Using interoperating conceptual tools to improve searches in Kaj Munk

Ulrik Petersen

Department of Communication and Psychology
Kroghstræde 3
DK – 9220 Aalborg East
Denmark
ulrikp@hum.aau.dk
http://www.kajmunk.hum.aau.dk

Abstract. Query Expansion is a technique whereby a query in an information retrieval system is expanded with more terms, thus most likely increasing the number of relevant documents retrieved. In this paper, we describe a prototype of a system built around a number of interoperating conceptual structures tools, and how it uses Query Expansion to retrieve greater numbers of relevant documents. Thus the system is an example of how interoperating conceptual structures tools can be used to implement an information retrieval system.

1 Introduction

In what ways is it possible to query a corpus of natural language text conceptually? That is the motivating question behind the research presented in this paper. In order to answer this question partially, we have built a prototype system which incorporates three technologies, namely the Amine Platform [1–6], the Emdros corpus query system [7–10], and some natural language processing software in the form of a lemmatizer and a part of speech tagger[1]. The system is able to increase the recall of queries for a given corpus of text, by expanding the query with lemmas taken from an Amine ontology. The system could not have been built without the integration of the three key technologies mentioned. In this paper, we show how the system works in terms of its architecture, and how it is able to achieve greater recall.

The organizational context of the present research is the Kaj Munk Research Centre at Aalborg University, Denmark. Kaj Munk (1898-1944) was a Danish playwright, pastor, poet, and author, who was very influential both in Danish cultural life and outside of Denmark in the period between the two World Wars. He was killed by the Germans in 1944 for his resistance stance.

The Kaj Munk Research Centre has bought the *nachlass* of Kaj Munk, and is in the process of digitizing the material for electronic publication on the web and

[1] The lemmatizer and part of speech tagger employed in this research are the ones developed by Centre for Language Technology (CST), Copenhagen, Denmark. See http://www.cst.dk.

in other ways. The envisaged website will feature advanced search capabilities that go beyond mere matching of text strings into the realm of semantics. In this endeavour, conceptual structures play a key role, and software tools that deal with conceptual structures become critical in the development of the underlying database technology.

The rest of the paper is laid out as follows. First, we introduce the literature behind our system. Second, we give an overview of our system. Third, we offer a more detailed look at the query-process that leads to semantic querying. Fourth, we give an example of the query process. Fifth, we give an analysis of the functionality in terms of the precision and recall of the system. Sixth, we report on the method of achieving interoperability between the various parts of the system. Finally, we conclude the paper.

2 Literature review

Within the field of information retrieval, the notions of precision and recall are often used to describe how well a search system performs. Briefly, recall is a percentage showing how many documents out of all relevant documents were retrieved, while precision is a percentage showing how many of the retrieved documents are in fact relevant. For more information, see [11] and [12].

Query Expansion refers to a class of techniques in Information Retrieval in which a query given by the user is expanded with more query terms. The intent is always either to increase the recall, or to increase the precision, or both. Query Expansion is an old technique, but as demonstrated by the literature, a very useful technique. See for example, [13–15]. In so far as WordNet [16] can be considered an ontology, [13, 17, 15] among many others show that ontologies can prove valuable in the process of Query Expansion. The article [18] shows how compound forms in Danish can be split into their respective lemmas, then used as a basis for Query Expansion using a thesaurus.

The present author has built a corpus query system called Emdros. The Emdros software is a generic query system for "analyzed or annotated text." As such, the software accommodates "text plus information about that text."[2] In the present research, Emdros is the component that stores and queries both the text corpus to be queried and the part of speech and lemma information with which each token is annotated. Additional information such as sentence boundaries, paragraph boundaries, and noun phrase boundaries are also present, but are not used in the present research. Document boundaries, however, are used.

Emdros was written in C++, but has language bindings for several programming languages including Java. These language bindings are provided through SWIG.[3] For more information on Emdros, the reader is invited to consult both

[2] This phrase is taken from [19], which is the PhD thesis of Crist-Jan Doedens. Emdros implements an extension of Doedens' database model, and a subset of Doedens' query language. As such, Emdros can be seen as a derivate of the labours of Dr. Doedens.

[3] See `http://www.swig.org`. See also [20].

the Emdros website[4] and [7–10], all of which can be downloaded from the author's website[5].

The Amine Platform is a platform intended for development of intelligent systems and multi-agent systems [5]. It implements a large array of the technology components needed for building Knowledge Systems, including an ontology builder, a CG layer with concomitant CG operations, and a logic inference engine built around the integration of Prolog and CGs.[6] The latter component is called Prolog+CG, and is the software hub in the prototype which we have developed.

3 System Overview

An overview of the system is given in Fig. 1. It is given in the form of a conceptual graph, with an implied ontology of concept types such as the one given in Fig. 2, and an implied relation hierarchy such as the one given in Fig. 3.

As can be seen from the ontology of concept types in Fig. 2, there are essentially two kinds of concepts in Fig. 1: Software and Data. Indeed, the relation types reflect this, as can be seen in Fig. 3, in which all subtypes of DataSoftwareRole have the signature (Data,Software), and all subtypes of SoftwareSoftwareRole have the signature (Software,Software). Consequently, the signature of Role in our small conceptual graph of Fig. 1 must be (Bits,Software), indicating that the outgoing arrow on every relation always is attached to a concept of type Software (cf. [21–24]).

There are three related but distinct flows of data in Fig. 1. The first flow starts with the TextCorpus at the left edge of the leftmost row. This TextCorpus (which, in our case, is the corpus of published sermons of Kaj Munk) is read by CST's part of speech tagger and lemmatizer to produce a pos-tagged and lemmatized corpus. This corpus is then read by a program which converts the corpus to a number of `CREATE OBJECT` statements in the MQL query language of Emdros. This produces the MQLCorpus, which is read by Emdros to produce the EmdrosDatabase at the bottom right hand corner of Fig. 1.

The second flow starts with the Amine Ontology Builder in the middle of the second row of Fig. 1, in which a domain expert creates an ontology of the domain which one would like to query. This produces the AmineOntology, which again is read by Amine's Prolog+CG engine. Notice that the method of production of the ontology is irrelevant for our prototype: It might just as well have been produced automatically. In our case, for simplicity and accuracy, we produced our own ontology "by hand."

The third, and main, flow starts with the user query in the top right hand corner of Fig. 1. This query is read by the Prolog+CG programs written for our

[4] `http://emdros.org`

[5] `http://ulrikp.org`

[6] However, Amine is much more than the few components listed here. Interested readers are invited to consult [5, 6] and the Amine Website: `http://amine-platform.sourceforge.net`

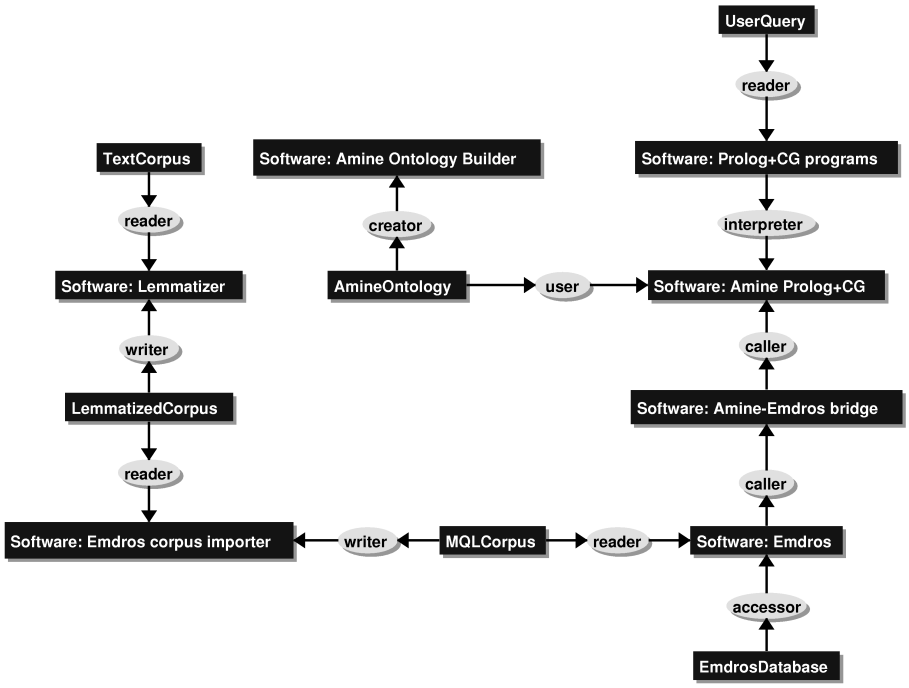

Fig. 1. Overview of our system

prototype, and is transformed, inside of Prolog, to an Emdros query based on the type(s) from the AmineOntology given in the query. This Emdros query is then fed to the Amine-Emdros bridge (through the interpreter-nature of Amine Prolog+CG), which takes care of calling Emdros to query the EmdrosDatabase in the bottom right hand corner of Fig. 1. An answer comes back from Emdros to the Amine-Emdros bridge. This answer is then further processed by the Amine-Emdros bridge in order to obtain not only the words found, but also their context.[7] This result is then passed back to the Prolog+CG programs (through the interpreter-nature of Amine Prolog+CG), which then displays the results to the user.

[7] Emdros's query language is designed to be very generic. As such, it uses a generic method of returning query results, which must then be interpreted in the context of a given database. This interpretation usually involves retrieval of more objects from the database, such as whole sentences and their constituent tokens, and/or the titles of the document(s) in which the results are found.

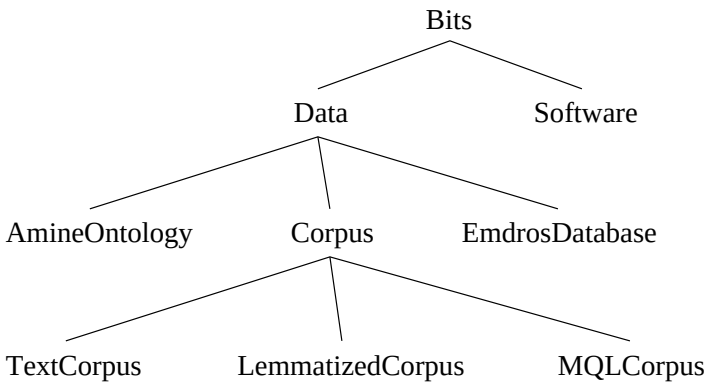

Fig. 2. A possible ontology for the concept types in our system

4 Querying

In our prototype, the user enters a query in the form of a set Q of types from the Amine ontology, along with an indication of whether supertypes or subtypes are wanted. If supertypes are wanted, a set E_i are constructed containing n levels of supertypes of each term t_i from Q. Similarly for subtypes, if subtypes are wanted.

An Emdros query is then constructed from the sets E_i, as follows. For each set E_i, a single query fragment (technically, an "object block" retrieving an object with certain characteristics) is created, finding all objects of type `token` whose lemma is one of the terms e_j in E_i, with Boolean disjunction being the operator between the comparison. This could look as follows:

```
[token lemma='party' or lemma='organization' or lemma='group']
```

If there is more than one set E_i, then all permutations of all sequential orders of the query fragments arising from each E_i is constructed, allowing for arbitrary space in between each query fragment, and using an OR construct on strings of blocks between each permutation. This results in a string of OR-separated strings of blocks, where each string of blocks represents one possible order of the query terms. Finally, this string of OR-separated strings is wrapped in a block indicating that the search is to be done within document boundaries.

Variations over this theme abound. For example, the number of levels n to go up or down in the ontology can be varied; sibling nodes may be considered; various measures of semantic distance may be employed in determining which concepts to include in the search; word-sense disambiguation may be performed based on either the query terms and their cooccurrence in the query, or on the documents actually stored in the database, or both; the context may be changed from "Document" to a more restrictive "paragraph" or even "sentence" textual unit, thus increasing the likelihood that the query terms do in fact have

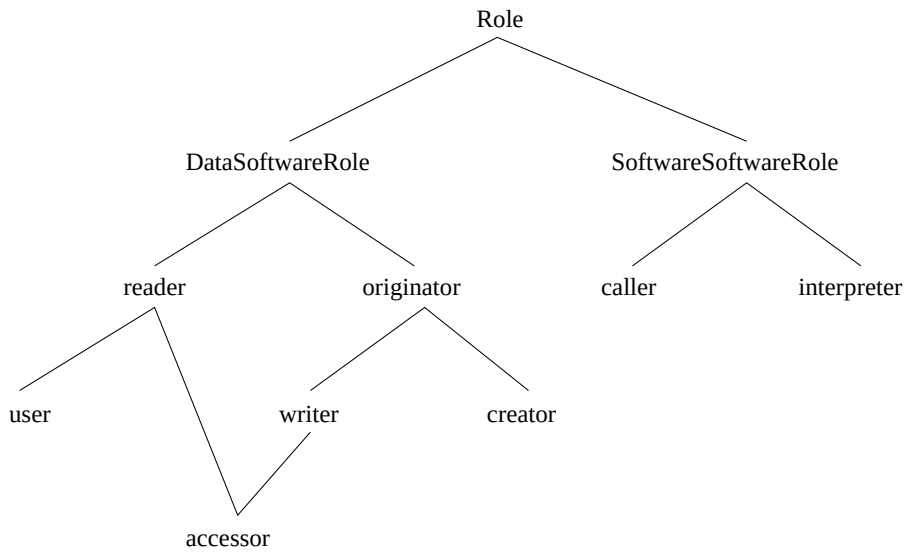

Fig. 3. A possible ontology for the relation types in our system

something to do with each other; named entity recognition may be performed, and named entities may be classified as per the ontology, thus aiding in increasing recall; compounds may be split and used as the basis of further Query Expansion, as described in [18]; parsing or chunking of the texts may be performed so as to aid in identifying noun phrases that could aid in identifying more precisely where to search for given kinds of entries from the ontology; the ontology may be enriched with part of speech information, such that this information can be taken into account when searching. Many other techniques have been tried over the years, all built up around the single, simple idea of Query Expansion.

5 Query Example

In this section, we give an example of how the query-process works.

Consider the small ontology in Fig. 4. It is a sample ontology of concepts from the political domain. Some of the concepts are annotated underneath with zero or more lemmas, separated by a vertical bar if more than one are present. Where no lemma corresponds to the type, the number of lemmas is zero.

Suppose the user enters a query in which the set Q of query types is { PartyMember, PoliticalEmployee }, and suppose that the user specifies that 3 levels of subtypes are to be used for query expansion. In this case, two sets $E_0 =$ { partymember, minister, primeminister, MP, parliamentmember } and $E_1 =$ { spindoctor } are constructed.

From these sets, the two object blocks:

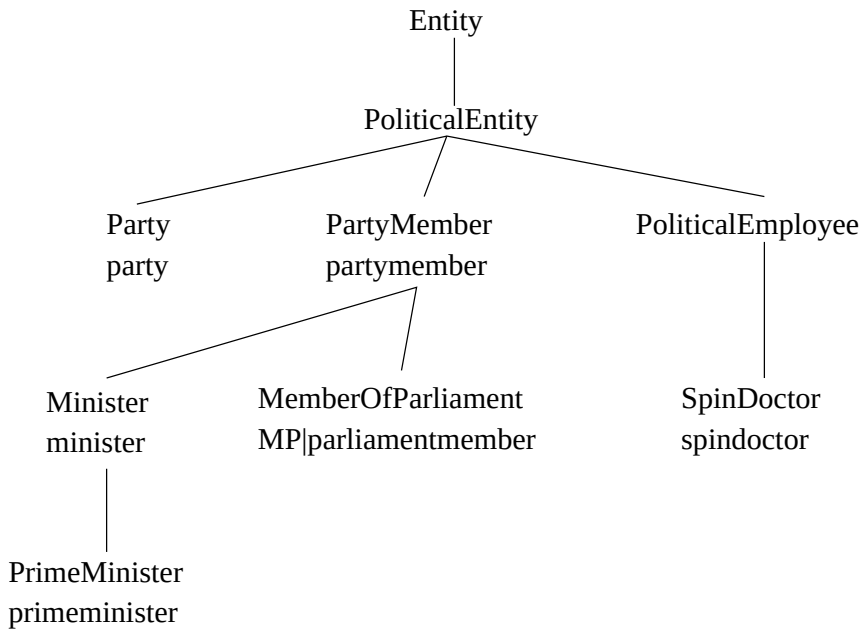

Fig. 4. A sample domain ontology of political language

```
[token lemma='partymember' OR lemma='minister'
       OR lemma='primeminister' OR lemma='MP'
       OR lemma='parliamentmember']
```

and

```
[token lemma='spindoctor']
```

are constructed. There are only two possible permutations of the order in which two objects can occur (2 factorial), so these object blocks give rise to the query shown in Fig. 5.

Briefly, the query means that, within the context of a document, two tokens must be found, in either order, where the lemma of each token is either drawn from the sets E_0 and E_1. The "`..`" between each `[token]` object block means that the tokens need not be adjacent, but may be separated by arbitrary space, within the scope of the context Document.

This query is executed by the Amine-Emdros bridge, and the results post-processed in order to get the context of the "hits", to be shown to the user by Prolog+CG.

```
[Document
  [token lemma='partymember' OR lemma='minister'
         OR lemma='primeminister' OR lemma='MP'
         OR lemma='parliamentmember'
  ]
  ..
  [token lemma='spindoctor']

  OR

  // Now the other order is tried...
  [token lemma='spindoctor']
  ..
  [token lemma='partymember' OR lemma='minister'
         OR lemma='primeminister' OR lemma='MP'
         OR lemma='parliamentmember'
  ]
]
```

Fig. 5. Example Emdros query

6 Precision and Recall

As mentioned in Sect. 2, "recall" is a measure used within Information Retrieval to describe how well a system performs; in particular, it shows how many documents were retrieved, divided by the total number of relevant documents for any given query. "Precision," on the other hand, is the number of relevant documents returned, divided by the number of documents returned [12].

As confirmed by the research reported in [13–15, 17], our system improves recall, and for the same reason that any Query Expansion technique in general improves recall: Since semantically similar terms are added to the query, more documents that contain semantically similar terms will be found. Since relevant documents may contain terms semantically similar to the original query terms, yet may not contain the actual query terms, increasing the number of documents retrieved with semantically similar terms will most likely increase recall.

We have not evaluated our system formally on either precision or recall measures, but this is something for future research.

7 Interoperability

This being a practical rather than theoretical paper, a number of comments on the interoperability of the various system components are in order.

Both Amine's ontology builder, Amine's Prolog+CG, and Emdros can be viewed as tools for dealing with conceptual structures; Amine's tools more so than Emdros. Amine's treatment of conceptual structures goes right to the core

of the very purpose for which Amine was created [5, 6]; thus a large part of Amine's codebase is centered around conceptual structures. Emdros, on the other hand, has a different focus, namely that of storage and retrieval of annotated text. Given that lemmas represent a unified form for all forms of a given word, and given that this simplifies the task of assigning meaning to any given word, and given that lemmas play an important role in the selection of labels for the concept types in many kinds of ontologies, and given that Emdros can store lemmas just as well as any other annotation, Emdros can be seen to be able to deal with conceptual structures.

The interoperability of Amine with Emdros was achieved through the use of a "bridge" written in Java. This bridge is simply a Java class which instantiates a connection to an Emdros datbase, receives queries, and "harvests" the results. The latter task involves processing the results of a query, then retrieving as much context as necessary for the purposes at hand. This usually involves things like retrieving document titles, all the words of the sentence surrounding a "hit", retrieval of other objects necessary for display of the hits, etc.

Amine's Prolog+CG supports calling arbitrary Java methods and instantiating arbitrary Java objects from within Prolog+CG. This is the method used in our prototype system, where Prolog+CG instantiates an "Amine-Emdros bridge" as a Java object, then calls methods on this bridge both to retrieve query results and to postprocess them as described above.

The present author found that Amine's Java-integration made it easy to call both the Amine API and the Emdros bridge. The ability to call arbitrary methods in the host language (Java, in this case) was key in making the interoperability work.

8 Conclusion

We have described a prototype system that enables a user to query a collection of documents semantically rather than just by keyword. This is done through the use of three key technologies: The Amine Platform, the Emdros Corpus Query System, and a lemmatizer and part of speech tagger for the target language. An ontology is used to guide the process of query expansion, leading to a greater number of relevant documents being returned than would have been the case, had the program only found documents containing the original query terms.

Pointers to further research have already been given.

Acknowledgements

Thanks are due to cand.scient. Jørgen Albretsen, who provided the ontology used in this prototype. Prof. dr.scient., PhD Peter Øhrstrøm provided many of the research ideas used in this research. The Danish Centre for Language Technology (CST) provided the part of speech tagger and lemmatizer used. Figure 1 was

drawn with the CharGer software written by Harry Delugach.[8] The SWIG team[9] made the integration of Emdros with Java possible. Finally, many thanks to Prof. Dr. Adil Kabbaj, who wrote the Amine-Platform, without which this research would have been much more difficult to carry out.

References

1. Kabbaj, A., Frasson, C., Kaltenbach, M., Djamen, J.Y.: A conceptual and contextual object-oriented logic programming: The PROLOG++ language. In Tepfenhart, W.M., Dick, J.P., Sowa, J.F., eds.: Conceptual Structures: Current Practices – Second International Conference on Conceptual Structures, ICCS'94, College Park, Maryland, USA, August 1994, Proceedings. Volume 835 of Lecture Notes in Artificial Intelligence (LNAI)., Berlin, Springer Verlag (1994) 251–274
2. Kabbaj, A.: Un systeme multi-paradigme pour la manipulation des connaissances utilisant la theorie des graphes conceptuels. PhD thesis, Univ. De Montreal, Canada (1996)
3. Kabbaj, A., Janta-Polczynski, M.: From PROLOG++ to PROLOG+CG : A CG object-oriented logic programming language. In Ganter, B., Mineau, G.W., eds.: Proceedings of ICCS 2000. Volume 1867 of Lecture Notes in Artificial Intelligence (LNAI)., Berlin, Springer Verlag (2000) 540–554
4. Kabbaj, A., Moulin, B., Gancet, J., Nadeau, D., Rouleau, O.: Uses, improvements, and extensions of Prolog+CG : Case studies. In Delugach, H., Stumme, G., eds.: Conceptual Structures: 9th International Conference on Conceptual Structures, ICCS 2001, Stanford, CA, USA, July/August 2001, Proceedings. Volume 2120 of Lecture Notes in Artificial Intelligence (LNAI)., Berlin, Springer Verlag (2001) 346–359
5. Kabbaj, A.: Development of intelligent systems and multi-agents systems with amine platform. [25] 286–299
6. Kabbaj, A., Bouzouba, K., El Hachimi, K., Ourdani, N.: Ontologies in Amine Platform: Structures and processes. [25] 300–313
7. Petersen, U.: Emdros — a text database engine for analyzed or annotated text. In: Proceedings of COLING 2004. (2004) 1190–1193 http://emdros.org/petersen-emdros-COLING-2004.pdf.
8. Petersen, U.: Evaluating corpus query systems on functionality and speed: TIGERSearch and Emdros. In Angelova, G., Bontcheva, K., Mitkov, R., Nicolov, N., Nikolov, N., eds.: International Conference Recent Advances in Natural Language Processing 2005, Proceedings, Borovets, Bulgaria, 21-23 September 2005, Shoumen, Bulgaria, INCOMA Ltd. (2005) 387–391
9. Petersen, U.: Principles, implementation strategies, and evaluation of a corpus query system. In: Proceedings of the FSMNLP 2005. Volume 4002 of Lecture Notes in Artifical Intelligence., Berlin, Heidelberg, New York, Springer Verlag (2006)
10. Petersen, U.: Querying both parallel and treebank corpora: Evaluation of a corpus query system. In: Proceedings of LREC 2006. (2006) Available as http://ulrikp.org/pdf/LREC2006.pdf.
11. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley (1999)

[8] `http://charger.sourceforge.net`
[9] `http://www.swig.org`, led by David Beazley.

12. Frakes, W.B., Baeza-Yates, R.: Information Retrieval: Data Structures and Algorithms. Prentice Hall (1992)
13. Voorhees, E.M.: Query expansion using lexical-semantic relations. In: SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, Springer-Verlag New York, Inc. (1994) 61–69
14. Mitra, M., Singhal, A., Buckley, C.: Improving automatic query expansion. In: SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (1998) 206–214
15. Moldovan, D.I., Mihalcea, R.: Using WordNet and lexical operators to improve internet searches. IEEE Internet Computing **4**(1) (2000) 34–43
16. Fellbaum, C., ed.: WordNet: An Electronic Lexical Database. MIT Press, London, England and Cambridge, Massachusetts (1998)
17. Smeaton, A.F., Quigley, I.: Experiments on using semantic distances between words in image caption retrieval. In: Research and Development in Information Retrieval. (1996) 174–180
18. Pedersen, B.S.: Using shallow linguistic analysis to improve search on Danish compounds. Nat. Lang. Eng. **13**(1) (2007) 75–90
19. Doedens, C.J.: Text Databases: One Database Model and Several Retrieval Languages. Number 14 in Language and Computers. Editions Rodopi Amsterdam, Amsterdam and Atlanta, GA (1994) ISBN 90-5183-729-1.
20. Beazley, D.M., Fletcher, D., Dumont, D.: Perl extension building with SWIG (1998) Presented at the O'Reilly Perl Conference 2.0, August 17-20, 1998, San Jose, California. Access online 2007-04-22: `http://www.swig.org/papers/Perl98/swigperl.htm`.
21. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading, MA. (1984)
22. Sowa, J.F.: Conceptual graphs summary. In Nagle, T.E., Nagle, J.A., Gerholz, L.L., Eklund, P.W., eds.: Conceptual Structures: Current Research and Practice. Ellis Horwood, New York (1992) 3–51
23. Sowa, J.F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks/Cole Thomson Learning, Pacific Grove, CA (2000)
24. Petersen, U., Schärfe, H., Øhrstrøm, P.: Online course in knowledge representation using conceptual graphs. On the web: `http://www.huminf.aau.dk/cg/` (2001-2007)
25. Henrik Schärfe, Pascal Hitzler, P.Ø., ed.: Conceptual Structures: Inspiration and Application. 14th International Conference on Conceptual Structures, ICCS 2006, Aalborg, Denmark, July 2006, Proceedings. In Henrik Schärfe, Pascal Hitzler, P.Ø., ed.: Conceptual Structures: Inspiration and Application. 14th International Conference on Conceptual Structures, ICCS 2006, Aalborg, Denmark, July 2006, Proceedings. Volume 4068 of Lecture Notes in Artificial Intelligence (LNAI)., Berlin, Heidelberg, Springer-Verlag (2006)