presentation to the consumer.

Ontologies have shown to be the right answer to the structuring and modelling problems arising in Knowledge Management. They provide a formal conceptualization of a particular domain that can be shared by a group of people (in and between organizations). Ontologies provide a sound semantic basis for the definition of meaning. They are typically used to provide the semantics for communications among humans and machines [18]. Therefore, they are also a means to provide the kind of formal semantics needed in sophisticated KM systems and can represent the conceptual backbone for the KM infrastructure. In this paper, we address several challenges in ontology engineering:

1. How to simplify and accelerate ontology construction using data mining techniques.
2. How to ensure that the induced ontology faithfully reflects application data requirements.

These challenges are essential for the application of ontologies in corporate Knowledge Management scenarios. The time needed for ontology construction is immediately associated with the cost and acceptance of a KM system. Dynamically evolving environments like corporate intranets require constant evolution and refinement of the ontology to ensure that the ontology reflects the managed content.

The TEXT-TO-ONTO framework takes a new approach: It leverages data mining and natural language processing techniques to assist the user in the development and maintenance task by analyzing web data[1]

[1]Especially for untrained knowledge engineers our assumption that most concepts and conceptual relations of a specific domain can be found in existing web data that describe the domain, holds.

(e.g. HTML free text, dictionaries etc.) and suggests modelling decisions.

To put our approach into practice the results of the implemented data mining techniques are aligned with the modelling primitives given in our ontology model. This enables the combination of results and realizes a multi-strategy learning architecture [20], which supports balancing between the advantages and disadvantages of the different data mining techniques.

Our implementation also follows the *balanced cooperative modeling* paradigm established by Morik [22]. Her work describes the interaction between knowledge acquisition and machine learning, where each modeling step can be done either by human or by machine. A stand-alone manual ontology engineering environment ONTOEDIT is fully embedded in TEXT-TO-ONTO, thus manual ontology construction is possible. The user can also modify propositions from TEXT-TO-ONTO'S extraction and maintenance components graphically or can reject them.

The content of this paper is structured as follows. Section 2 gives a formal basis for our notion of ontologies and identifies the modeling steps humans usually perform in ontology engineering. In section 3 the overall architecture of TEXT-TO-ONTO is presented. Section 4 illustrates how the framework can be used for ontology extraction and emphasizes TEXT-TO-ONTO's capabilities in the increasingly important field of ontology maintenance. Section 5 gives a suggestion how the framework could be evaluated and adopts standard measures from information retrieval for this purpose. We conclude with contrasting our work with related efforts and give an outlook for further work.

2 A Notion of Ontologies

This section gives a brief definition of the entities that define an ontology with respect to our ontology learning task and the environment TEXT-TO-ONTO. A basic building block for ontologies are concepts. Typically conceps are hierarchically organized in a concept hierarchy. We define a set of concepts and a concept hierarchy as follows:

- A set $\mathcal{C}$ whose elements are called concepts
- A **concept hierarchy** $\mathcal{H}^{\mathcal{C}}$: $\mathcal{H}^{\mathcal{C}}$ is a directed relation $\mathcal{H}^{\mathcal{C}} \subseteq \mathcal{C} \times \mathcal{C}$ which is called concept hierarchy or taxonomy. $\mathcal{H}^{\mathcal{C}}(C_1, C_2)$ means that C_1 is a subconcept of C_2.

Concepts and the concept hierarchy are further extended with non-taxonomic relations between concepts and a set of axioms. We define them as follows:

- A set $\mathcal{R}$ whose elements are called relations, the sets $\mathcal{C}$ and $\mathcal{R}$ are disjoint
- A **function** $rel : \mathcal{R} \rightarrow \mathcal{C} \times \mathcal{C}$, that relates concepts non-taxonomically[2]. The function dom: $\mathcal{R} \rightarrow \mathcal{C}$ with $\text{dom}(R) := \Pi_1(rel(R))$ gives the domain of R, and range: $\mathcal{R} \rightarrow \mathcal{C}$ with $\text{range}(R) := \Pi_2(rel(R))$ give its range. For $rel(R) = (C_1, C_2)$ we also write $R(C_1, C_2)$.
- A set of ontology **axioms** $\mathcal{A}^{\mathcal{O}}$, expressed in an appropriate logical language, e.g. first order logic.

For the operation of TEXT-TO-ONTO's components it is necessary to provide a link from document content (specifically single words) to ontological entities. This mapping is provided by a lexicon. A lexicon for the ontology structure $\mathcal{O} := \{\mathcal{C}, \mathcal{R}, \mathcal{H}^{\mathcal{C}}, rel, \mathcal{A}^{\mathcal{O}}\}$ is a 4-tupel $\mathcal{L} := \{\mathcal{L}^{\mathcal{C}}, \mathcal{L}^{\mathcal{R}}, \mathcal{F}, \mathcal{G}\}$ consisting of

- Two sets $\mathcal{L}^{\mathcal{C}}$ and $\mathcal{L}^{\mathcal{R}}$, whose elements are called **lexical entries** for concepts and relations, respectively.
- Two relations $\mathcal{F} \subseteq \mathcal{L}^{\mathcal{C}} \times \mathcal{C}$ and $\mathcal{G} \subseteq \mathcal{L}^{\mathcal{R}} \times \mathcal{R}$ called **references** for concepts and relations, respectively. Based on $\mathcal{F}$, let for $L \in \mathcal{L}^{\mathcal{C}}$, $\mathcal{F}(L) = \{C \in \mathcal{C} | (L, C) \in \mathcal{F}\}$ and for and $\mathcal{F}^{-1}(C) = \{L \in \mathcal{L}^{\mathcal{C}} | (L, C) \in \mathcal{F}\}$ ($\mathcal{G}$ and $\mathcal{G}^{-1}$ are defined analogously).

[2] In this generic definition one does not distinguish between relations and attributes.

Formal semantics for ontologies is a *sine qua non*, in our implementation we use Frame-Logics [12] and its concrete implementation in the SilRI inference engine (cf. [2]) to provide this for the above definitions. By using F-Logics we are additionally able to state domain-specific axioms $\mathcal{A}^{\mathcal{O}}$(cf. [26]) and a knowledge base consisting of instances and relations between them. A more comprehensive explanation of the ontology structure introduced above and the definition of an associated knowledge base structure is given in [18].

An Example. The following example illustrates an instantiated ontology: Assume $\mathcal{C} := \{x_1, x_2, x_3\}$ and $\mathcal{R} := \{x_4\}$. The following hierarchical relation, $\mathcal{H}^{\mathcal{C}}(x_2, x_1)$, and, the non-taxonomic relation, $x_4(x_2, x_3)$, is defined. The lexicon is given as $\mathcal{L}^{\mathcal{C}} = \{$"Person", "Employee", "Organization"$\}$ and $\mathcal{L}^{\mathcal{R}} = \{$"works at organization"$\}$. The function $\mathcal{F}$ and $\mathcal{G}$ map the lexical entries to the concepts and relations of the ontology. $\mathcal{F}$ is applied as follows: $\mathcal{F}($"Person"$) = x_1, \mathcal{F}($"Employee"$) = x_2, \mathcal{F}($"Organization"$) = x_3$ and $\mathcal{G}($"works at organization"$) = x_4$. Figure 1 depicts this small example graphically.

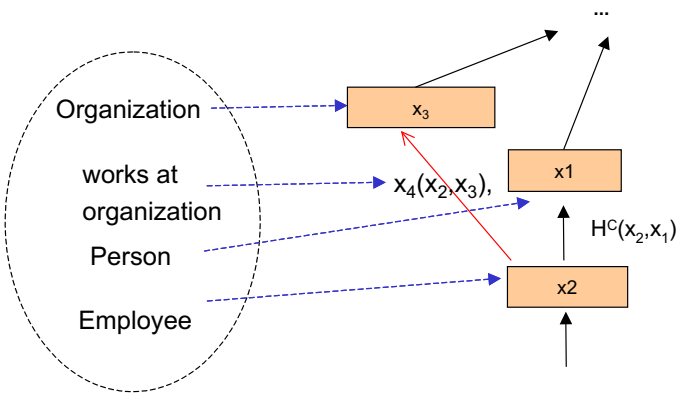

Figure 1: Example of an instantiated ontology structure

3 Architecture

Our overall goal is to learn and maintain ontologies from web data. To reach this goal we are applying several processing steps. First, web data is pre-processed by a resource processing component and a natural language processing system. Then, machine learning algorithms are applied. The results of these learning algorithms are standardized towards the modelling primitives described in section 2 and can be combined and modified by the user. For user modeling OntoEdit, a system for manual ontology engineering, is incorporated into the system. The main components of the Text-To-Onto system and their interactions are depicted in figure 2. We follow and refine a more general architecture for the approach of semi-automatic ontology learning from natural language that has been described in [16]. The central component in our architecture is the core ontology model, which is an implementation of the ontology model presented in section 2.

3.1 Data Import and Processing

The data import and processing modules facilitate the extraction and preparation of natural-language texts from web documents. Normally, web data must be prepared for natural language processing. Thus, mechanisms to extract plain text from HTML, PDF and PostScript are provided. The fact that abbreviations and acronyms are often in web documents requires further pre-processing before starting the natural language processing itself. Therefore facilities for the management of substitution rules (based on regular expressions) are provided in the framework. Applying these substitutions tremendously improved the actual learning tasks.

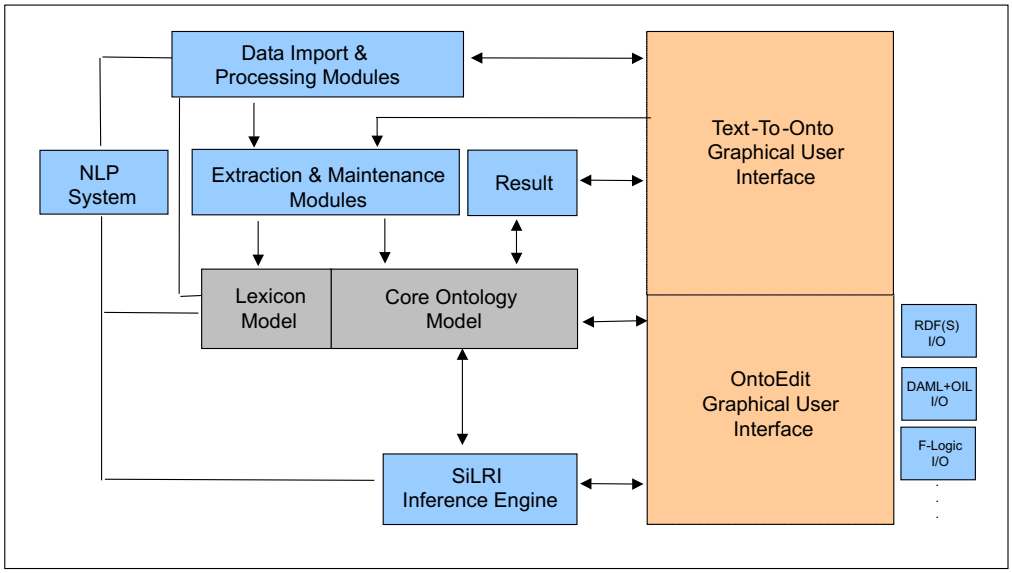

Figure 2: Components of Text-To-Onto

3.2 Natural Language Processing System

On top of the prepared plain text natural language processing is applied. The necessary functionality is provided by the system SMES (Saarbruecken Message Extraction System), a shallow text processor for German (cf. [23]). We will give a short survey of the functionality provided by SMES to illustrate what kind of data is actually passed to the learning algorithms.

SMES is a generic natural language processing component that adheres to several principles that are crucial for our objectives: The architecture of SMES comprises of a tokenizer based on regular expressions, a lexical analysis component including a general word lexicon and a domain-specific lexicon (the lexicon of ontology structure introduced in definition 2) and a chunk parser. The tokenizer scans the text to identify boundaries of words, recognizes complex expression like "$20.00" and returns a token stream. The user is able to provide domain-specific words like department names, which was very important in one of our case studies [28]. The lexicon is shared with the ontology $\mathcal{O}$, this links word stems to concepts in the ontology. Morphological analysis is provided on top of the token stream. This returns a sequence of tuples, each containing word stem and morphological information (i.e. part-of-speech, flexion etc.). The next layer, again performing on the output provided by morphological analysis, is a cascaded chunk parser based on finite state grammars for named entity recognition. It provides phrase recognition (returning nominal phrases, verb phrases and verb groups) and sentence pattern recognition (based on dependency theory). Phrases and sentence patterns can also reference ontological elements. Each level of output can be used by the learning algorithms.

3.3 Algorithm Library

The algorithm library supports several distinct ontology engineering tasks, which can be grouped into two task sets:

- First, we concentrated on algorithms for **ontology extraction**. This involves algorithms for extracting the set of concepts $\mathcal{C}$, their lexical entries $\mathcal{L}^{\mathcal{C}}$, a taxonomic order on these concepts $\mathcal{H}^{\mathcal{C}}$ and a set of relations $\mathcal{R}$ between concepts. A more detailed explanation is given in section 4.
- Second, we developed algorithms for **ontology maintenance**. This refers particularly to algorithms for ontology reduction (also called pruning) and algorithms for ontology evolution (also called refinement). These algorithms can be used to ensure that the induced ontology faithfully reflects the application data requirements. Section 5 gives a detailed explanation.

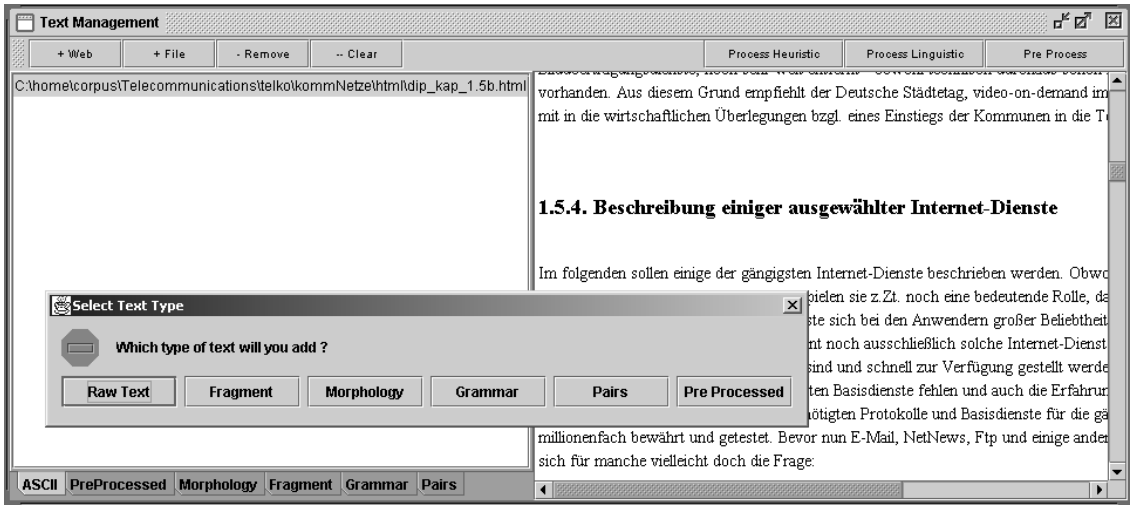

Figure 3: Screenshot – Data Selection and Processing

3.4 Result Presentation

It is necessary to standardize the output in a common way to be able to combine the results from different extraction algorithms. Therefore a common result structure for all learning methods is provided (see figure 4). Results are combined if several learning algorithms obtain identical results. Identical results are presented to the user only once. This implements the mentioned multi-strategy learning approach. This is better suited for the complex task of ontology engineering, as the benefits and drawbacks of the individual algorithms can be balanced.

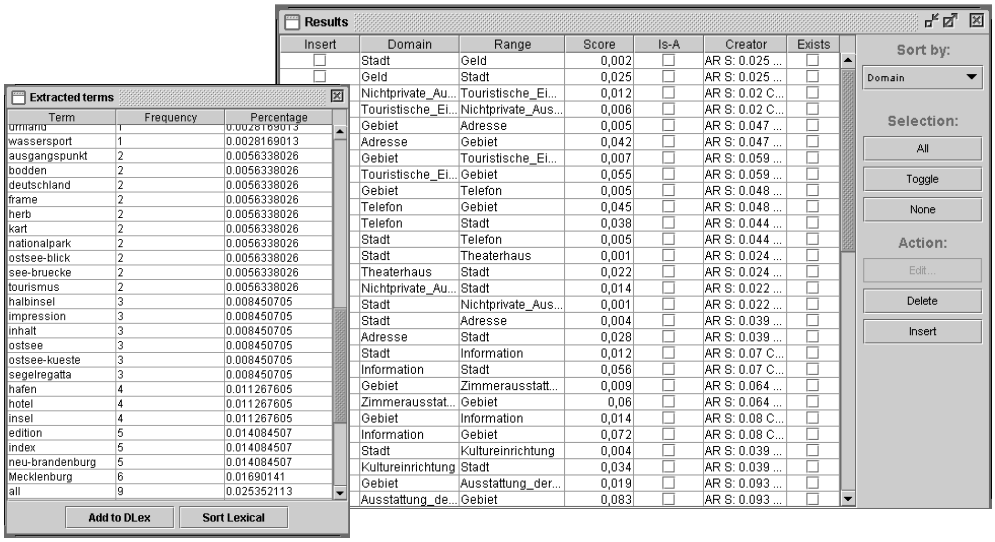

Figure 4: Result Views - The window on the left depicts the single entity view for extracted lexical entries and concepts. The window on the right shows the result for binary relations between entities.

Results are presented in two different views (see figure 4): The first view presents the extracted entities (e.g. entries in $\mathcal{L}^C$). The second view depicts the extracted binary relations between entities ($\mathcal{H}^C, R$).

3.5 Ontology Engineering Environment

Our approach of semi-automatic ontology acquisition concedes extended support for manual ontology engineering. To implement this approach, a system for manual ontology modeling and visualization, called ONTOEDIT[3], is incorporated into TEXT-TO-ONTO. It allows to edit and to browse existing ontological structures. ONTOEDIT provides different views for manual engineering ontologies:

- definition of concepts C
- organizing the concepts C in the concept hierarchy H^C
- modeling of non-taxonomic relations R
- definition of the lexicalizations $\mathcal{L}$ of concepts and relations

ONTOEDIT supports different formats including the internal ontology representation in XML (OXML), serial Frame-Logic syntax [12], RDF-Schema [1] and DAML+OIL [4]. By providing several import / export filters to other ontology representation languages the capabilities of the system can be reused in external applications. As already mentioned above, ONTOEDIT accesses the F-Logic inference engine described in further detail in [2, 3].

3.6 Text-To-Onto Graphical User Interface

Especially business scenarios demand that KM software can be used easily. To meet this demand, TEXT-TO-ONTO provides sophisticated user interfaces to simplify the interaction with the framework. They enable selection of relevant data, the application of processing and transformation techniques or starting a specific extraction mechanism. Data processing may also be triggered by the selection of an ontology learning algorithm that requires a specific representation. Results are presented to the ontology engineer using different views onto the structures (e.g. lists, graph-based, ...). Example user interfaces are shown in figures 4 and 5. We emphasize this aspect, due to the experience gained in the case studies [11] where ontology learning was deployed in real-world ontology engineering.

4 Ontology Extraction & Maintenance

This section illustrates how TEXT-TO-ONTO can be used to extract and maintain ontologies from web data. Our approach reflects that KM systems are dynamically evolving environments. Therefore the learning process always takes the existing conceptual structures into consideration. The algorithms can work from from scratch, though, if no conceptual structures are available. As seen in our case studies (see [11]) some kind of "knowledge structure" (e.g., in the form of a thesaurus, or by looking up WordNet[4], a existing company dictionary) are very beneficial for the learning tasks. Thus, our approach is incremental to ensure that the ontology faithfully reflects the current data.

In this paper we provide an overview from a tool perspective. The implemented algorithms are presented with respect to their applicability for extraction and maintenance. We do not present the actual learning algorithms (and their evaluation) from a technical perspective, this has been done before (see [17, 11, 14, 15]).

4.1 Ontology Extraction

Naturally the results of the ontology extraction algorithms are elements of the ontology structure $\mathcal{O}$. Accordingly the user can extract:

1. Lexical entries referring to concepts $\mathcal{L}^{\mathcal{C}}$ and concepts $\mathcal{C}$
2. The concept hierarchy $\mathcal{H}^{\mathcal{C}}$

[3] A comprehensive description of the ontology engineering system ONTOEDIT and the underlying methodology is given in [27]

[4] WordNet is a lexical semantic net for the English language, see http://www.cogsci.princeton.edu/ wn/.

3. Lexical signs for relations $\mathcal{L}^{\mathcal{R}}$ and non-taxonomic relations $\mathcal{R}$

The reader may note, that the presented sequence corresponds to most ontology engineering methodologies, which start with identifying target concepts and continue with the actual conceptualization of the identified concepts, which is put into practise by construction of the concept hierarchy and establishing conceptual relations. The implemented algorithms can also be distinguished from a methodological point of view into the following approaches.

Statistical & Data Mining Approaches. We implemented several statistical and data mining based algorithms. One algorithm, presented in detail in [28], retrieves term frequencies from text (in analogy to the algorithms described in [24, 19]). The output of this algorithm can be used for the creation of concepts and their corresponding lexical signs. To derive the concept hierarchy, we adopted a hierarchical clustering algorithm[5] that accesses background knowledge from existing ontological entities to label the extracted hierarchy.

To acquire of non-taxonomic conceptual relations we implemented an algorithm that is based on frequent couplings of concepts by identifying linguistically related pairs of words (see [14] for in depth coverage). The actual algorithm is a slightly modified version of the standard association rule algorithm (Generalized Apriori, see [25]) that operates on multi-sets to find reflexive binary rules. The background knowledge from the taxonomy is also used to propose relations at the appropriate level of abstraction.

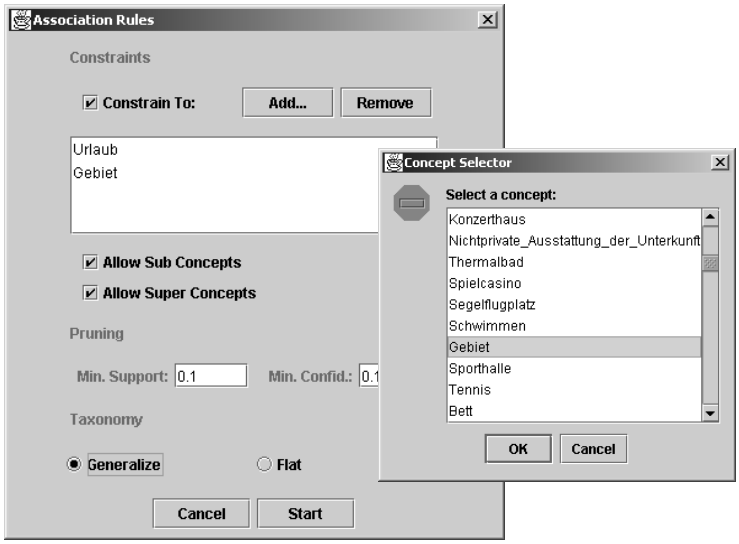

Figure 5: Non-taxonomic Relation Extraction with Association Rule

For instance, the linguistic processing may find that the concept TARIFF frequently co-occurs with each of the concepts MOBILE_CALL, CITY_CALL, and INTERNATIONAL_CALL in sentences such as "Special tariffs are calculated for mobile calls." (the lexical entry "tariffs" refers to the concept TARIFF, the lexical entry "mobile calls" refers to the concept MOBILE_CALL). Finally, the algorithm may recommend to introduce a new relation HAS_TARIFF(CALL,TARIFF) between the concepts CALL and TARIFF, which the user may name with the lexical entry "has tariff".

Again extensive user support must be supplied to controll algorithm parameters. Figure 5 depicts the graphical interface for starting the relation extraction algorithm. The dialogue in the left side of figure 5 offers different configurations that may be defined by the user: First, the algorithm may be constraint to specific concepts (e.g. as in the example to URLAUB and GEBIET). Second, the user may define different support & confidence values. Typically, the user may start with high support & confidence values to explore general relations (that appear often) and then, subsequently decrease the values to explore more specific

[5] see [19] for an overview about the extensive research by the statistics community on the derivation of concept hierarchies

relations. The extracted relations are presented to the user using the result set depicted in Figure 4. The naming of the relations using lexical entries $\mathcal{L}^{\mathcal{R}}$ is currently done manually by the user.

Pattern-based Approaches. Pattern-based approaches are heuristic methods based on regular expressions (see [10] for a related approach to taxonomy extraction using patterns). Patterns work very well in our setting due to the fact that the output of the natural language component is regular (see [28] for in-depth coverage). Patterns can be used to acquire taxonomic ($\mathcal{H}^{\mathcal{C}}$) as well as non-taxonomic relations. The pattern approach proved to be extremely valuable in the aforementioned case study [28], where a corporate thesaurus proved to be a valuable knowledge source. Other patterns deal with compounds, that are very frequent in the German language. Take for example "Arbeitslosenentschaedigung" (the German word for "unemployment benefits"): the defined pattern will find out, that "unemployment benefits" are a special kind of "benefits". The drawback of pattern-based approaches is of course the need to define the patterns, which is a time consuming but often very valuable task.

4.2 Ontology Maintenance

Ontologies applied in real-world settings are evolving. In our framework we regard ontology maintenance as being composed of two subtasks: One is **ontology pruning**, which refers to the process of removing elements from the ontology that are no more relevant to a given application domain. Secondly **ontology refinement**, which focuses on learning the meaning of unknown words, viz. the recognition of important words that are not reflected in the ontology.

Ontology Pruning For example, pruning is of need, if one adopts a (generic) ontology to a given domain. Again, we take the assumption that the occurrence of specific concepts and conceptual relations in web documents are vital for the decision whether or not a given concept or relation should remain in an ontology. We take a frequency based approach determining concept frequencies in a corpus ([24]). Entities that are frequent in a given corpus are considered as a constituent of a given domain. But - in contrast to ontology extraction - the mere frequency of ontological entities is not sufficient.

To determine domain relevance ontological entities retrieved from a domain corpus are compared to frequencies obtained from a generic corpus. The user can select several relevance measures for frequency computation. The ontology pruning algorithm uses the computed frequencies to determine the relative relevancy of each concept contained in the ontology. All existing concepts and relations, which are more frequent[6] in the domain-specific corpus remain in the ontology. The user can also control the pruning of concepts which are neither contained in the domain-specific nor in the generic corpus.

Ontology Refinement. An important aspect of ontology maintenance is the incremental extension of an ontology with concepts $\mathcal{C}$ and associated lexical entries $\mathcal{L}^{\mathcal{C}}$. The reader may note that, due to incremental approach taken for ontology extraction, all extraction algorithms can also used be used for ontology refinement. However, these algorithms can not align unknown words with given concepts, nor they can not detect synonyms that refer to the same concept.

Our refinement approach is based on the following assumption: Unknown words might share a similar conceptual behavior as known words[7]. A small example may illustrate the idea. For example looking at the unknown lexical entry "weekend excursion", one may say that it shares the same conceptual behaviour as the concept EXCURSION. Sharing conceptual behaviour may be computed using measures of distributional similarity (e.g., cosine measure, kullback leibler divergence, etc.) based on concept vectors. The data for this learning task is represented in a concept/lexical entry-concept matrix as given in table 1[8].

[6] by a user supplied factor

[7] We are talking about a conceptual behavior as already known words are already mapped to concepts)

[8] The counts for dependency between a lexical entry or concept and a concept entered in the table are computed using the different linguistic and heuristic indicators provided by our natural language processing system.

ID	ACCOMODATION	HOTEL	EVENT
EXCURSION	5	4	2
"weekend excursion"	4	4	1
CAR	1	1	1

Table 1: Example matrix

5 Evaluation

In general, the evaluation of ontology learning is a challenging task. No standard evaluation measures, like precision and recall (in the information retrieval community) or accuracy (in the machine learning community), are available to estimate the quality of ontology learning techniques. The unsupervised nature of ontology learning techniques makes evaluation even more difficult than supervised learning, e.g. such as classification.

Therefore we propose to use an implicit evaluation approach: We compute the similarity between a manually-built reference ontology, and an ontology that has been generated by applying a particular ontology learning technique. It is assumed that a high similarity between the hand-modeled ontology and the ontology learning-based acquired ontology indicates a successful application of a particular ontology learning technique. This approach is based on the well-known idea of having a so-called gold standard as a reference. We have developed a number of measures to compare the similarity between two given ontologies. The reader may note, that the involvement of manual modelling and the ability to combine different learning techniques complicates an attempt to provide a meaningful evaluation tremendously. We introduce two straight forward evaluation measures, that are derived from information retrieval's precision and recall measures.

Precision and Recall adopted for Ontology Learning. The first measures that are considered are *precision* and *recall* as typically used in information retrieval. The general setting is that for many problems a set of targets is given (in the ontology learning task, e.g. $\mathcal{L}^{\mathcal{C}}, \mathcal{C}, \mathcal{H}^{\mathcal{C}}, \mathcal{R}$) contained within a larger collection (e.g. all possible combinations of a given set of elements). The algorithm or the system then decides on a specific selected set of elements. Based on this counts of the number of items given in the classical definition, a definition of precision and recall for ontology learning is adopted. Precision is a measure of the proportion of selected items the system got right:

$$\text{precision} = \frac{tp}{tp + fp} \tag{1}$$

Adopted to the ontology learning task precision is given as follows:

$$\text{precision}_{OL} = \frac{|Comp \cap Ref|}{|Comp|} \tag{2}$$

The set Ref is the set of elements that are given in the reference ontology. $Comp$ is the set of elements that are contained in the comparison ontology. Elements in this sense are primitives according to the ontology structure $\mathcal{O}$, e.g. like the concept lexical entries $\mathcal{L}^{\mathcal{C}}$, concepts $\mathcal{C}$, the concept hierarchy or non-taxonomic relations between concepts. Recall is a measure of the propertion of the target items that the system selected:

$$\text{recall} = \frac{tp}{tp + fn} \tag{3}$$

Recall adopted to the ontology learning task is given as follows:

$$\text{recall}_{OL} = \frac{|Comp \cap Ref|}{|Ref|} \tag{4}$$

Figure 6 depicts the results obtained by comparing the automatically extracted set of non-taxonomic relations extracted by using the adopted association rule algorithm (with varying precision / recall thresholds) with a set of human modeled non-taxonomic relations (a detailed description of this evaluation is provided in [13]). The reader may note that the well-known trade-off between precision and recall becomes obvious again in this figure.

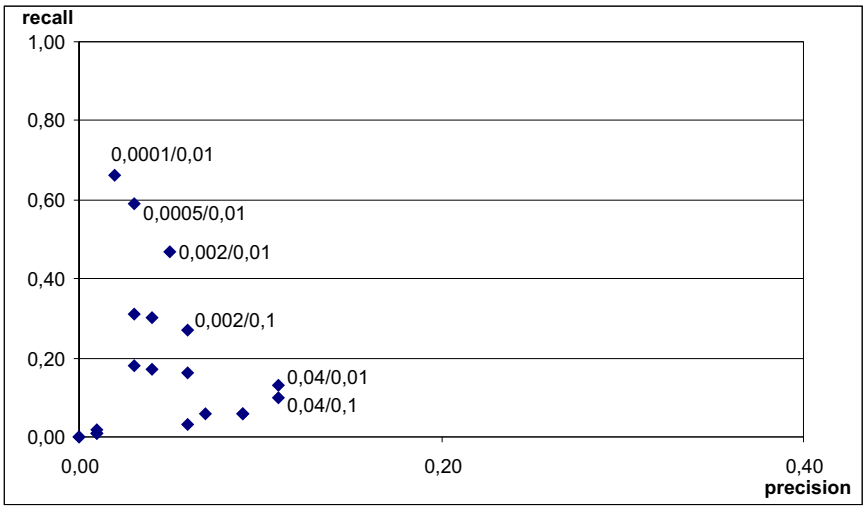

Figure 6: Precision and Recall for Non-Taxonomic Relation Extraction

Precision and recall gave us some hints about how to gauge the thresholds for the algorithms. Nevertheless, these measures lack a sense for the sliding scale of adequacy prevalent in the hierarchical target structures. Thus, more complex similarity measures that include the conceptual structures (with specific focus on the taxonomy) given in the ontology have been developed, which can not presented here due to lack of space, although a more comprehensive approach to evaluating a particular ontology learning algorithm using these measures has been described in [14]. The evaluation of each particular learning algorithm has been presented before (see [17, 11, 14, 15]).

6 Related Work

There are only a few approaches that describe the development of frameworks and workbenches for extracting ontologies from textual data. This section gives a short survey on systems and approaches that deal with supporting the ontology engineer by analyzing domain-specific documents.

Indeed, a number of proposals have been made to facilitate ontological engineering through automatic discovery from domain data, particularly natural language texts (e.g. [7, 9]). However, an overall framework for ontology extraction and maintenance from text does not exist. Faure & Nedellec [6] present a cooperative machine learning system, ASIUM, which acquires taxonomic relations and subcategorization frames of verbs based on syntactic input. The ASIUM system hierarchically clusters nouns based on the verbs that they are syntactically related with and *vice versa*. Thus, they cooperatively extend the lexicon, the set of concepts, and the concept hierarchy $(\mathcal{L}, \mathcal{C}, \mathcal{H}^{\mathcal{C}})$. Hahn and Schnattinger [9] introduce a methodology for the maintenance of domain-specific taxonomies. An ontology is incrementally updated as new concepts are acquired from real-world texts. The acquisition process is centered around linguistic and conceptual "quality" of various forms of evidence underlying the generation and refinement of concept hypotheses. Their ontology learning approach is embedded in a framework for natural language understanding, named Syndicate [8]. With respect to TEXT-TO-ONTO, both approaches fail to treat non-taxonomic conceptual relations and do not include user modelling facilities.

Mikheev & Finch [21] present the KAWB Workbench for "Acquisition of Domain Knowledge form Natural Language". The workbench compromises a set of computational tools for uncovering internal

structure in natural language texts. The main idea behind the workbench is the independence of the text representation and text analysis phases. At the representation phase the text is converted from a sequence of characters to features of interest by means of the annotation tools. At the analysis phase those features are used by statistics gathering and inference tools for finding significant correlations in the texts. The analysis tools are independent of particular assumptions about the nature of the feature-set and work on the abstract level of feature elements represented as SGML items. The KAWB Workbench is not targeted at ontologies.

In contrast to the tools described above, the TEXT-TO-ONTO system defines a common framework into which extraction and maintenance mechanisms may be easily plugged-in. In addition we provide a tight integration to a manual engineering system allowing semi-automatic bootstrapping of a domain ontology.

7 Conclusion

In this paper we have introduced the TEXT-TO-ONTO system for ontology extraction and maintenance and its underlying principles and mechanisms.

How to obtain a knowledge representation that is apt for both human and machine agents is one further challenge to ontology engineering. On the one hand the deployment of ontologies in computer applications requires a representation that is suitable for software. At the same time it is necessary to provide a human readable representation. This makes knowledge representation a challenging task. As a side effect our approach also addresses this challenge, due to the fact that the applied techniques must operate on human readable input, all ontological entities are referenced by identifiers in human language. These identifiers are reused to provide a representation apt for human users.

TEXT-TO-ONTO has been implemented as a stand-alone system. However, many functionalities (e.g. ontology services) could be dedicated to a server. We are currently working on the development of a comprehensive server architecture, KAON[9], that can be used with clients like TEXT-TO-ONTO. As mentioned earlier, we have not further detailed the ontology structure definition in this paper with respect to axioms and instances such as given in [18]. In the future we will further research how axioms (or rules), like the information that the COOPERATESWITH relation is symmetric, may be automatically derived by applying ontology learning techniques. The learning and extraction of instances has been done in several other research communities, such as the information extraction community or the wrapper generation community. The work on semi-automatically generation of ontology-based instances from web pages has been described in [5] and will be pursued in the future with a tight integration to ontology learning.

References

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

[2] S. Decker, D. Brickley, J. Saarela, and J. Angele. A Query and Inference Service for RDF. In *Proceedings of the W3C Query Language Workshop (QL-98), http://www.w3.org/TandS/QL/QL98/, Boston, MA, December 3-4*, 1998.

[3] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al., editors, *Database Semantics: Semantic Issues in Multimedia Systems*, pages 351–369. Kluwer Academic Publisher, 1999.

[4] S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. Klein, and J. Broekstra. Knowledge representation on the web. In *Proceedings of the 2000 International Workshop on Description Logics (DL2000), Aachen, Germany*, 2000.

[5] M. Erdmann, A. Maedche, H.-P. Schnurr, and Steffen Staab. 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, August 2000.

[6] D. Faure and C. Nedellec. A corpus-based conceptual clustering method for verb frames and ontology acquisition. In *LREC workshop on adapting lexical and corpus resources to sublanguages and applications*, Granada, Spain, 1998.

[9] http://kaon.aifb.uni-karlsruhe.de

[7] D. Faure and T. Poibeau. First experiments of using semantic knowledge learned by asium for information extraction task using intex. In *Proceedings of the ECAI'2000 Workshop Ontology Learning*, 2000.

[8] U. Hahn and M. Romacker. Content management in the syndikate system — how technical documents are automatically transformed to text knowledge bases. *Data & Knowledge Engineering*, 35:137–159, 2000.

[9] U. Hahn and K. Schnattinger. Towards text knowledge engineering. In *Proc. of AAAI '98*, pages 129–144, 1998.

[10] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics. Nantes, France*, 1992.

[11] J.-U. Kietz, A. Maedche, and R. Volz. Semi-automatic ontology acquisition from a corporate intranet. In *International Conference on Grammar Inference (ICGI-2000)*, to appear: Lecture Notes in Artificial Intelligence, LNAI, 2000.

[12] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42:741–843, 1995.

[13] A. Maedche. *Ontology Learning for the Semantic Web*. to appear in: Kluwer, Dordrecht, Boston, London, 2002.

[14] A. Maedche and S. Staab. Discovering conceptual relations from text. In *Proceedings of ECAI-2000*. IOS Press, Amsterdam, 2000.

[15] A. Maedche and S. Staab. Mining Ontologies from Text. In *Proceedings of EKAW-2000, Springer Lecture Notes in Artificial Intelligence (LNAI-1937), Juan-Les-Pins, France, 2000*. Springer, 2000.

[16] A. Maedche and S. Staab. Semi-automatic engineering of ontologies from text. In *Proceedings of the 12th Internal Conference on Software and Knowledge Engineering. Chicago, USA, July, 5-7, 2000*. KSI, 2000.

[17] A. Maedche and S. Staab. Semi-automatic engineering of ontologies from text. In *Proceedings of the 12th Internal Conference on Software and Knowledge Engineering. Chicago, USA, July, 5-7, 2000*. KSI, 2000.

[18] A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. *SEmantic PortAL - The SEAL approach*. to appear: In Creating the Semantic Web. D. Fensel, J. Hendler, H. Lieberman, W. Wahlster (eds.) MIT Press, MA, Cambridge, 2001.

[19] C.D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.

[20] R. Michalski and K. Kaufmann. Data mining and knowledge discovery: A review of issues and multistrategy approach. In *Machine Learning and Data Mining Methods and Applications*. John Wiley, England, 1998.

[21] A. Mikheev and S. Finch. A workbench for finding structure in text. In *In Proceedings of the 5th Conference on Applied Natural Language Processing — ANLP'97, March 1997, Washington DC, USA*, pages 372–379, 1997.

[22] K. Morik. Balanced cooperative modeling. *Machine Learning*, 11:217–235, 1993.

[23] G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun. An information extraction core system for real world german text processing. In *ANLP'97 — Proceedings of the Conference on Applied Natural Language Processing*, pages 208–215, Washington, USA, 1997.

[24] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1988.

[25] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of VLDB '95*, pages 407–419, 1995.

[26] S. Staab, M. Erdmann, and A. Maedche. Engineering Ontologies using Semantic Patterns. In *Proceedings of the IJCAI-2001 Workshop on E-Business & Intelligent Web. Seattle, August 03, 2001*, 2001.

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

[28] Raphael Volz. Akquisition von ontologien mit text-mining-verfahren. Technical Report 27 - ISSN 1424-4691, Rentenanstalt / Swiss Life, IT Research and Development, CC/ITRD, CH-8022 Zrich, Switzerland, November 2001.