Ontology Learning for the Semantic Web

Alexander Maedche and Steffen Staab
Institute AIFB, D-76128 Karlsruhe, Germany
http://www.aifb.uni-karlsruhe.de/WBS
and
Ontoprise GmbH, Haid-und-Neu-Strasse 7, 76131 Karlsruhe, Germany
http://www.ontoprise.com

Word Count: 5541

Abstract

The Semantic Web relies heavily on the formal ontologies that structure underlying data for the purpose of comprehensive and transportable machine understanding. Therefore, the success of the Semantic Web depends strongly on the proliferation of ontologies, which requires fast and easy engineering of ontologies and avoidance of a knowledge acquisition bottleneck.

Ontology Learning greatly facilitates the construction of ontologies by the ontology engineer. The vision of ontology learning that we propose here includes a number of complementary disciplines that feed on different types of unstructured, semi-structured and fully structured data in order to support a semi-automatic, cooperative ontology engineering process. Our ontology learning framework proceeds through ontology import, extraction, pruning, refinement, and evaluation giving the ontology engineer a wealth of coordinated tools for ontology modeling. Besides of the general framework and architecture, we show in this paper some exemplary techniques in the ontology learning cycle that we have implemented in our ontology learning environment, *Text-To-Onto*, such as ontology learning from free text, from dictionaries, or from legacy ontologies, and refer to some others that need to complement the complete architecture, such as reverse engineering of ontologies from database schemata or learning from XML documents.

Ontologies for the Semantic Web

Conceptual structures that define an underlying *ontology* are germane to the idea of machine processable data on the Semantic Web. Ontologies are (meta)data schemas, providing a controlled vocabulary of concepts, each with an explicitly defined and machine processable semantics. By defining shared and common domain theories, ontologies help both people and machines to communicate concisely,

supporting the exchange of semantics and not only syntax. Hence, the cheap and fast construction of domain-specific ontologies is crucial for the success and the proliferation of the Semantic Web.

Though ontology engineering tools have become mature over the last decade (cf. [2]), the manual acquisition of ontologies still remains a tedious, cumbersome task resulting easily in a *knowledge acquisition bottleneck*. Having developed our ontology engineering workbench, *OntoEdit*, we had to face exactly this issue, in particular we were given questions like

- Can you develop an ontology fast? (time)
- Is it difficult to build an ontology? (difficulty)
- How do you know that you've got the ontology right? (confidence)

In fact, these problems on time, difficulty and confidence that we ended up with were similar to what knowledge engineers had dealt with over the last two decades when they elaborated on methodologies for knowledge acquisition or workbenches for defining knowledge bases. A method that proved extremely beneficial for the knowledge acquisition task was the integration of knowledge acquisition with machine learning techniques [12]. The drawback of these approaches, e.g. the work described in [6], however, was their rather strong focus on structured knowledge or data bases, from which they induced their rules.

In contrast, in the Web environment that we encounter when building Web ontologies, the structured knowledge or data base is rather the exception than the norm. Hence, intelligent means for an ontology engineer takes on a different meaning than the — very seminal — integration architectures for more conventional knowledge acquisition [1].

Our notion of *Ontology Learning* aims at the integration of a multitude of disciplines in order to facilitate the construction of ontologies, in particular machine learning. Because the fully automatic acquisition of knowledge by machines remains in the distant future, we consider the process of ontology learning as semi-automatic with human intervention, adopting the paradigm of *balanced cooperative modeling* [5] for the construction of ontologies for the Semantic Web. This objective in mind, we have built an architecture that combines knowledge acquisition with machine learning, feeding on the resources that we nowadays find on the syntactic Web, *viz.* free text, semi-structured text, schema definitions (DTDs), etc. Thereby, modules in our framework serve different steps in the engineering cycle, which here consists of the following five steps (cf. Figure 1):

First, existing ontologies are **imported** and **reused** by merging existing structures or defining mapping rules between existing structures and the ontology to be established. For instance, [9] describe how ontological structures contained in Cyc are used in order to facilitate the construction of a domain-specific ontology. Second, in the ontology **extraction** phase major parts of the target ontology are modeled with learning support feeding from web documents. Third, this rough outline of the target ontology needs to be **pruned** in order to better adjust the ontology to its prime purpose. Fourth, ontology **refinement** profits from the given domain ontology, but completes the ontology at a fine granularity (also in contrast to extraction). Fifth, the prime target application serves as a measure for validating the resulting ontology [11]. Finally, one may revolve again in this cycle, e.g. for including new domains into the constructed ontology or for maintaining and updating its scope.

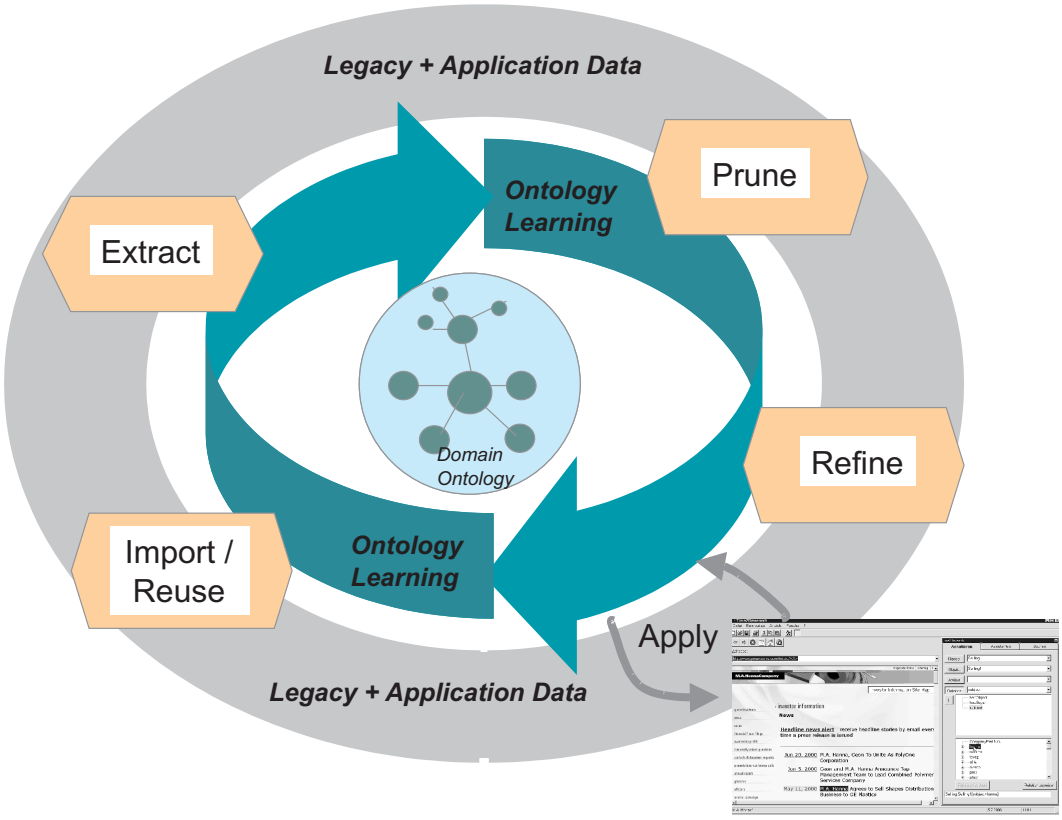

Figure 1: Ontology Learning process steps

An Architecture for Ontology Learning

Given the task of constructing and maintaining an ontology for a Semantic Web application, e.g. for an ontology-based knowledge portal that we have been dealing with (cf. [10]), we have produced a wish list of what kind of support we would fancy.

Ontology Engineering Workbench *OntoEdit*. As core to our approach we have built a graphical user interface to support the ontology engineering process manually performed by the ontology engineer. Here, we offer sophisticated graphical means for manual modeling and refining the final ontology. Different views are offered to the user targeting the epistemological level rather than a particular representation language. However, the ontological structures built there may be exported to standard Semantic Web representation languages, such as OIL and DAML-ONT, as well as our own F-Logic based extensions of RDF(S). In addition, executable representations for constraint checking and application debugging can be generated and then accessed via SilRi[1], our F-Logic inference engine, that is directly connected with *OntoEdit*.

The sophisticated ontology engineering tools we knew, e.g. the *Protégé* modeling environment for knowledge-based systems [2], would offer capabilities roughly comparable to *OntoEdit*. However, given the task of constructing a knowledge portal, we found that there was this large conceptual bridge between the ontology engineering tool and the input (often legacy data), such as Web documents, Web document schemata, databases on the Web, and Web ontologies, which ultimately determined the target ontology. Into this void we have positioned new components of our ontology learning architecture (cf. Figure 2). The new components support the ontology engineer in importing existing ontology primitives, extracting new ones, pruning given ones, or refining with additional ontology primitives. In our case, the ontology primitives comprise:

- a set of strings that describe lexical entries $\mathcal{L}$ for concepts and relations;
- a set of concepts[2] — $\mathcal{C}$;
- a taxonomy of concepts with multiple inheritance (heterarchy) $\mathcal{H}_\mathcal{C}$;
- a set of non-taxonomic relations — $\mathcal{R}$ — described by their domain and range restrictions;

[1] `http://www.ontoprise.com/` — then `download area`.

[2] Concepts in our framework are roughly akin to synsets in WordNet [4].

- a heterarchy of relations, i.e. a set of taxonomic relations $\mathcal{H}_{\mathcal{R}}$;
- relations $\mathcal{F}$ and $\mathcal{G}$ that relate concepts and relations with their lexical entries, respectively; and, finally,
- a set of axioms $\mathcal{A}$ that describe additional constraints on the ontology and allow to make implicit facts explicit [10].

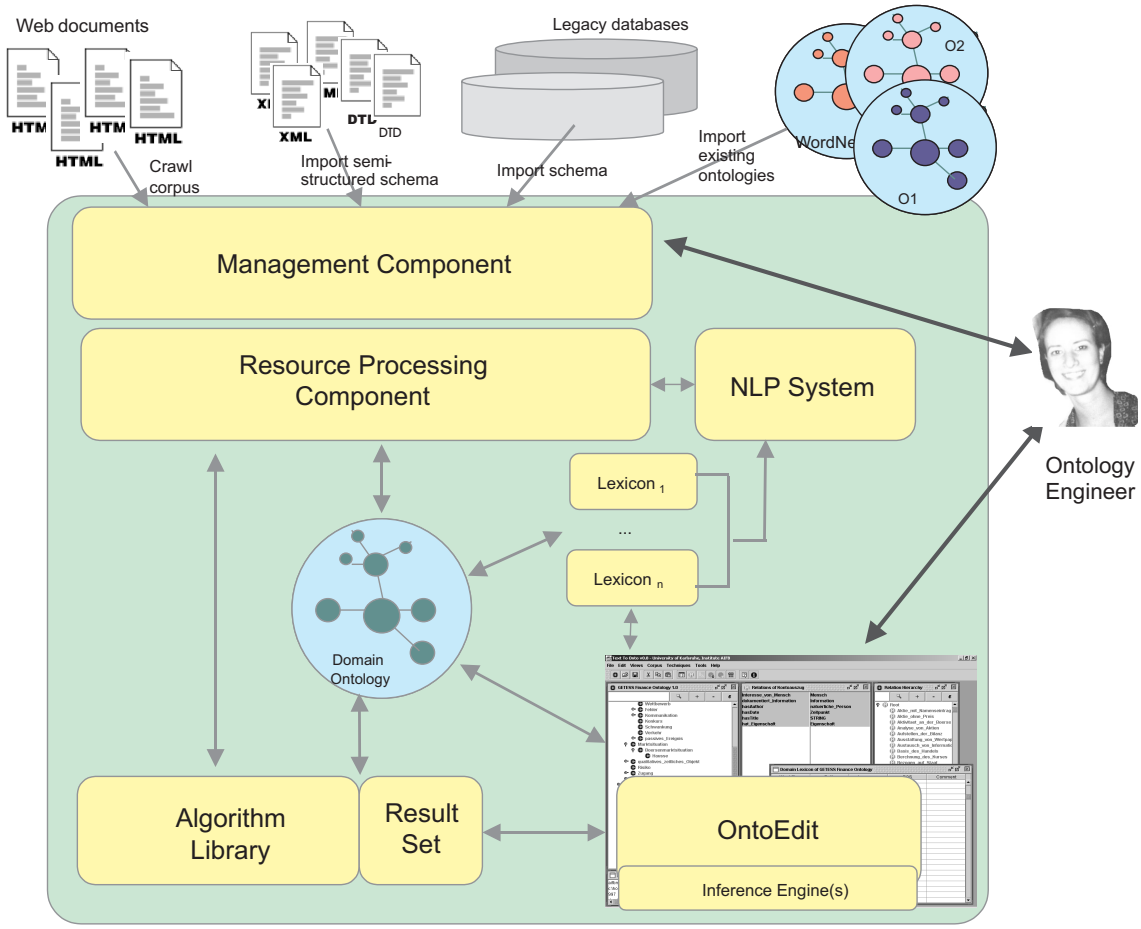

Figure 2: Architecture for Learning Ontologies for the Semantic Web

This structure corresponds closely to RDFS, the one exception is the explicit consideration of lexical entries. The separation of concept reference and concept denotation, which may be easily expressed in RDF, allows to provide very domain-specific ontologies without incurring an instantaneous conflict when merging ontologies — a standard request in the Semantic Web. For instance, the lexical entry “school” in one ontology may refer to a building in ontology A, but to an organization in ontol-

ogy B, or to both in ontology C. Also in ontology A the concept refered to in English by “school” and “school building” may be referred to in German by “Schule” and “Schulgebäude”.

Ontology learning relies on ontology structures given along these lines and on input data as described above in order to propose new knowledge about reasonably interesting concepts, relations, lexical entries, or about links between these entities — proposing the addition, the deletion, or the merging of some of them. The results of the ontology learning process are presented to the ontology engineer by the graphical result set representation (cf. Figure 4 for an example of how extracted properties may be presented). The ontology engineer may then browse the results and decide to follow, delete, or modify the proposals in accordance to the purpose of her task.

Components for Learning Ontologies

Integrating the considerations from above into a coherent generic architecture for extracting and maintaining ontologies from data on the Web we have identified several core components. There are, *(i)*, a generic management component dealing with delegation of tasks and constituting the infrastructure backbone, *(ii)*, a resource processing component working on input data from the Web including, in particular, a natural language processing system, *(iii)*, an algorithm library working on the output of the resource processing component as well as the ontology structures sketched above and returning result sets also mentioned above and, *(iv)*, the graphical user interface for ontology engineering, *OntoEdit*.

Management component. The ontology engineer uses the management component to select input data, i.e. relevant resources such as HTML & XML documents, document type definitions, databases, or existing ontologies that are exploited in the further discovery process. Secondly, using the management component, the ontology engineer also chooses among a set of resource processing methods available at the resource processing component and among a set of algorithms available in the algorithm library.

Furthermore, the management component even supports the ontology engineer in discovering task-relevant legacy data, e.g. an ontology-based crawler gathers HTML documents that are relevant to a given core ontology and an RDF crawler follows URIs (i.e., unique identifiers in XML/RDF) that are also URLs in order to cover parts of the so far tiny, but growing Semantic Web.

Resource processing component. Resource processing strategies differ depending on the type of input data made available:

- HTML documents may be indexed and reduced to free text.
- Semi-structured documents, like dictionaries, may be transformed into a predefined relational structure.
- Semi-structured and structured schema data (like DTD's, structured database schemata, and existing ontologies) are handeled following different strategies for import as described later in this paper.
- For processing free natural text our system accesses the natural language processing system SMES (Saarbrücken Message Extraction System), a shallow text processor for German (cf. [7]). SMES comprises a *tokenizer* based on regular expressions, a *lexical analysis* component including various word *lexicons*, a *morphological analysis* module, a *named entity recognizer*, a *part-of-speech tagger* and a *chunk parser*.

After first preprocessing according to one of these or similar strategies, the resource processing module transforms the data into an algorithm-specific relational representation.

Algorithm Library. As described above an ontology may be described by a number of sets of concepts, relations, lexical entries, and links between these entities. An existing ontology definition (including $\mathcal{L}, \mathcal{C}, \mathcal{H}_\mathcal{C}, \mathcal{R}, \mathcal{H}_\mathcal{R}, \mathcal{A}, \mathcal{F}, \mathcal{G}$) may be acquired using various algorithms working on this definition and the preprocessed input data. While specific algorithms may greatly vary from one type of input to the next, there is also considerable overlap concerning underlying learning approaches like association rules, formal concept analysis, or clustering. Hence, we may reuse algorithms from the library for acquiring different parts of the ontology definition.

Subsequently, we introduce some of these algorithms available in our implementation. In general, we use a multi-strategy learning and result combination approach, i.e. each algorithm that is plugged into the library generates normalized results that adhere to the ontology structures sketched above and that may be combined into a coherent ontology definition.

Import & Reuse

Given our experiences in medicine, telecommunication, and insurance, we expect that for almost any commercially significant domain there are some kind of domain conceptualizations available. Thus, we need mechanisms and strategies to *import & reuse* domain conceptualizations from existing (schema) structures. Thereby, the conceptualizations may be recovered, e.g., from legacy database schemata, document-type definitions (DTDs), or from existing ontologies that conceptualize some relevant part of the target ontology.

In the first part of the *import & reuse* step, the schema structures are identified and their general content need to be discussed with domain experts. Each of these knowledge sources must be imported separately. Import may be performed manually — which may include the manual definition of transformation rules. Alternatively, reverse engineering tools, such as exist for recovering extended entity-relationship diagrams from the SQL description of a given database (cf. reference [19, 11] in survey, Table 1), may facilitate the recovery of conceptual structures.

In the second part of the *import & reuse* step, imported conceptual structures need to be merged or aligned in order to constitute a single common ground from which to take-off into the subsequent ontology learning phases of *extracting*, *pruning* and *refining*. While the general research issue concerning merging and aligning is still an open problem, recent proposals (e.g., [8]) have shown how to improve the manual process of merging/aligning. Existing methods for merging/aligning mostly rely on matching heuristics for proposing the merge of concepts and similar knowledge-base operations. Our current research also integrates mechanisms that use a application data oriented, bottom-up approach. For instance, formal concept analysis allows to discover patterns between application data on the one hand and the usage of concepts and relations and the semantics given by their heterarchies on the other hand in a formally concise way (cf. reference [7] in survey, Table 1, on formal concept analysis).

Overall, the import and reuse step in ontology learning seems to be the one that is the hardest to generalize. The task may remind vaguely of the general problems with data warehousing adding, however, challenging problems of its own.

Extracting Ontologies

In the ontology extraction phase of the ontology learning process, major parts, i.e. the complete ontology or large chunks reflecting a new subdomain of the ontology, are modeled with learning support exploiting various types of (Web) sources. Thereby, ontology learning techniques partially rely on given ontology parts. Thus, we here encounter an iterative model where previous revisions through the ontology learning cycle may propel subsequent ones and more sophisticated algorithms may work on structures proposed by more straightforward ones before.

Describing this phase, we sketch some of the techniques and algorithms that have been embedded in our framework and implemented in our ontology learning environment *Text-To-Onto* (cf. Figure 3). Doing so, we cover a very substantial part of the overall ontology learning task in the extraction phase. *Text-To-Onto* proposes many different ontology components, which we have described above (i.e. $\mathcal{L}, \mathcal{C}, \mathcal{R}, \ldots$), to the ontology engineer feeding on several types of input.

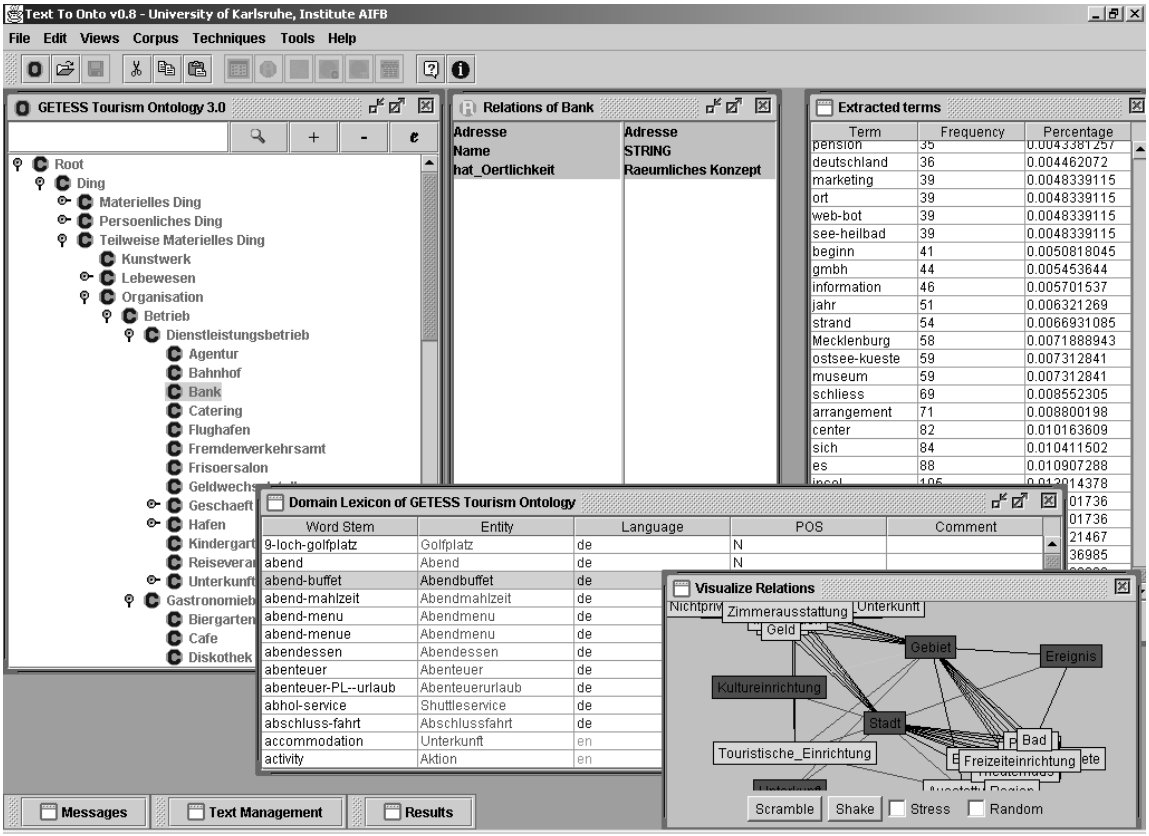

Figure 3: Screenshot of our Ontology Learning Workbench *Text-To-Onto*

Lexical Entry & Concept Extraction. This technique is one of the baseline methods applied in our framework for acquiring lexical entries with corresponding concepts. In *Text-To-Onto*, web documents are morphologically processed, including the treatment of multi-word terms such as "database reverse engineering" by N-grams, a simple statistics means. Based on this text preprocessing, term extraction techniques, which are based on (weighted) statistical frequencies, are applied in order to propose new lexical entries for $\mathcal{L}$.

Often, the ontology engineer follows the proposal by the lexical entry & concept extraction mechanism and includes a new lexical entry in the ontology. Because the new lexical entry comes without an associated concept, the ontology engineer must then decide (possibly with help from further processing) whether to introduce a new concept or link the new lexical entry to an existing concept.

Hierarchical Concept Clustering. Given a lexicon and a set of concepts, one major next step is the taxonomic classification of concepts. One generally applicable method with to this regard is hierarchical clustering. Hierarchical clustering exploits the similarity of items in order to propose a hierarchy of item categories. The similarity measure is defined on the properties of items.

Given the task of extracting a hierarchy from natural language text, adjacency of terms or syntactical relationships between terms are two properties that yield considerable descriptive power to induce the semantic hierarchy of concepts related to these terms.

A sophisticated example for hierarchical clustering is given by Faure & Nedellec (cf. reference [6] in survey, Table 1): They 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 heterarchy $(\mathcal{L}, \mathcal{C}, \mathcal{H}_{\mathcal{C}})$.

Dictionary Parsing. Machine-readable dictionaries (MRD) are frequently available for many domains. Though their internal structure is free text to a large extent, there are comparatively few patterns that are used to give text definitions. Hence, MRDs exhibit a large degree of regularity that may be exploited for extracting a domain conceptualization and proposing it to the ontology engineer.

Text-To-Onto has been used to generate a taxonomy of concepts from a machine-readable dictionary of an insurance company (cf. reference [13] in survey, Table 1). Likewise to term extraction from free text morphological processing is applied, this time however complementing several pattern-

matching heuristics. For example the dictionary contained the following entry:

> ***Automatic Debit Transfer:*** *Electronic service arising from a debit authorization of the Yellow Account holder for a recipient to debit bills that fall due direct from the account..*

Several heuristics were applied to the morphologically analyzed definitions. For instance, one simple heuristic relates the definition term, here "automatic debit transfer", with the first noun phrase occurring in the definition, here "electronic service". Their corresponding concepts are linked in the heterarchy $\mathcal{H}_\mathcal{C}$: $\mathcal{H}_\mathcal{C}$(AUTOMATIC DEBIT TRANSFER, ELECTRONIC SERVICE). Applying this heuristic iteratively, one may propose large parts of the target ontology, more precisely $\mathcal{L}, \mathcal{C}$ and $\mathcal{H}_\mathcal{C}$ to the ontology engineer. In fact, because verbs tend to be modeled as relations, $\mathcal{R}$ (and the linkage between $\mathcal{R}$ and $\mathcal{L}$) may be extended by this way, too.

Association Rules. Association rule learning algorithms are typically used for prototypical applications of data mining, like finding associations that occur between items, e.g. supermarket products, in a set of transactions, e.g. customers' purchases. The generalized association rule learning algorithm extends its baseline by aiming at descriptions at the appropriate level of the taxonomy, e.g. "snacks are purchased together with drinks" rather than "chips are purchased with beer" and "peanuts are purchased with soda".

In *Text-To-Onto* (cf. reference [14] in survey, Table 1) we use a modification of the generalized association rule learning algorithm for discovering properties between classes. A given class hierarchy $\mathcal{H}_\mathcal{C}$ serves as background knowledge. Pairs of syntactically related classes (e.g. pair(FESTIVAL,ISLAND) describing the head-modifier relationship contained in the sentence "The festival on Usedom[3] attracts tourists from all over the world.") are given as input to the algorithm. The algorithm generates association rules comparing the relevance of different rules while climbing up and/or down the taxonomy. The appearingly most relevant binary rules are proposed to the ontology engineer for modeling relations into the ontology, thus extending $\mathcal{R}$.

As the number of generated rules is typically high, we offer various modes of interaction. For example, it is possible to restrict the number of suggested relations by defining so-called restriction classes that have to participate in the relations that are extracted. Another way of focusing is the flexible enabling / disabling of the use of taxonomic knowledge for extracting relations.

[3] Usedom is an island located in north-east of Germany in the Baltic Sea.

Results are presented offering various views onto the results as depicted in Figure 4. A generalized relation that may be induced by the partially given example data above may be the PROPERTY(EVENT,AREA), which may be named by the ontology engineer as LOCATEDIN, *viz.* EVENTS are located in an AREA (thus extending $\mathcal{L}$ and F). The user may add the extracted relations to the ontology by drag-and-drop. To explore and determine the right aggregation level of adding a relation to the ontology, the user may browse the hierarchy view on extracted properties as given in the left part of Figure 4. This view may also support the ontology engineer in defining appropriate SUBPROPERTYOF relations between properties, such as SUBPROPERTYOF(HASDOUBLEROOM,HASROOM) (thereby extending $\mathcal{H}_{\mathcal{R}}$).

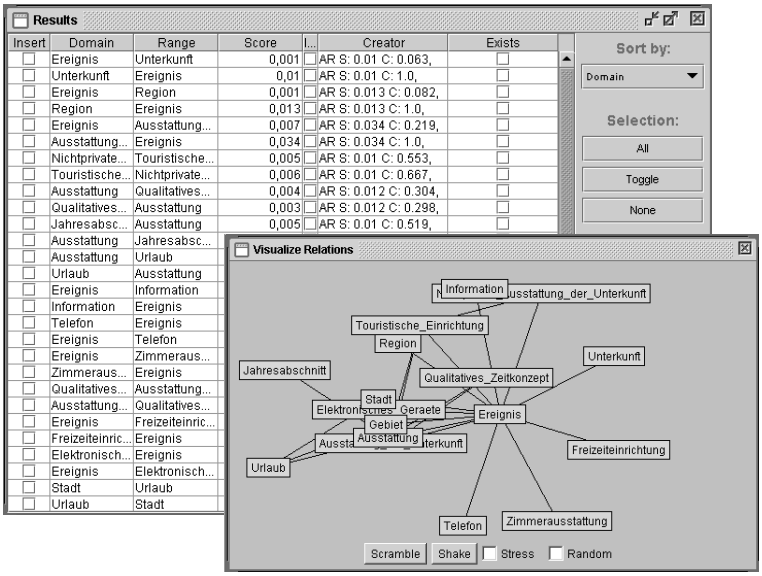

Figure 4: Result Presentation in *Text-To-Onto*

Pruning the Ontology

A common theme of modeling in various disciplines is the balance between completeness and scarcity of the domain model. It is a widely held belief that targeting completeness for the domain model on the one hand appears to be practically inmanagable and computationally intractable, and targeting the scarcest model on the other hand is overly limiting with regard to expressiveness. Hence, what we strive for is the balance between these two, which is really working. We aim at a model that captures a rich conceptualization of the target domain, but that excludes parts that are out of its focus. The *import & reuse* of ontologies as well as the *extraction* of ontologies considerably pull the lever of

the scale into the imbalance where out-of-focus concepts reign. Therefore, we pursue the appropriate diminishing of the ontology in the *pruning* phase.

There are at least two dimensions to look at the problem of pruning. First, one needs to clarify how the pruning of particular parts of the ontology (e.g., the removal of a concept or a relation) affects the rest. For instance, Peterson *et. al.* [9] have described strategies that leave the user with a coherent ontology (i.e. no dangling or broken links). Second, one may consider strategies for proposing ontology items that should be either kept or pruned. We have investigated several mechanisms for generating proposals from application data Given a set of application-specific documents there are several strategies for pruning the ontology. They are based on absolute or relative counts of frequency of terms (cf. reference [13] in survey, Table 1).

Refining the Ontology

Refining plays a similar role as *extracting*. Their difference exists rather on a sliding scale than by a clear-cut distinction. While extracting serves mostly for cooperative modeling of the overall ontology (or at least of very significant chunks of it), the refinement phase is about fine tuning the target ontology and the support of its evolving nature. The refinement phase may use data that comes from the concrete Semantic Web application, e.g. log files of user queries or generic user data. Adapting and refining the ontology with respect to user requirements plays a major role for the acceptance of the application and its further development.

In principle, the same algorithms may be used for extraction as for refinement. However, during refinement one must consider in detail the existing ontology and the existing connections into the ontology, while extraction works more often than not practically from scratch.

A prototypical approach for refinement (though not for extraction!) has been presented by Hahn & Schnattinger (cf. reference [8] in survey, Table 1). They have introduced a methodology for automating the maintenance of domain-specific taxonomies. An ontology is *incrementally* updated as new concepts are acquired from text. The acquisition process is centered around the linguistic and conceptual "quality" of various forms of evidence underlying the generation and refinement of concept hypothesis. In particular they consider semantic conflicts and analogous semantic structures from the knowledge base into the ontology in order to determine the quality of a particular proposal. Thus, they extend an existing ontology with new lexical entries for $\mathcal{L}$, new concepts for $\mathcal{C}$ and new relations

for $\mathcal{H}_C$.

Challenges

Ontology Learning may add significant leverage to the Semantic Web, because it propels the construction of domain ontologies, which are needed fastly and cheaply for the Semantic Web to succeed. We have presented a comprehensive framework for Ontology Learning that crosses the boundaries of single disciplines, touching on a number of challenges. Table 1 gives a survey of what types of techniques should be included in a full-fledged ontology learning and engineering environment. The good news however is that one does not need perfect or optimal support for cooperative modeling of ontologies. At least according to our experience "cheap" methods in an integrated environment may yield tremendous help for the ontology engineer.

While a number of problems remain with the single disciplines, some more challenges come up regarding the particular problem of Ontology Learning for the Semantic Web. First, with the XML-based namespace mechanisms the notion of an ontology with well-defined boundaries, e.g. only definitions that are in one file, will disappear. Rather, the Semantic Web may yield an "amoeba-like" structure regarding ontology boundaries, because ontologies refer to each other and import each other (cf. e.g. the DAML-ONT primitive `import`). However, it is not yet clear how the semantics of these structures will look like. In light of these facts the importance of methods like ontology pruning and crawling of ontologies will drastically increase still. Second, we have so far restricted our attention in ontology learning to the conceptual structures that are (almost) contained in RDF(S) proper. Additional semantic layers on top of RDF (e.g. future OIL or DAML-ONT with axioms, $\mathcal{A}$) will require new means for improved ontology engineering with axioms, too!

Acknowledgements. We thank our students, Dirk Wenke and Raphael Volz, for work at *OntoEdit* and *Text-To-Onto*. Research for this paper was partially financed by Ontoprise GmbH, Karlsruhe, Germany, by US Air Force in the DARPA DAML project "OntoAgents", by European Union in the IST-1999-10132 project "On-To-Knowledge", and by German BMBF in the project "GETESS" (01IN901C0).

Table 1: Survey of Ontology Learning Approaches

Until recently ontology learning *per se*, i.e. for comprehensive construction of ontologies, has not existed. However, much work in a number of disciplines — computational linguistics, information retrieval, machine learning, databases, software engineering — has actually researched and practiced techniques for solving part of the overall problem. Hence, techniques and methods relevant for ontology learning may be found under terms like

- Acquisition of selectional restrictions: Resnik [17] and Basili et al. [2]
- Word sense disambiguation and learning of word senses: Hastings [20]
- Computation of concept lattices from formal contexts: Ganter & Wille [7]
- Reverse Engineering in software engineering: Mueller et al. [16]

Ontology Learning puts a number of research activities, which focus on different types of inputs, but share their target of a common domain conceptualization, into one perspective. One may recognize that these activities are spread between very many communities incurring references from 20 completely different events / journals:

Domain	Method	Features used	Prime purpose	Papers
Free Text	Clustering	Syntax	Extract	Buitelaar [3], Assadi [1] and Faure & Nedellec [6]
	Inductive Logic Programming	Syntax, Logic representation	Extract	Esposito et al. [5]
	Association rules	Syntax, Tokens	Extract	Maedche & Staab [14]
	Frequency-based	Syntax	Prune	Kietz et al. [13]
	Pattern-Matching		Extract	Morin [15]
	Classification	Syntax, Semantics	Refine	Schnattinger & Hahn [8]
Dictionary	Information extraction	Syntax	Extract	Hearst [9], Wilks [21] and Kietz et al. [13]
	Page rank	Tokens		Jannink & Wiederhold [10]
Knowledge base	Concept Induction, A-Box mining	Relations	Extract	Kietz & Morik [12] and Schlobach [18]
Semi-structured schemata	Naive Bayes	Relations	Reverse engineering	Doan et al. [4]
Relational schemata	Data Correlation	Relations	Reverse engineering	Johannesson [11] and Tari et al. [19]

References of Survey

[1] H. Assadi. Construction of a regional ontology from text and its use within a documentary system. In *Proceedings of the International Conference on Formal Ontology and Information Systems (FOIS-98)*, Trento, Italy, 1998. IOS Press, Amsterdam, 1998.

[2] R. Basili, M. T. Pazienza, and P. Velardi. Acquisition of selectional patterns in a sublanguage. *Machine Translation*, 8(1), 1993.

[3] P. Buitelaar. CORELEX*: Systematic Polysemy and Underspecification*. PhD thesis, Brandeis University, Department of Computer Science, 1998.

[4] A. Doan, P. Domingos, and A. Levy. Learning Source Descriptions for Data Integration. In *Proceedings of the International Workshop on The Web and Databases (WebDB-2000)*, Dallas, Texas, 2000, pages 81-86.

[5] F. Esposito and S. Ferilli and N. Fanizzi and G. Semeraro Learning from Parsed Sentences with INTHELEX. In *Proceedings of Learning Language in Logic Workshop (LLL-2000)*, Lisbon, Portugal, 2000.

[6] D. Faure and C. Nedellec. A corpus-based conceptual clustering method for verb frames and ontology acquisition. In *LREC-98 Workshop on Adapting Lexical and Corpus Resources to Sublanguages and Applications*, Granada, Spain, 1998.

[7] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin - Heidelberg - New York, 1999.

[8] U. Hahn and K. Schnattinger. Towards text knowledge engineering. In *Proceedings of AAAI-98*, 1998, pages 524-531.

[9] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING-92*, Nantes, France, 1992.

[10] J. Jannink and G. Wiederhold. Thesaurus entry extraction from an on-line dictionary. In *Proceedings of Fusion-99*, California, USA, 1999.

[11] P. Johannesson. A method for transforming relational schemas into conceptual schemas. In *International Conference on Data Engineering (IDCE-94)*, Houston, 1994. IEEE Press, 1994, pages 190-201.

[12] J.-U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(2), 1994, pages 193-217.

[13] J.-U. Kietz, A. Maedche, and R. Volz. Semi-automatic Ontology Acquisition from a Corporate Intranet. In *Proceedings of Learning Language in Logic Workshop (LLL-2000)*, Lisbon, Portugal, 2000.

[14] A. Maedche and S. Staab. Discovering conceptual relations from text. In *Proceedings of ECAI-00*. IOS Press, Amsterdam, 2000, pages 321-325.

[15] E. Morin. Automatic acquisition of semantic relations between terms from technical corpora. In *Proc. of the Fifth International Congress on Terminology and Knowledge Engineering (TKE-99)*, Innsbruck, Austria, 1999.

[16] H. A. Mueller, J. H. Jahnke, D. B. Smith, M.-A. Storey, S. R. Tilley, and K. Wong. Reverse Engineering: A Roadmap. In *Proceedings the International Conference on Software Engineering (ICSE-00)*, Limerick, Ireland. Springer, 2000, pages 47-60.

[17] P. Resnik. *Selection and Information: A Class-based Approach to Lexical Relationships*. PhD thesis, University of Pennsylania, 1993.

[18] S. Schlobach. Assertional mining in description logics. In *Proceedings of the 2000 International Workshop on Description Logics (DL-2000)*, 2000, pages 237-246. http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-33/.

[19] Z. Tari, O. Bukhres, J. Stokes, and S. Hammoudi. The Reengineering of Relational Databases based on Key and Data Correlations. In *Proceedings of the 7th Conference on Database Semantics (DS-7)*, Leysin, Switzerland. Chapman & Hall, 1998.

[20] P. Wiemer-Hastings, A. Graesser, and K. Wiemer-Hastings. Inferring the meaning of verbs from context. In *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society (CogSci-98)*, Wisconsin, Madison, 1998.

[21] Y. Wilks, B. Slator, and L. Guthrie. *Electric Words: Dictionaries, Computers, and Meanings*. MIT Press, Cambridge, MA, 1996.

References

[1] B. Gaines and M. Shaw. Integrated knowledge acquisition architectures. *Journal of Intelligent Information Systems*, 1(1), 1992, pages 9-34.

[2] E. Grosso, H. Eriksson, R. Fergerson, S. Tu and M. Musen. Knowledge modeling at the millennium — the design and evolution of Protégé-2000. In *Proceedings of KAW-99, Banff, Canada*, 1999.

[3] 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, 2000, pages 137-159.

[4] G. Miller. WordNet: A lexical database for English. *CACM*, 38(11), 1995, pages 39-41.

[5] K. Morik. Balanced cooperative modeling. *Machine Learning*, 11(1), 1993, pages 217-235.

[6] K. Morik, S. Wrobel, J.-U. Kietz, and W. Emde. *Knowledge acquisition and machine learning: Theory, methods, and applications*. Academic Press, London, 1993.

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

[8] N. Fridman Noy and M. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of AAAI-2000, Austin, Texas.* MIT Press/AAAI Press, 2000, pages 450-455.

[9] B. Peterson, W. Andersen, and J. Engel. Knowledge bus: Generating application-focused databases from large ontologies. In *Proceedings of KRDB-98, Seattle, Washington, USA*, 1998, pages 2.1-2.10.

[10] S. Staab, and A. Maedche. Knowledge Portals — Ontologies at work. *AI Magazine*, 21(2), Summer 2001 (to appear).

[11] S. Staab, H.-P. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1), 2001.

[12] G. Webb, J. Wells, and Z. Zheng. An experimental evaluation of integrating machine learning with knowledge acquisition. *Machine Learning*, 35(1), 1999, pages 5-23.

Alexander Maedche. Alexander Maedche is a Ph.D. student at the Institute of Applied Informatics and Formal Description Methods, University of Karlsruhe. In 1999 he has received a diploma in industrial engineering, majoring in computer science and operations research, also from the University of Karlsruhe. His diploma thesis on knowledge discovery earned him a best thesis award at the University of Karlsruhe. Alexander Maedche‘s research interests cover knowledge discovery in data and text, ontology engineering, learning and application of ontologies, and the semantic web. Recently, he has started part-time consulting at Ontoprise GmbH in order to apply research on ontologies in commercial practice. Contact Alex at Institute AIFB, University of Karlsruhe, 76128 Karlsruhe; EMail: ama@aifb.uni-karlsruhe.de

Steffen Staab. Steffen Staab received a M.S.E. from the Univ. of Pennsylvania in 1994, and the Dr. rer. nat. from the University of Freiburg, in 1998 both in informatics. After a stint doing consulting with the Fraunhofer Institute for Industrial Engineering, Stuttgart, he joined the Univ. of Karlsruhe, where he is now an assistant professor. In 1999 he has co-founded Ontoprise GmbH, a company providing a wide range of technology around ontologies. Steffen has been working and publishing on computational linguistics, text mining, knowledge management, ontologies, and the semantic web. He won a best paper award for a paper on constraint reasoning at ECAI-1998 and he is chairing the Semantic Web workshop in Hongkong at WWW10. Contact Steffen at Institute AIFB, University of Karlsruhe, 76128 Karlsruhe; EMail: sst@aifb.uni-karlsruhe.de