An Ontology for Domain-oriented Semantic Similarity Search on XML Data

Anja Theobald
Department of Computer Science
University of the Saarland, Germany
WWW: http://www-dbs.cs.uni-sb.de
E-mail: theobald@cs.uni-sb.de

Abstract: Query languages for XML such as XPath or XQuery support Boolean retrieval where a query result is a (possibly restructured) subset of XML elements or entire documents that satisfy the search conditions of the query. Web search engines, on the other hand, are based on the ranked retrieval paradigm, but do not consider the additional information and rich annotations provided by the structure of XML documents and their element names. Furthermore, web search engines have very little "semantic" data and are thus unable to cope with ambiguous search terms. Ontological knowledge and appropriate index structures are necessary for semantic similarity search on XML data extracted from the web. In this paper we present a powerful ontology index which supports domain-specific semantic similarity search with a new measure for expressing the relevance of query results.

1. Introduction

1.1 Motivation

XML is becoming the standard for integrating and exchanging data over the Internet and within intranets, covering the complete spectrum from largely unstructured, ad hoc documents to highly structured, schematic data. For searching information in open environments such as the Web or intranets of large corporations, ranked retrieval is required: a query result is a rank list of XML elements in descending order of relevance.
If you are looking for information about the composition of a chip you will get many relevant documents. But this is a mix of information about games (e.g. poker), hardware (e.g. microchip), animals (e.g. cow chip), food (e.g. potato chip) and further domains. The ambiguity of words and the need for a semantically meaningful interpretation of data call for ontology-based domain-oriented semantic similarity search and an appropriate ontology index.

Definition 1: An *ontology/thesaurus* is a specification of representational vocabulary of words/*terms* including *hierarchical relationships* and *associative relationships* between these words. It is used for indexing and investigation as well as for support knowledge sharing and reuse [Gru93, Gua98].

Consider the word "chip". For this term we know several related terms:

snack food	— is a broader term (hypernym) of →	chip
microprocessor	— is a narrower term (hyponym) of →	chip
blue chip	— is a narrower term (hyponym) of →	chip

If we insert this information into a simple tree-based term hierarchy (see Figure 1a) as described in [STW01, TW02] we obtain the wrong impression that "snack food" and "blue chip" are related terms.

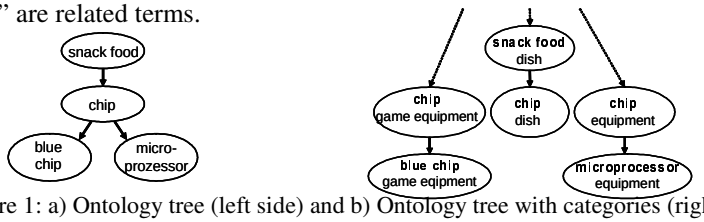

Figure 1: a) Ontology tree (left side) and b) Ontology tree with categories (right side)

So we need a method to distinguish different senses of a term. For this purpose we introduce a notion of a *category* for more precisely specifying the meaning of a term. As a representative of a category for term t we use a broader term (hypernym) that comprises t as a narrower concept (hyponym), e.g. term: **chip** – category: microchip. The upper bold-faced entries are ontology terms and the second entries are the corresponding categories (see also Figure 1b). To produce a ranked result for an ontology-based search we need a similarity assessment of a relationship between two ontology terms.

For elaborating the idea above an obvious approach would be to use an existing thesaurus. For example, WordNet [WN98] is an extensive electronic lexical database. For a given search term WordNet is able to distinguish several senses of this term and is able to return related terms to the given term. Unfortunately, WordNet supports only term-based search (no search for a given sense) without any quantification of the relationships between related terms.

Our work extracts semantic information from WordNet (and possibly other ontological sources) for building a quantified graph-like structure as a backbone for XML similarity search.

1.2 Related Work

Researchers in artificial intelligence first developed logic-based ontologies to facilitate knowledge sharing and reuse. In the last decade ontologies have become a popular research topic and several AI research communities such as knowledge engineering, natural language processing and knowledge representation have investigated them. The interest in ontologies has been revived with the recent discussion about the "Semantic Web" [SemWeb, MWK00, SAD+00]. In contrast to the extremely ambitious early AI approaches toward building universal ontologies (see, e.g., [LG90, RN95]), more recent proposals are aiming at domain- or user-specific ontologies and are based on more tractable logics (see, e.g., [Hor02, SAD+00]).

Recent publications on formalization of ontologies cover a wide spectrum from algebraic approaches [BM99] and logic-based languages for modeling ontologies [DEFS99, BEH+02] to ontologies for conceptual data modeling and data interpretation [Gua98, NFM00, SM00]. Similar work has been done in the context of multi-databases [BHP94, PM98]. These publications do not consider quantification of relationships. Only few papers [KKS01, KKC94, KI90, STW01, BHP94] describe weighted ontologies. These approaches often define simple weights for term similarity not appropriate for query processing.

To our knowledge, the role of ontologies in searching semistructured data has not yet been discussed in any depth. The unique characteristic of our approach lies in the combination of ontological knowledge and information retrieval techniques for domain-oriented semantic similarity search on XML data.

1.3 Contribution and Outline of the Paper

In this paper we present a combination of ontological knowledge and information retrieval techniques for domain-oriented semantic similarity search on XML data. For this purpose we have developed an ontology which contains a category for each term to recognize domains. Furthermore, we introduce a quantification of relationships between ontology terms using similarity weights. For automatic ontology construction we build on the WordNet thesaurus. For semantic domain-oriented querying XML data we use the flexible XML search language called XXL.

The rest of the paper is organized as follows. Section 2 gives a short description of the flexible XML search language called XXL and the architecture of the XXL search engine. Section 3 presents the modeling of the ontology index using categories and differentiated relationships between ontology terms. Section 4 outlines the application of the ontology index for evaluation of semantic similarity search conditions within XXL queries.

2. XXL: A Flexible XML Search Language for Ranked Retrieval

For the flexible XML search language called XXL we have adopted several concepts from XML-QL, XQuery and similar languages as the core, with certain simplifications and resulting restrictions.

For searching publications over chips according to the example scenario given in the motivation we can express such a query in our language XXL as follows:

```
SELECT   T                          // output of the XXL query
FROM     INDEX                      // search space of the XXL query
WHERE    #.~publication AS P        // search condition
   AND   P.title AS T
   AND   P.description ~ "chip"
```

We define the Where clause of a query as the logical conjunction of *path expressions*, where a path expression is a regular expression over *elementary conditions* and an elementary condition refers to the name or content of a single element or attribute.

In contrast to other XML query languages we introduce a new operator "~" to express semantic similarity search conditions on XML element names as well as on XML element contents. The result of an XXL query is a subgraph of the XML data graph, where the nodes are annotated with local relevance probabilities called *similarity score* for the elementary search conditions given by the query. These similarity scores are combined into a global similarity score for expressing the relevance of the entire result graph. Full details of the semantics of XXL and especially the probabilistic computation of similarity scores can be found in [TW00, TW02].

This XML search language is implemented within the XXL search engine. The XXL search engine is a client-server application with a Java-based GUI. The server programs

consist of 1) service components: the crawler and the query processor (both Java servlets), 2) algorithmic components: parsing and indexing documents, parsing and checking XXL queries, and 3) data components: data structures and their methods for storing various kinds of information like the element path index (EPI), the element content index (ECI), and the element-name ontology index (NOI). For storing the ECI we use Oracle.

An XXL query is evaluated as follows [TW02]. The Where clause of an XXL query is a logical conjunction of n regular expressions for search conditions over XML element paths. The query processor (QP) decomposes the given query into n subqueries and constructs a graph-based representation for each subquery similar to a finite state automaton. The global order of evaluating the n subqueries and the local evaluation strategy (e.g., top-down vs. bottom-up) for each subquery are automatically chosen by the QP.

For each subquery, simple path expressions with element names and the wildcard symbol # are looked up in the EPI. For example, all occurrences of a pattern *#.publication.description* can be retrieved from the EPI. Content conditions are evaluated by the ECI, a text index on element and attribute contents. For "semantic" similarity conditions such as *description ~ "chip"* the ECI yields approximate matches and a similarity score based on IR-style tf*idf measures [BR99] and "semantic distances" between concepts in the ontology. Finally, for semantic similarity conditions on element names, for example, *~publication*, the NOI is used to expand the query in order to capture semantically related element names such as *article*; again, a similarity score is computed for result ranking.

The ranked lists of relevant subgraphs from the index-based subquery evaluation are composed into a list of global graphs, each of which has assigned to it a global similarity score derived from the local scores by elementary probability computation. The QP finally extracts the result as specified by the Select clause and constructs an XML document that is returned to the XXL client.

3. Category-based Ontology

Recall from Definition 1 that an *ontology* is a specification of representational vocabulary of *terms* and contains *relationships* between these terms. Now we present a definition of an ontology containing terms and their categories and relationships between terms based on an existing thesaurus which is able to produce related terms to a given term. In our approach we will use WordNet as a common sense thesaurus.

3.1 Graph-based Data Structure

Our approach pursues the several goals, e.g. modelling of element names and keywords of element contents in separate ontology indexes, improving the similarity weights for expressing correlations between ontology terms, construct the ontology index automatically, and implementing a tool for human management of an ontology index.
For this purpose we have developed an ontology index which supports domain-oriented search using categories as well as more detailed relevance ranking using several relationships between two ontology terms.

First we give a short description of the most important features of WordNet necessary for our following work. For a search term WordNet provides synonyms, 1-level hy-

pernyms (broader terms), 2-level hypernyms, … , hyponyms (narrower terms), meronyms (part of something), holonyms (whole of something) for each meaning of the given term. A 2-level hypernym is more general than a 1-level hypernym, etc.

Definition 2: A *category-based ontology index* (OI) is a directed, labeled graph G=(V,E) where V is a finite set of nodes and E is a finite set of edges. Each node n=(t,c) consists of an ontology *term* t and its *category* c.

For the *category* c of an ontology term t we choose the 2-level hypernym provided by WordNet which best describes the meaning of t. The ontology index contains a special root node n=(entity,entity) where entity is called *top term*. This node has only outgoing edges.

There is an edge between two nodes n1 and n2, if n1.t and n2.t are related terms (e.g. n1.t is a hypernym of n2.t). An edge between two nodes n1 and n2 is a tuple e=(n1,n2,type,weight) where type ∈ {is_synonym_of, is_hypernym_of, is_hyponym_of, is_holonym_of, is_meronym_of} defines the relationship between the two terms n1.t and n2.t and its corresponding categories n1.c and n2.c and the weight denotes the similarity of the ontology terms of n1 and n2 according to its categories.

Figure 2 displays a part of such an ontology index where a node contains the bold faced ontology term and its category. Each edge is labeled with a relationship and a weight between two connected ontology terms (hyper is a shortcut for "is_hypernym_of", etc.). The given weights are fictitious. It is not necessary to model the edges of type hyponym and meronym explicitly, because they are reverse directions of hypernym and holonym, respectively. The relationship "Synonym" is a symmetric, reflexive, and transitive relation.

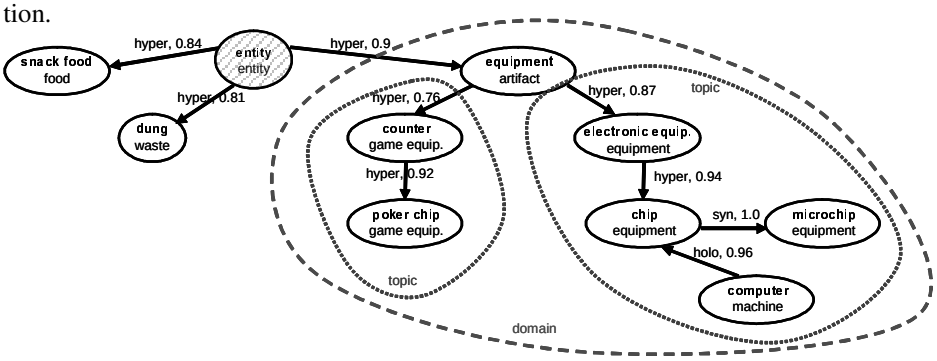

Figure 2: Graph-based ontology index

The ontology index represents the relationships of terms as well as the relationships of categories.

Definition 3: A *topic* is a subgraph of OI which consists of nodes with the same category. All topics are distinct subgraphs in terms of their set of nodes. Two topics are connected, if there exists at least one edge between a node of the first topic and a node of the second topic. Each category of a child of the root=(entity,entity) of the OI is a root (*domain node*) of a subgraph called *domain*.

According to Definition 2 each edge is also denoted with a *weight* to express the similarity of the two corresponding nodes. For a given path p we define the similarity between the start node and the end node using all weights of the corresponding edges and their

distance to the start node. To obtain a fair result we consider the forward path and the backward path of the given path p as defined in Definition 4.

Definition 4: The *similarity sim(v_i,v_j)* between two terms of two ontology nodes v_i and v_j connected by a path p_{ij}=<v_i …v_j> is computed as follows.
Let p_{ij}=<n_0…n_k> where v_i=n_1 and v_j=n_k the forward path and pji=<n_k…n_0> the backward path of p between nodes v_i and v_j. Let length(p)=|p_{ij}|=|p_{ji}|=k.

$$sim(v_i,v_j) = \frac{(weight(p_{ij}) + weight(p_{ji}))}{2 \cdot length(p)}$$

$$weight(p_{ij}) = \sum_{m=0}^{length(p)-1} \frac{length(p)-m}{length(p)} weight(< n_m, n_{m+1} >)$$

$$weight(p_{ji}) = \sum_{m=0}^{length(p)-1} \frac{length(p)-m}{length(p)} weight(< n_{m+1}, n_m >)$$

For example, let v.t=”electronic equipment” be a given search term and w.t=”microchip” a similar term as shown in Figure 2. The similarity score sim(v,w) = ((1·0.94 + 1/2·1.0) + (1·1.0 + 1/2·0.94)) / (2·2) = 0.73.

The rationale for this formula is that the length of a path has direct influence on the similarity score. That means, the similarity score for a short path will be better than for a longer one. We need an appropriate algorithm for computing shortest path with best similarity.

3.2 Automatic Building of Ontology Indexes

For building an ontology index from scratch we start with a top down method (specialization) according to the given top term “entity”. For the subsequent insertion of nodes and for the corresponding update process we use a hybrid method. That means, we determine a set of related terms to a given term and uses specialization method (top down) as well as generalization method (bottom-up) for insertion all nodes.

The following algorithm consists of seven main steps. At the beginning we have an ontology index with one node n=(entity,entity) for the top term “entity”. For illustration consider the given term “chip” and the ontology index as shown in Figure 2.

Step 1: Extracting a term from an XML document during parsing this document.
During parsing of an XML document an appropriate term (noun) will be extracted. Each element name and each important keyword of an element content (e.g., each noun) is an appropriate term for insertion into the OI. For extracting keywords from element content we are using stopword elimination and statistical information about term frequencies.
Step 2: Finding categories and related terms to the term given by step 1 with the help of an existing thesaurus like WordNet.
For the term “chip” we obtain several meanings. For each meaning we obtain following categories according to the 2-level hypernyms provided by WordNet for the given term according to each sense:

i) fecal matter ii) dish iii) game equipment iv) equipment

This way we have four *basic nodes*, e.g n1=(chip, fecal matter). Now we are able to compute *related nodes* for each basic node using WordNet. In our example we obtain a set of related terms for the given term “chip”, e.g.:

	i) fecal matter	ii) dish	iii) game equipment	iv) equipment
synonyms:	cow chip,…	potato chip,…	poker chip,…	microchip,…
hypernyms:	dung,…	snack food,...	counter,...	elect. equip.,...
…				

Step 3: Computing the weights for two given terms using standard web search engines like Google, Yahoo, Altavista, etc.

For a given term and one of its related terms we use a standard web search engine to get a weight for the similarity of these two terms. For example for the two terms "chip" and "microprocessor" of the basic node n4=(chip,equipment) and the related node m=(microprocessor,equipment) we obtain the following occurrences using Google:

basic node n4:	+chip +equipment	N1=923.000
related node m:	+microprocessor +equipment	N2=203.00
corelation of the nodes n4 and m:	+chip +microprocessor +equipment	N3=71.100

The weight is computed by N3/(N1+N2–N3)=0.067. Because of very small values we normalize against the up to this point largest value as 98%.

Step 4: Insert a basic node and its related nodes into the ontology index

Let n=(t,c) the node to be inserted by a method *insert(n)*. We create a new node n, if n does not exist and the following case fails. If there exists a node k=(t,c1) with a category c1 different to the given category c, then we have to adjust the relationship between c and c1. If c (c1) is a broader term of c1 (c) we define c (c1) as the new category of node k.

In our example we try to insert the following basic nodes n1, n2, n3, n4 and their corresponding related nodes m containing related terms as computed in step 2:

n1=(chip,fecal matter)	m=(cow chip,fecal matter)	with r=m is synonym of n1
	m=(dung,fecal matter)	with r=m is hypernym of n1

…

Step 5: Create edges between a basic node and its related nodes

Let n'=insert(n) the basic node after insertion of basic node n into the ontology index. The following cases describe the creation of a new edge for each type of relationship defined in definition 1.

For all m with r=synonym or r=hypernym or r=holonym (e.g. m is synonym of n' (n), etc.) we have to create a new edge of the given type. Let m' the related node after insertion of related node m into ontology index, then E = E + {e=(n',m',is_[r]_of,1.0)}.

For all m with r=hyponym or r=meronym (e.g. m is hyponym of n' (n), etc.) we have to create a new edge of the inverse relationship r'=hypernym or r'=holonym (e.g. n'(n) is hypernym of m, etc) according to the given relationship. Let m' the related node after insertion of related node m into ontology index, then E=E+{e=(n',m',is_[r']_of,sim(n',m')}

Step 6: Adjust the weights of new edges and existing edges

Let x a basic node, y a corresponding related node and e a new edge between x and y. Let z a parent node (child node) of x and y where z – x and z – y have the same relationship type.

First, we assume x is a synonym of y and z is a hypernym (holonym, hyponym, meronym) of x and y. Then we define sim(x,z) = sim(y,z) = max{sim(x,z), sim(y,z)}.

Second, we assume x is a hypernym of y. If z is a hypernym (holonym) of x and y, then we define sim(z,y) = sim(z,x)*sim(x,y). If z is a hyponym (meronym) of x and y, then we define sim(x,z) = sim(x,y)*sim(y,z).

The cases "x is a holonym (hyponym, meronym) of y" are defined analogously.

Step 7: Connectivity

If step 5 creates no new edge between an existing node and a new node then it is necessary to create a hypernym-edge from the root node to one of the basic nodes and its do-

main node.

This algorithm based on a WordNet-style ontology as source gives two main guarantees. (1) There are no isolated nodes (strong connectivity of the graph). (2) A long path between two nodes v and w can not be more relevant than a short path between v and w.

4 Evaluation of Semantic Similarity Search Conditions using Ontology Indexes

In the XXL Search Engine we are using two ontology indexes, one over names of XML elements and XML attributes called NOI (element name ontology index) and a second over keywords extracted from the contents of XML elements called COI (content ontology index). In addition, there are an element path index (EPI) for evaluating path expressions and an element content index (ECI) for evaluating element content conditions.

To find similar terms for a given search term within an ontology index, we proceed two steps. First, we are looking for the node k containing the search term. This takes time O(log n) where n is the number of nodes. Second, we traverse the ontology index starting at the node k in a breadth-first manner. A term of a node w is similar to the given term of node k, if there is a path between w and k which either contains only edges of types hypernym and synonym or contains only edges of types hyponym and synonym. Crossing the root node is not allowed, because this would change the domain. The second step takes time O(n).

The worst-case runtime for this algorithm is O(n log n). The search for related terms using the ontology index can be restricted by the number of edges followed in one breadth-first search process or by a threshold for the similarity score of the related terms found in the ontology.

A relevance computation for a query result is described in detail in [TW02]. In this section we only present the application of the index structure to evaluate an *elementary similarity path expression* and an *elementary similarity content condition* within an XXL query.

Case I: elementary similarity path expression within an XXL query

A subquery which is interested in all documents containing an element named "chair" or a related element name can be expressed as Expr1: "~chair".

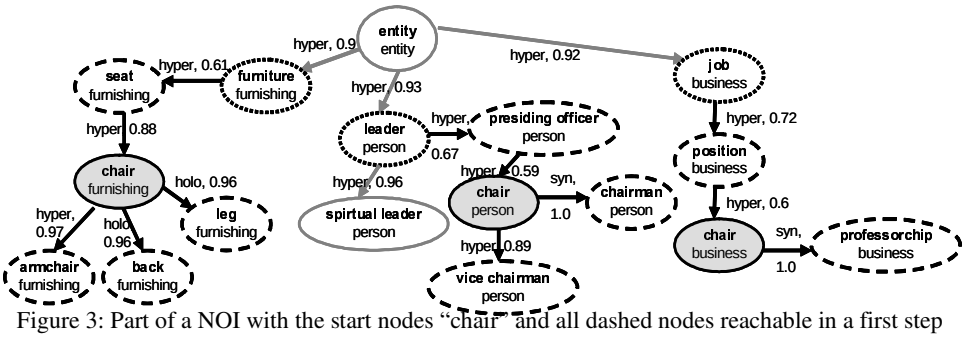

Figure 3: Part of a NOI with the start nodes "chair" and all dashed nodes reachable in a first step and all dotted nodes reachable in a second step of the breadth-first search algorithm

Consider the graph shown in Figure 3 as part of a NOI with the corresponding relationships and fictitious weights.
In a first step we compute all related terms using the NOI to the given term "chair". All similarity scores are computed as defined in Definition 4 in section 3.1. The algorithm provides the following information:

- start nodes: chair (furnishing): 1.0; chair (person): 1.0; chair (business): 1.0
- first step: armchair (furnishing): 0.97; back (furnishing): 0.96; leg (furnishing): 0.96,…
- second step: furniture (furnishing): 0.56; job (business): 0.49; leader (person): 0.48; …

In a second step we group these nodes by their domains. According to the NOI of Figure 3 we have three children of the root node (entity,entity), namely (furniture,furnishing), (leader,person) and (job,business). Therefore in our example we have information of three different domains.

- furnishing: chair 1.0; armchair 0.97; …
- person: chair 1.0; chairman 1.0; …
- business: chair 1.0; position 0.6; …

In the third step we create for each domain a new elementary query according to the given related terms and its similarities provided by the NOI.

- Expr1': "chair|armchair|back|leg|seat|furniture"
- Expr1'': "chair|chairman|presiding officer|vice chairman|leader"
- Expr1''': "chair|position|professorship|job"

In the last step we use the element path index (EPI) and the similarity scores from the first step to compute a set of result nodes and its local relevance values for each new domain-oriented elementary query Expr1', Expr1'', and Expr1'''.

Case II: elementary similarity content condition
A subquery wich is interested in all elements named "title" with content about algorithms for XML written in Java and not written in C++ can be expressed by this expression:

Expr2: "title ~ '(java or xml) and algorithm and not c++'"

In the first step we decompose the complex content condition into a binary *LogicTree* where an inner node contains an operator and a leaf node contains a search string.
In the next step for each search string (leaf node) we call the content ontology index (COI) for related terms and their categories. Then, analogously to step 1 and step 2 of case I above we group them by their domain. In the fourth step we consider all terms of one domain for domain-oriented search. For each search string we combine its related terms in a disjunctive manner. For each leaf we use the element content index (ECI) implemented in Oracle9i and its Contains-function to find appropriate elements. For the relevance computation for one found element we multiply the COI-based similarity score and the ECI-based tf*idf-based similarity score. Finally, we evaluate the LogicTree.

5 Conclusion

Currently ontologies considered as shared conceptualizations of some domain are increasingly seen as the key to further automation of information processing. Although many approaches for representing and applying ontologies have already been devised, they have not found their way into search engines for querying XML data. In this paper we have shown that ontologies using categories and differentiated semantic relationships

are useful to improve the similarity search on XML data. Our approach is implemented in the XXL search engine.

6 References

[BEH+02] E. Bozsak, et. al.: KAON - Towards a large scale Semantic Web. In: Proc. of EC-Web 2002. LNCS, Springer, 2002.

[BHP94] M.W. Bright, A.R. Hurson, S. Pakzad: Automated Resolution of Semantic Heterogeneity in Multidatabases. ACM Transactions on Database Systems, 19(2) 1994, pp. 212-253.

[BM99] T. Bench-Capon, G. Malcolm: Formalising Ontologies and Their Relations. Proc. of the 10th DEXA Conf. 1999, pp: 250-259.

[BR99] R. Baeza-Yates, B. Ribeiro-Neto: Modern Information Retrieval, Addison Wesley, 1999.

[DEFS99] S. Decker, M. Erdmann, D. Fensel, R. Studer: OntoBroker: Ontology based Access to Distributed and Semi-Structured Information. Semantic Issues in Multimedia Systems, Proc. of DS-8, Kluwer, 1999, pp. 351-369.

[FAF+02] R. Feldman, Y. Aumann, M. Finkelstein-Landau, E. Hurvitz, Y. Regev, A. Yaroshevich: A Comparative Study of Information Extraction Strategies. Proc. of the CICLing Conf. 2002.

[FDH98] R. Feldman, I. Dagan, H. Hirsh: Mining Text Using Keyword Distributions. Journal of Intelligent Information Systems 10 (1998), pp. 281-300.

[Gru93] T.R. Gruber: Towards Principles for the Design of Ontologies used for Knowledge Sharing. Proc. of the International Workshop on Formal Ontology, 1993.

[Gua98] N. Guarino: Formal Ontology in Information Systems. Proc. of FOIS'98, IOS Press.

[Hor02] I. Horrocks: DAML+OIL: A Reasonable Web Ontology Language. Proc. of the 8th EDBT Conf. 2002, pp. 2-13.

[KI90] H. Kimoto, T. Iwadera: Construction of a Dynamic Thesaurus and its Use for Associated Information Retrieval. Proc. of the SIGIR Conf. 1990, pp. 227-240.

[KKC94] O. Kwon, M.-C. Kim, K.-S. Choi: Query Expansion Using Domain Adapted, Weighted Thesaurus in an Extended Boolean Model. Proc. of the CIKM Conf. 1994, pp. 140-146.

[KKS01] L. Kerschberg, W. Kim, A. Scime: A Semantic Taxonomy-Based Personalizable Meta-Search Agent. Proc. of the WISE Conf. 2001.

[MWK00] P. Mitra, G. Wiederhold, M.L. Kersten: Articulation of Ontology Interdependencies Using a Graph-Oriented Approach, Proc. of the 7th EDBT Conf., Constance, Germany, 2000.

[PM98] M.P. Papazoglou, S. Milliner: Subject-based Organization of the Information Space in Multi-Database Networks. Proc. of the 10th CAiSE Conf. 1998.

[RN95] S. Russel, P. Norvig: Artificial Intelligence - A Modern Approach, Prentice Hall, 1995

[SAD+00] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Mädche, H.-P. Schnurr, R. Studer: Semantic Community Web Portals, 9th WWW Conference, 2000.

[SemWeb] World Wide Web Consortium: Semantic Web Activity, http://www.w3.org/2001/sw/

[SM00] S. Staab, A. Maedche: Ontology Engineering beyond the Modeling of Concepts and Relations. Proc. of the 14th ECAI Conf., Workshop on Applications of Ontologies and Problem-Solving Methods, 2000.

[STW01] S. Sizov, A. Theobald, G. Weikum: Ähnlichkeitssuche auf XML-Daten. In: A. Heuer, F. Leymann, D. Priebe (Eds.): Datenbanksystem in Büro, Technik und Wissenschaft (BTW) 2001, 9. GI-Fachtagung, Oldenburg, In: Informatik aktuell, pp 364-383, Springer Verlag, 2001.

[TW00] A. Theobald, G. Weikum: Adding Relevance to XML. In: D. Suciu, G. Vossen (Eds.): The World Wide Web and Databases. Lecture Notes in Computer Science 1997, pp 105-124, Berlin: Springer, 2001.

[TW02] A. Theobald, G. Weikum: The Index-based XXL Search Engine for Querying XML Data with Relevance Ranking. In: Ch. S. Jensen, K. G. Jeffery, J. Pokorny, S. Saltenis, E. Bertino, K. Böhm, M. Jarke (Eds.): EDBT Conf. 2002. LNCS 2287, Berlin: Springer, 2002.

[WN98] C. Fellbaum (ed.): WordNet: An Electronic Lexical Database. MIT Press 1998.