Ontology Learning for the Semantic Web

Alexander Maedche and Steffen Staab, *University of Karlsruhe*

The authors present an ontology-learning framework that extends typical ontology engineering environments by using semiautomatic ontology-construction tools. The framework encompasses ontology import, extraction, pruning, refinement, and evaluation.

The Semantic Web relies heavily on formal ontologies to structure data for comprehensive and transportable machine understanding. Thus, the proliferation of ontologies factors largely in the Semantic Web's success. *Ontology learning* greatly helps ontology engineers construct ontologies. The vision of ontology learning that we propose includes a number of complementary disciplines that feed on different types of unstructured, semistructured, and fully structured data to support semiautomatic, cooperative ontology engineering. Our ontology-learning framework proceeds through ontology import, extraction, pruning, refinement, and evaluation, giving the ontology engineer coordinated tools for ontology modeling. Besides the general framework and architecture, this article discusses techniques in the ontology-learning cycle that we implemented in our ontology-learning environment, such as ontology learning from free text, dictionaries, and legacy ontologies. We also refer to other techniques for future implementation, such as reverse engineering of ontologies from database schemata or learning from XML documents.

Ontologies for the Semantic Web

The conceptual structures that define an underlying ontology provide the key to machine-processable data on the Semantic Web. *Ontologies* serve as metadata schemas, providing a controlled vocabulary of concepts, each with explicitly defined and machine-processable semantics. By defining shared and common domain theories, ontologies help people and machines to communicate concisely—supporting semantics exchange, not just syntax. Hence, the Semantic Web's success and proliferation depends on quickly and cheaply constructing domain-specific ontologies.

Although ontology-engineering tools have matured over the last decade,[1] manual ontology acquisition remains a tedious, cumbersome task that can easily result in a knowledge acquisition bottleneck. When developing our ontology-engineering workbench, OntoEdit, we particularly faced this question as we were asked questions that dealt with time ("Can you develop an ontology quickly?"), difficulty, ("Is it difficult to build an ontology?"), and confidence ("How do you know that you've got the ontology right?").

These problems resemble those that knowledge engineers have dealt with over the last two decades as they worked on knowledge acquisition methodologies or workbenches for defining knowledge bases. The integration of knowledge acquisition with machine-learning techniques proved extremely beneficial for knowledge acquisition.[2] The drawback to such approaches,[3] however, was their rather strong focus on structured knowledge or databases, from which they induced their rules.

Conversely, in the Web environment we encounter when building Web ontologies, structured knowledge bases or databases are the exception rather than the norm. Hence, intelligent support tools for an ontology engineer take on a different meaning than the integration architectures for more conventional knowledge acquisition.[4]

In ontology learning, we aim to integrate numerous disciplines to facilitate ontology construction, particularly machine learning. Because fully automatic machine knowledge acquisition remains in the distant future, we consider ontology learning as semiautomatic with human intervention, adopting the paradigm of balanced cooperative modeling for constructing ontologies for the Semantic Web.[5] With this objective in mind, we built an architecture that combines knowledge acquisition with machine learning, drawing on resources that we find on the syntactic Web—free text, semistructured text, schema definitions (such as document type definitions [DTDs]), and so on. Thereby, our framework's modules serve different steps in the engineering cycle (see Figure 1):

- Merging existing structures or defining mapping rules between these structures allows *importing* and *reusing* existing ontologies. **(**For instance, Cyc's ontological structures have been used to construct a domain-specific ontology.[6])
- Ontology *extraction* models major parts of the target ontology, with learning support fed from Web documents.
- The target ontology's rough outline, which results from import, reuse, and extraction, is *pruned* to better fit the ontology to its primary purpose.
- Ontology *refinement* profits from the pruned ontology but completes the ontology at a fine granularity (in contrast to extraction).
- The target application serves as a measure for validating the resulting ontology.[7]

Finally, the ontology engineer can begin this cycle again—for example, to include new domains in the constructed ontology or to maintain and update its scope.

Architecture

Given the task of constructing and maintaining an ontology for a Semantic Web application such as an ontology-based knowledge portal,[8] we produced support for the ontology engineer embedded in a comprehensive architecture (see Figure 2). The ontology engineer only interacts via the graphical interfaces, which comprise two of the four components: the OntoEdit Ontology Engineering Workbench and the Management Component. Resource Processing and the Algorithm Library are the architecture's remaining components.

The OntoEdit Ontology Engineering Workbench offers sophisticated graphical means for manual modeling and refining of the final ontology. The interface gives the user different views, targeting the epistemological level rather than a particular representation language. However, the user can export the ontological structures to standard Semantic Web representation languages such as OIL (ontology interchange language) and DAML-ONT (*DAML ont*ology language), as well as our own F-Logic-based extensions of RDF(S)—we use RDF(S) to refer to the combined technologies

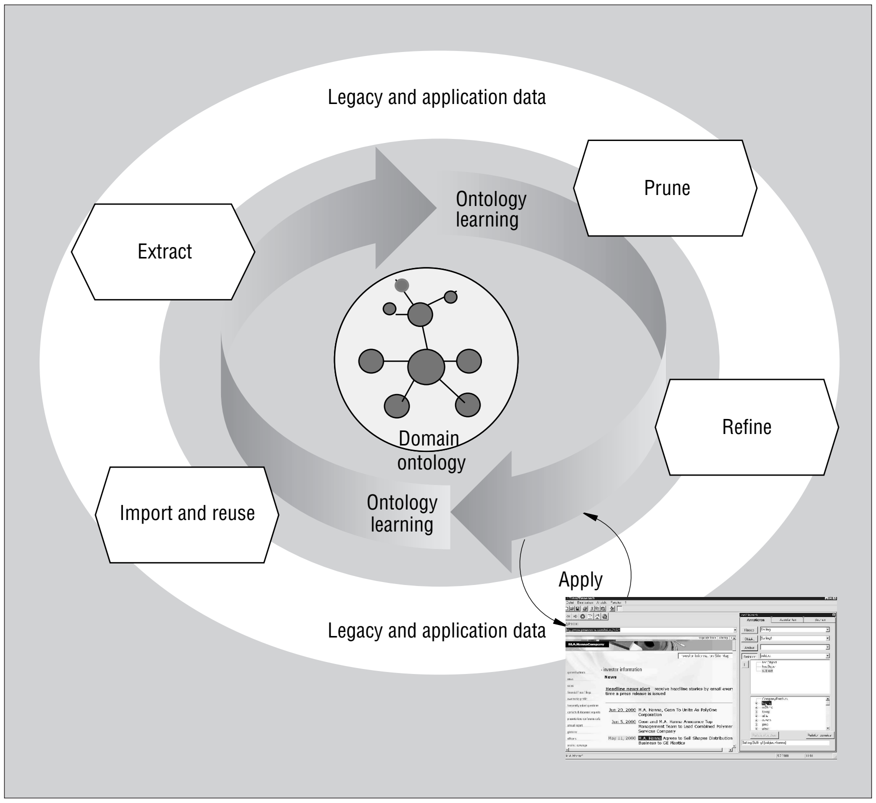


Figure 1. The ontology-learning process.

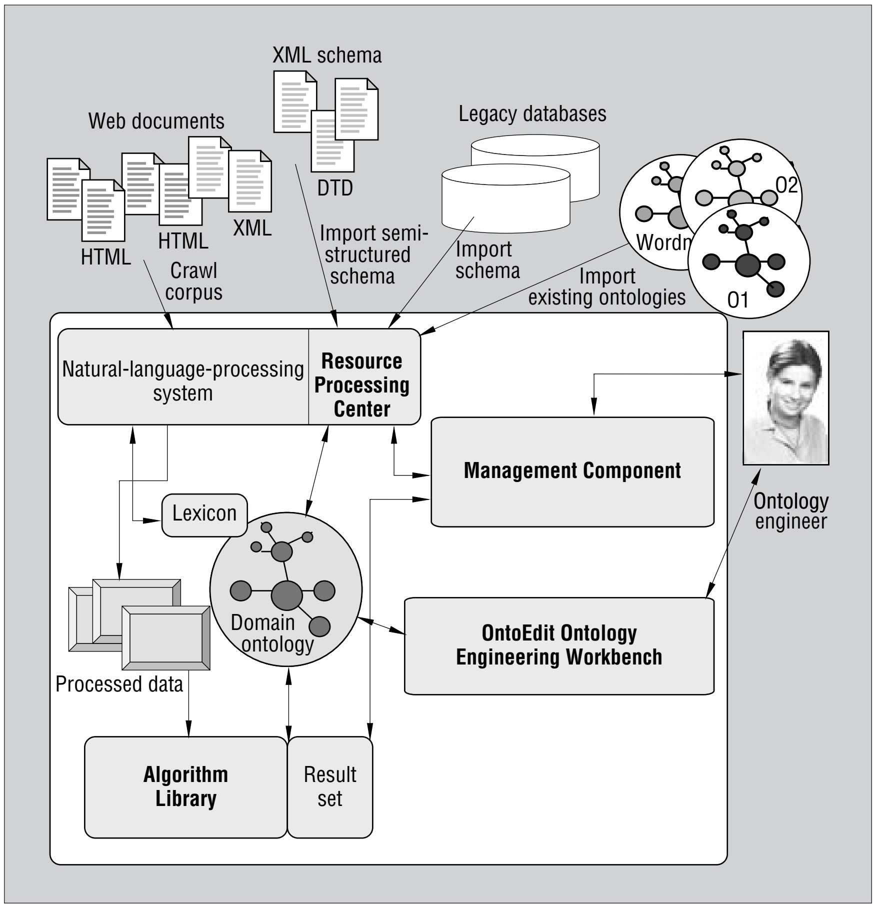


Figure 2. Ontology-learning architecture for the Semantic Web.

of the resource description framework and RDF Schema. Additionally, users can generate and access executable representations for constraint checking and application debugging through SilRi (*si*mple *l*ogic-based *R*DF *i*nterpreter, www.ontoprise.de), our F-Logic inference engine, which connects directly to OntoEdit.

We knew that sophisticated ontology-engineering tools—for example, the Protégé modeling environment for knowledge-based systems[1]—would offer capabilities roughly comparable to OntoEdit. However, in trying to construct a knowledge portal, we found that a large conceptual gap existed 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 determine the target ontology. Into this void we have positioned new components of our ontology-learning architecture (see 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 L for concepts and relations;
- a set of concepts C (roughly akin to synsets in WordNet[9]);
- a taxonomy of concepts with multiple inheritance (heterarchy) H_C;
- a set of nontaxonomic relations R described by their domain and range restrictions;
- a heterarchy of relations—H_R;
- relations F and G that relate concepts and relations with their lexical entries; and
- a set of axioms A that describe additional constraints on the ontology and make implicit facts explicit.[8]

This structure corresponds closely to RDF(S), except for the explicit consideration of lexical entries. Separating concept reference from concept denotation permits very domain-specific ontologies without incurring an instantaneous conflict when `merging` ontologies—a standard Semantic Web request. For instance, the lexical entry *school* in one ontology might refer to a building in ontology A, an organization in ontology B, or both in ontology C. Also, in ontology A, we can refer to the concept referred to in English by *school* and *school building* by the German *Schule* and *Schulgebäude*.

Ontology learning relies on an ontology structured along these lines and on input data as described earlier to propose new knowledge about reasonably interesting concepts, relations, and lexical entries or about links between these entities—proposing some for addition, deletion, or merging. The graphical result set presents the ontology-learning process's results to the ontology engineer (we'll discuss this further in the "Association rules" section). The ontology engineer can then browse the results and decide to follow, delete, or modify the proposals, as the task requires.

Components

By integrating the previously discussed considerations into a coherent generic architecture for extracting and maintaining ontologies from Web data, we have identified several core components (including the graphical user interface discussed earlier).

In trying to construct a knowledge portal, we found that a large conceptual gap existed between the ontology-engineering tool and the input (often legacy data).

Management component graphical user interface

The ontology engineer uses the management component to select input data—that is, relevant resources such as HTML and XML documents, DTDs, databases, or existing ontologies that the discovery process can further exploit. Then, using the management component, the engineer chooses from a set of resource-processing methods available in the resource-processing component and from a set of algorithms available in the algorithm library.

The management component also supports the engineer in discovering task-relevant legacy data—for example, an ontology-based crawler gathers HTML documents that are relevant to a given core ontology.

Resource processing

Depending on the available input data, the engineer can choose various strategies for resource processing:

- Index and reduce HTML documents to free text.
- Transform semistructured documents, such as dictionaries, into a predefined relational structure.
- Handle semistructured and structured schema data (such as DTDs, structured database schemata, and existing ontologies) by following different strategies for import, as described later in this article.
- Process free natural text. Our system accesses the natural-language-processing system Saarbrücken Message Extraction System, a shallow-text processor for German.[10] SMES comprises a *tokenizer* based on regular expressions, a *lexical analysis* component including various word *lexicons*, an *amorphological analysis* module, a *named-entity recognizer*, a *part-of-speech tagger,* and a *chunk parser*.

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

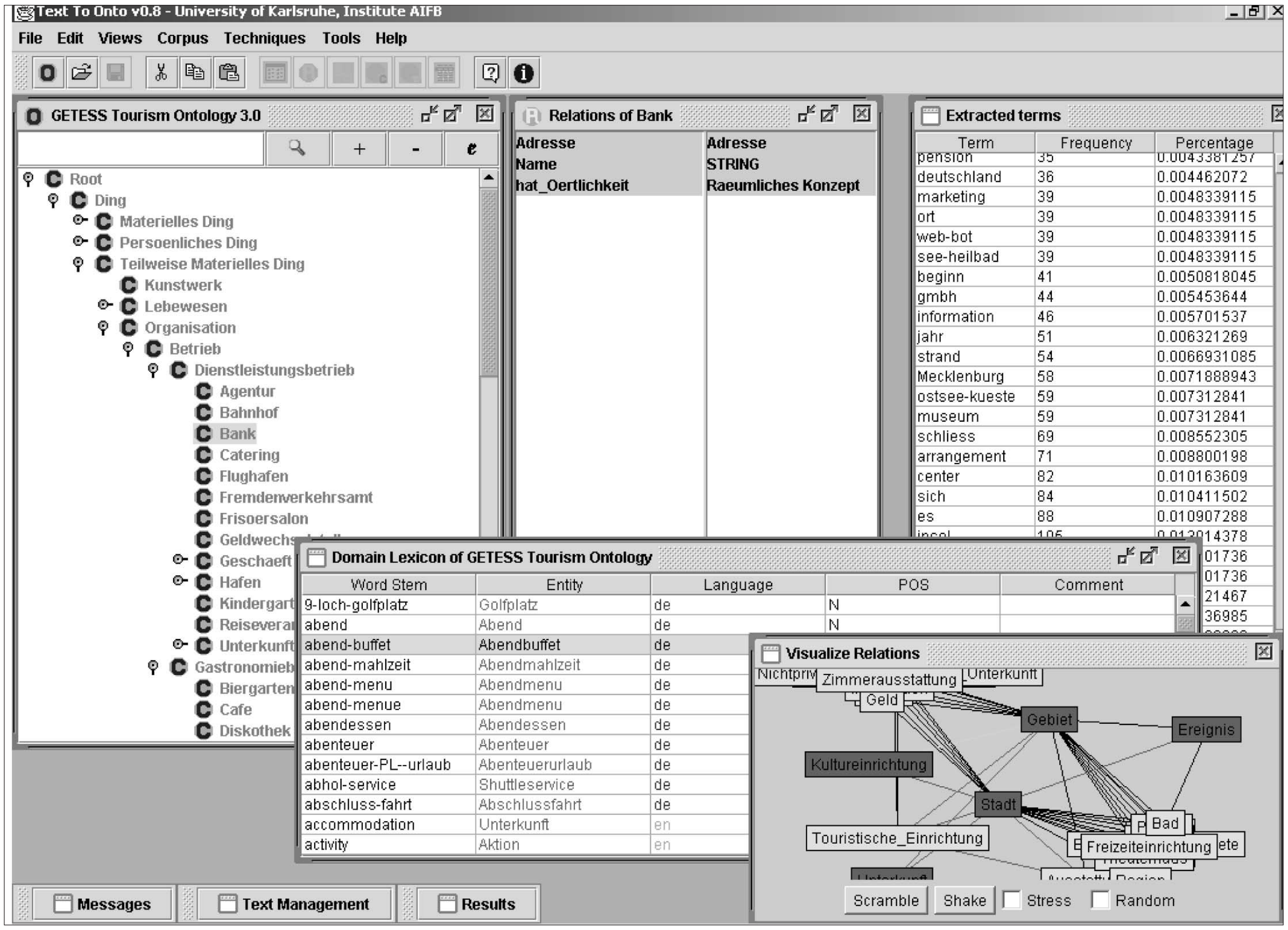
Algorithm library

We can describe an ontology by a number of sets of concepts, relations, lexical entries, and links between these entities. We can acquire an existing ontology definition (including L, C, H_C, R, H_R, A, F, and G), using various algorithms that work on this definition and the preprocessed input data. Although specific algorithms can vary greatly from one type of input to the next, a considerable overlap exists for underlying learning approaches such as association rules, formal concept analysis, or clustering. Hence, we can reuse algorithms from the library for acquiring different parts of the ontology definition.

In our implementation, we generally use a multistrategy learning and result combination approach. Thus, each algorithm plugged into the library generates normalized results that adhere to the ontology structures we've discussed and that we can apply toward a coherent ontology definition.

Import and reuse

Given our experiences in medicine,



- Root
  - Ding
    - Materielles Ding
    - Persoenliches Ding
    - Teilweise Materielles Ding
      - Kunstwerk
      - Lebewesen
      - Organisation
        - Betrieb
          - Dienstleistungsbetrieb
            - Agentur
            - Bahnhof
            - Bank
            - Catering
            - Flughafen
            - Fremdenverkehrsamt
            - Frisoersalon
            - Geldwechs
            - Geschaeft
            - Hafen
            - Kindergart
            - Reiseveran
            - Unterkunft
          - Gastronomieb
            - Biergarten
            - Cafe
            - Diskothek

Adresse	Adresse
Name	STRING
hat_Oertlichkeit	Raeumliches Konzept

Term	Frequency	Percentage
pension	35	0.0043381257
deutschland	36	0.004462072
marketing	39	0.0048339115
ort	39	0.0048339115
web-bot	39	0.0048339115
see-heilbad	39	0.0048339115
beginn	41	0.0050818045
gmbh	44	0.005453644
information	46	0.005701537
jahr	51	0.006321269
strand	54	0.0066931085
Mecklenburg	58	0.0071888943
ostsee-kueste	59	0.007312841
museum	59	0.007312841
schliess	69	0.008552305
arrangement	71	0.008800198
center	82	0.010163609
sich	84	0.010411502
es	88	0.010907288
insel	105	0.013014378
		01736
		01736
		21467
		36985

Word Stem	Entity	Language	POS	Comment
9-loch-golfplatz	Golfplatz	de	N	
abend	Abend	de	N	
abend-buffet	Abendbuffet	de		
abend-mahlzeit	Abendmahlzeit	de		
abend-menu	Abendmenu	de		
abend-menue	Abendmenu	de		
abendessen	Abendessen	de		
abenteuer	Abenteuer	de		
abenteuer-PL--urlaub	Abenteuerurlaub	de		
abhol-service	Shuttleservice	de		
abschluss-fahrt	Abschlussfahrt	de		
accommodation	Unterkunft	en		
activity	Aktion	en		

Figure 3. Screenshot of our ontology-learning workbench, Text-To-Onto.

telecommunications, tourism, and insurance, we expect that domain conceptualizations are available for almost any commercially significant domain. Thus, we need mechanisms and strategies to import and reuse domain conceptualizations from existing (schema) structures. We can recover the conceptualizations, for example, from legacy database schemata, DTDs, or from existing ontologies that conceptualize some relevant part of the target ontology.

In the first part of import and reuse, we identify the schema structures and discuss their general content with domain experts. We must import each of these knowledge sources separately. We can also import manually—which can include a manual definition of transformation rules. Alternatively, reverse-engineering tools—such as those that exist for recovering extended entity-relationship diagrams from a given database's SQL description (see the sidebar)—might facilitate the recovery of conceptual structures.

In the second part of the import and reuse step, we must merge or align imported conceptual structures to form a single common ground from which to springboard into the subsequent ontology-learning phases of extracting, pruning, and refining. Although the general research issue of merging and aligning is still an open problem, recent proposals have shown how to improve the manual merging and aligning process. Existing methods mostly rely on matching heuristics for proposing the merger of concepts and similar knowledge base operations. Our research also integrates mechanisms that use an application-data–oriented, bottom-up approach.[11] For instance, formal concept analysis lets us discover patterns between application data and the use of concepts, on one hand, and their heterarchies' relations and semantics, on the other, in a formally concise way (see B. Ganter and R. Wille's work on formal concept analysis in the sidebar).

Overall, the ontology-learning import and reuse step seems to be the hardest to generalize. The task vaguely resembles the general problems encountered in data-warehousing—adding, however, challenging problems of its own.

Extraction

Ontology-extraction models major parts—the complete ontology or large chunks representing a new ontology subdomain—with learning support exploiting various types of Web sources. Ontology-learning techniques partially rely on given ontology parts. Thus, we here encounter an iterative model where previous revisions through the ontology-learning cycle can propel subsequent ones, and more sophisticated algorithms can work on structures that previous, more straightforward algorithms have proposed.

To describe this phase, let's look at some of the techniques and algorithms that we embedded in our framework and implemented in our ontology-learning environment Text-To-Onto (see Figure 3). We cover a substantial part of the overall ontology-learning task in the extraction phase. Text-To-Onto proposes many different ontology learning algorithms for primitives, which we described previously (that is, L, C, R, and so on), to the ontology engineer building on several types of input.

A Common Perspective

Until recently, ontology learning—for comprehensive ontology construction—did not exist. However, much work in numerous disciplines—computational linguistics, information retrieval, machine learning, databases, and software engineering—has researched and practiced techniques that we can use in ontology learning. Hence, we can find techniques and methods relevant for ontology learning referred to as

- "acquisition of selectional restrictions,"[1,2]
- "word sense disambiguation and learning of word senses,"[3]
- "computation of concept lattices from formal contexts,"[4] and
- "reverse engineering in software engineering."[5]

Ontology learning puts many research activities—which focus on different input types but share a common domain conceptualization—into one perspective. The activities in Table A span a variety of communities, with references from 20 completely different events and journals.

References

1. P. Resnik, *Selection and Information: A Class-Based Approach to Lexical Relationships*, PhD thesis, Dept. of Computer Science, Univ. of Pennsylvania, Philadelphia, 1993.
2. R. Basili, M.T. Pazienza, and P. Velardi, "Acquisition of Selectional Patterns in a Sublanguage," *Machine Translation*, vol. 8, no. 1, 1993, pp. 175–201.
3. P. Wiemer-Hastings, A. Graesser, and K. Wiemer-Hastings, "Inferring the Meaning of Verbs from Context," *Proc. 20th Ann. Conf. Cognitive Science Society* (CogSci-98), Lawrence Erlbaum, New York, 1998.
4. B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer-Verlag, Berlin, 1999.
5. H.A. Mueller et al., "Reverse Engineering: A Roadmap," *Proc. Int'l Conf. Software Eng.* (ICSE-00), ACM Press, New York, 2000, pp. 47–60.
6. P. Buitelaar, *CORELEX Systematic Polysemy and Underspecification*, PhD thesis, Dept. of Computer Science, Brandeis Univ., Waltham, Mass., 1998.
7. H. Assadi, "Construction of a Regional Ontology from Text and Its Use within a Documentary System," *Proc. Int'l Conf. Formal Ontology and Information Systems* (FOIS-98), IOS Press, Amsterdam.
8. D. Faure and C. Nedellec, "A Corpus-Based Conceptual Clustering Method for Verb Frames and Ontology Acquisition," *Proc. LREC-98 Workshop on Adapting Lexical and Corpus Resources to Sublanguages and Applications*, European Language Resources—Distribution Agency, Paris, 1998.
9. F. Esposito et al., "Learning from Parsed Sentences with INTHELEX," *Proc. Learning Language in Logic Workshop* (LLL-2000) and *Learning Language in Logic Workshop* (LLL-2000), Assoc. for Computational Linguistics, New Brunswick, N.J., 2000, pp. 194-198.
10. A. Maedche and S. Staab, "Discovering Conceptual Relations from Text," *Proc. European Conf. Artificial Intelligence* (ECAI-00), IOS Press, Amsterdam, 2000, pp. 321–325.
11. J.-U. Kietz, A. Maedche, and R. Volz, "Semi-Automatic Ontology Acquisition from a Corporate Intranet." *Proc. Learning Language in Logic Workshop* (LLL-2000), ACL, New Brunswick, N.J., 2000, pp. 31–43.
12. E. Morin, "Automatic Acquisition of Semantic Relations between Terms from Technical Corpora," *Proc. of the Fifth Int'l Congress on Terminology and Knowledge Engineering* (TKE-99), TermNet-Verlag, Vienna, 1999.
13. U. Hahn and K. Schnattinger, "Towards Text Knowledge Engineering," *Proc. Am. Assoc. for Artificial Intelligence* (AAAI-98), AAAI/MIT Press, Menlo Park, Calif., 1998.
14. M.A. Hearst, "Automatic Acquisition of Hyponyms from Large Text Corpora," *Proc. Conf. Computational Linguistics* (COLING-92), 1992.
15. Y. Wilks, B. Slator, and L. Guthrie, *Electric Words: Dictionaries, Computers, and Meanings*, MIT Press, Cambridge, Mass., 1996.
16. J. Jannink and G. Wiederhold, "Thesaurus Entry Extraction from an On-Line Dictionary," *Proc. Second Int'l Conf. Information Fusion* (Fusion-99), Omnipress, Wisconsin, 1999.
17. J.-U. Kietz and K. Morik, "A Polynomial Approach to the Constructive Induction of Structural Knowledge," *Machine Learning*, vol. 14, no. 2, 1994, pp. 193–211.
18. S. Schlobach, "Assertional Mining in Description Logics," *Proc. 2000 Int'l Workshop on Description Logics* (DL-2000), 2000; http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-33.
19. A. Doan, P. Domingos, and A. Levy, "Learning Source Descriptions for Data Integration," *Proc. Int'l Workshop on The Web and Databases* (WebDB-2000), Springer-Verlag, Berlin, 2000, pp. 60–71.
20. P. Johannesson, "A Method for Transforming Relational Schemas into Conceptual Schemas," *Proc. Int'l Conf. Data Engineering* (IDCE-94), IEEE Press, Piscataway, N.J., 1994, pp. 190–201.
21. Z. Tari et al., "The Reengineering of Relational Databases Based on Key and Data Correlations," *Proc. Seventh Conf. Database Semantics* (DS-7), Chapman & Hall, 1998, pp. 40–52.

Table A. A survey of ontology-learning approaches.

Domain	Methods	Features used	Prime purpose	Papers
Free Text	Clustering	Syntax	Extract	Paul Buitelaar,[6] H. Assadi,[7] and David Faure and Claure Nedellec[8]
	Inductive logic programming	Syntax, logic representation	Extract	Frederique Esposito et al.[9]
	Association rules	Syntax, Tokens	Extract	Alexander Maedche and Steffen Staab[10]
	Frequency-based	Syntax	Prune	Joerg-Uwe Kietz et al.[11]
	Pattern matching	—	Extract	Emanuelle Morin[12]
	Classification	Syntax, semantics	Refine	Udo Hahn and Klemens Schnattinger[13]
Dictionary	Information extraction	Syntax	Extract	Marti Hearst,[14] Yorik Wilks,[15] and Joerg-Uwe Kietz et al.[11]
	Page rank	Tokens	—	Jan Jannink and Gio Wiederhold[16]
Knowledge base	Concept induction, A-Box mining	Relations	Extract	Joerg-Uwe Kietz and Katharina Morik[17] and S. Schlobach[18]
Semistructured schemata	Naive Bayes	Relations	Reverse engineering	Anahai Doan et al.[19]
Relational schemata	Data correlation	Relations	Reverse engineering	Paul Johannesson[20] and Zahir Tari et al.[21]

Lexical entry and concept extraction

One of the baseline methods applied in our framework for acquiring lexical entries with corresponding concepts is lexical entry and concept extraction. Text-To-Onto processes Web documents on the morphological level, including multiword terms such as "database reverse engineering" by n-grams, a simple statistics-based technique. Based on this text preprocessing, we apply term-extraction techniques, which are based on (weighted) statistical frequencies, to propose new lexical entries for L.

Often, the ontology engineer follows the proposal by the lexical entry and 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 taxonomic concept classification. One generally applicable method with regard to this is hierarchical clustering, which exploits items' similarities to propose a hierarchy of item categories. We compute the similarity measure on the properties of items.

When extracting a hierarchy from natural-language text, term adjacency or syntactical relationships between terms yield considerable descriptive power to induce the semantic hierarchy of concepts related to these terms.

David Faure and Claure Nedellec give a sophisticated example for hierarchical clustering (see the sidebar). They present a cooperative machine-learning system, Asium (*a*cquisition of *s*emant*i*c knowledge *u*sing *m*achine-learning method), which acquires taxonomic relations and subcategorization frames of verbs based on syntactic input. The Asium system hierarchically clusters nouns based on the verbs to which they are syntactically related and vice versa. Thus, they cooperatively extend the lexicon, the concept set, and the concept heterarchy (L, C, H_C).

Dictionary parsing

Machine-readable dictionaries are frequently available for many domains. Although their internal structure is mostly free text, comparatively few patterns are used to give text definitions. Hence, MRDs exhibit a large degree of regularity that can be exploited to extract a domain conceptualization.

We have used Text-To-Onto to generate a concept taxonomy from an insurance company's MRD (see the sidebar). Likewise, we've applied morphological processing to term extraction from free text—this time, however, complementing several pattern-matching heuristics. Take, for example, the following dictionary 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….

We applied several heuristics to the morphologically analyzed definitions. For instance, one simple heuristic relates the definition term, here *automatic debit transfer*, with the first noun phrase in the definition, here *electronic service*. The heterarchy $H_C : H_C$ (automatic debit transfer, electronic service) links their corresponding concepts. Applying this heuristic iteratively, we can propose large parts of the target ontology—more precisely, L, C, and H_C to the ontology engineer. In fact, because verbs tend to be modeled as relations, we can also use this method to extend R (and the linkage between R and L).

Targeting completeness for the domain model appears to be practically unmanageable and computationally intractable, but targeting the scarcest model overly limits expressiveness.

Association rules

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

In Text-To-Onto (see the sidebar), we use a modified generalized association-rule-learning algorithm to discover relations between concepts. A given class hierarchy H_C serves as background knowledge. Pairs of syntactically related concepts—for example, pair (festival,island) describing the head–modifier relationship contained in the sentence "The festival on Usedom attracts tourists from all over the world."—are given as input to the algorithm. The algorithm generates association rules that compare the relevance of different rules while climbing up or down the taxonomy. The algorithm proposes what appears to be the most relevant binary rules to the ontology engineer for modeling relations into the ontology, thus extending R.

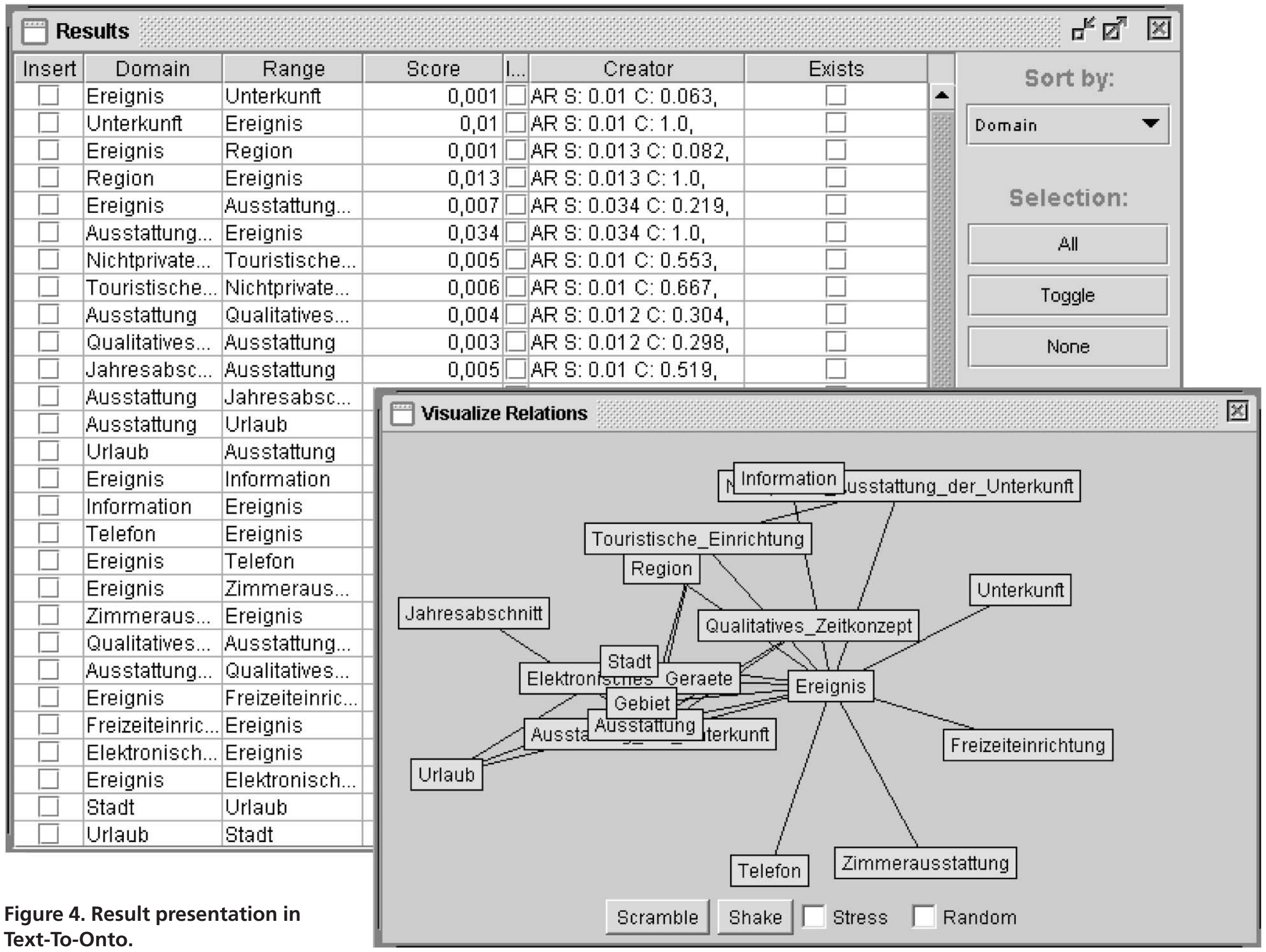
As the algorithm tends to generate a high number of rules, we offer various interaction modes. For example, the ontology engineer can restrict the number of suggested relations by defining so-called restriction concepts that must participate in the extracted relations. The flexible enabling and disabling of taxonomic knowledge for extracting relations is another way of focusing.

Figure 4 shows various views of the results. We can induce a generalized relation from the example data given earlier—relation rel(event,area), which the ontology engineer could name locatedin, namely, events located in an area (which extends L and G). The user can add extracted relations to the ontology by dragging and dropping them. To explore and determine the right aggregation level of adding a relation to the ontology, the user can browse the relation views for extracted properties (see the left side of Figure 4).

Pruning

A common theme of modeling in various disciplines is the balance between completeness and domain-model scarcity. Targeting completeness for the domain model appears to be practically unmanageable and computationally intractable, but targeting the scarcest model overly limits expressiveness. Hence, we aim for a balance between the two that works. Our model should capture a rich target-domain conceptualization but exclude the parts out of its focus. Ontology import and reuse as well as ontology extraction put the scale considerably out of balance where out-of-focus concepts reign. Therefore, we appropriately diminish the ontology in the pruning phase.

We can view the problem of pruning in at least two ways. First, we need to clarify how



Insert	Domain	Range	Score	I...	Creator	Exists
☐	Ereignis	Unterkunft	0,001	☐	AR S: 0.01 C: 0.063,	☐
☐	Unterkunft	Ereignis	0,01	☐	AR S: 0.01 C: 1.0,	☐
☐	Ereignis	Region	0,001	☐	AR S: 0.013 C: 0.082,	☐
☐	Region	Ereignis	0,013	☐	AR S: 0.013 C: 1.0,	☐
☐	Ereignis	Ausstattung...	0,007	☐	AR S: 0.034 C: 0.219,	☐
☐	Ausstattung...	Ereignis	0,034	☐	AR S: 0.034 C: 1.0,	☐
☐	Nichtprivate...	Touristische...	0,005	☐	AR S: 0.01 C: 0.553,	☐
☐	Touristische...	Nichtprivate...	0,006	☐	AR S: 0.01 C: 0.667,	☐
☐	Ausstattung	Qualitatives...	0,004	☐	AR S: 0.012 C: 0.304,	☐
☐	Qualitatives...	Ausstattung	0,003	☐	AR S: 0.012 C: 0.298,	☐
☐	Jahresabsc...	Ausstattung	0,005	☐	AR S: 0.01 C: 0.519,	☐
☐	Ausstattung	Jahresabsc...				
☐	Ausstattung	Urlaub				
☐	Urlaub	Ausstattung				
☐	Ereignis	Information				
☐	Information	Ereignis				
☐	Telefon	Ereignis				
☐	Ereignis	Telefon				
☐	Ereignis	Zimmeraus...				
☐	Zimmeraus...	Ereignis				
☐	Qualitatives...	Ausstattung...				
☐	Ausstattung...	Qualitatives...				
☐	Ereignis	Freizeiteinric...				
☐	Freizeiteinric...	Ereignis				
☐	Elektronisch...	Ereignis				
☐	Ereignis	Elektronisch...				
☐	Stadt	Urlaub				
☐	Urlaub	Stadt				

Figure 4. Result presentation in Text-To-Onto.

pruning particular parts of the ontology (for example, removing a concept or relation) affects the rest. For instance, Brian Peterson and his colleagues have described strategies that leave the user with a coherent ontology (that is, no dangling or broken links).[6] Second, we can consider strategies for proposing ontology items that we should either keep or prune. Given a set of application-specific documents, several strategies exist for pruning the ontology that are based on absolute or relative counts of term frequency combined with the ontology's background knowledge (see the sidebar).

Refinement

Refining plays a similar role to extracting—the difference is on a sliding scale rather than a clear-cut distinction. Although extracting serves mainly 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 can use data that comes from a concrete Semantic Web application—for example, log files of user queries or generic user data. Adapting and refining the ontology with respect to user requirements plays a major role in the application's acceptance and its further development.

In principle, we can use the same algorithms for extraction and refinement. However, during refinement, we must consider in detail the existing ontology and its existing connections, while extraction works more often than not practically from scratch.

Udo Hahn and Klemens Schnattinger presented a prototypical approach for refinement (see the sidebar)—although not for extraction! They introduced a methodology for automating the maintenance of domain-specific taxonomies. This incrementally updates an ontology as it acquires new concepts from text. The acquisition process is centered on the linguistic and conceptual "quality" of various forms of evidence underlying concept-hypothesis generation and refinement. Particularly, to determine a particular proposal's quality, Hahn and Schnattinger consider semantic conflicts and analogous semantic structures from the knowledge base for the ontology, thus extending an existing ontology with new lexical entries for L, new concepts for C, and new relations for H_C.

Ontology learning could add significant leverage to the Semantic Web because it propels the construction of domain ontologies, which the Semantic Web needs to succeed. We have presented a comprehensive framework for ontology learning that crosses the boundaries of single disciplines, touching on a number of challenges. The good news is, however, that you don't need perfect or optimal support for cooperative ontology modeling. At least according to our experience, cheap methods in an integrated environment can tremendously help the ontology engineer.

While a number of problems remain within individual disciplines, additional challenges arise that specifically pertain to applying ontology learning to the Semantic Web. With the use of XML-based namespace mechanisms, the notion of an ontology with well-defined boundaries—for example, only definitions that are in one file—will disappear. Rather, the Semantic Web might yield an amoeba-like structure regarding ontology boundaries because ontologies refer to and import each other (for example, the DAML-ONT primitive import). However, we do not yet know what the semantics of these structures will look like. In light of these facts, the importance of methods such as ontology pruning and crawling will drastically increase. Moreover, we have so far restricted our attention in ontology learning to the conceptual structures that are almost contained in RDF(S). Additional semantic layers on top of RDF (for example, future OIL or DAML-ONT with axioms, A) will require new means for improved ontology engineering with axioms, too! ■

Acknowledgments

We thank our students, Dirk Wenke, and Raphael Volz for work on *OntoEdit* and *Text-To-Onto*. Swiss Life/Rentenanstalt (Zurich, Switzerland), Ontoprise GmbH (Karlsruhe, Germany), the US Air Force DARPA DAML ("OntoAgents" project), the European Union (IST-1999-10132 "On-To-Knowledge" project), and the German BMBF (01IN901C0 "GETESS" project) partly financed research for this article.

References

1. E. Grosso et al., "Knowledge Modeling at the Millennium—the Design and Evolution of Protégé-2000," *Proc. 12th Int'l Workshop Knowledge Acquisition, Modeling and Management* (KAW-99), 1999.
2. G. Webb, J. Wells, and Z. Zheng, "An Experimental Evaluation of Integrating Machine Learning with Knowledge Acquisition," *Machine Learning*, vol. 35, no. 1, 1999, pp. 5–23.
3. K. Morik et al., *Knowledge Acquisition and Machine Learning: Theory, Methods, and Applications*, Academic Press, London, 1993.
4. B. Gaines and M. Shaw, "Integrated Knowledge Acquisition Architectures," *J. Intelligent Information Systems*, vol. 1, no. 1, 1992, pp. 9–34.
5. K. Morik, "Balanced Cooperative Modeling," *Machine Learning*, vol. 11, no. 1, 1993, pp. 217–235.
6. B. Peterson, W. Andersen, and J. Engel, "Knowledge Bus: Generating Application-Focused Databases from Large Ontologies," *Proc. Fifth Workshop Knowledge Representation Meets Databases* (KRDB-98), 1998, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-10 (current 19 Mar. 2001).
7. S. Staab et al., "Knowledge Processes and Ontologies," *IEEE Intelligent Systems*, vol. 16, no. 1, Jan./Feb. 2001, pp. 26–34.
8. S. Staab and A. Maedche, "Knowledge Portals—Ontologies at Work," to be published in *AI Magazine*, vol. 21, no. 2, Summer 2001.
9. G. Miller, "WordNet: A Lexical Database for English," *Comm. ACM*, vol. 38, no. 11, Nov. 1995, pp. 39–41.
10. G. Neumann et al., "An Information Extraction Core System for Real World German Text Processing," *Proc. Fifth Conf. Applied Natural Language Processing* (ANLP-97), 1997, pp. 208–215.
11. G. Stumme and A. Maedche, "FCA-Merge: A Bottom-Up Approach for Merging Ontologies," to be published in *Proc. 17th Int'l Joint Conf. Artificial Intelligence* (IJCAI '01), Morgan Kaufmann, San Francisco, 2001.

The Authors

Alexander Maedche is a PhD student at the Institute of Applied Informatics and Formal Description Methods at the University of Karlsruhe. His research interests include knowledge discovery in data and text, ontology engineering, learning and application of ontologies, and the Semantic Web. He recently founded together with Rudi Studer a research group at the FZI Research Center for Information Technologies at the University of Karlsruhe that researches Semantic Web technologies and applies them to knowledge management applications in practice. He received a diploma in industrial engineering, majoring in computer science and operations research, from the University of Karlsruhe. Contact him at the Institute AIFB, Univ. of Karlsruhe, 76128 Karlsruhe, Germany; ama@aifb.uni-karlsruhe.de.

Steffen Staab is an assistant professor at the University of Karlsruhe and cofounder of Ontoprise GmbH. His research interests include computational linguistics, text mining, knowledge management, ontologies, and the Semantic Web. He received an MSE from the University of Pennsylvania and a Dr. rer. nat. from the University of Freiburg, both in informatics. He organized several national and international conferences and workshops, and is now chairing the Semantic Web Workshop in Hongkong at WWW10. Contact him at the Institute AIFB, Univ. of Karlsruhe, 76128 Karlsruhe, Germany; sst@aifb.uni-karlsruhe.de.