On Querying Ontologies and Databases

Henrik Bulskov, Rasmus Knappe, and Troels Andreasen

Department of Computer Science,
Roskilde University,
P.O. Box 260, DK-4000 Roskilde, Denmark
{bulskov,knappe,troels}@ruc.dk

Abstract. This paper concerns the motivation for and subsequently the analysis of proposed additions, in the form of new operators, to a concept language Ontolog for use in querying a content-based text retrieval system. The expressiveness of the proposed query language introduces the possibility for querying both objects in the base but also mechanisms for direct querying of the ontology, and it furthermore enables the end-user to tailor the evaluation principle of the system by influencing query expansion in the ontology.

1 Introduction

In working with ontology-based information retrieval systems, one important aim is to utilize knowledge from a domain-specific ontology to obtain better and closer answers on a semantical basis, thus to compare concepts rather than words. Within this field our primary focus has been on representation of the ontology, description of the searchable objects, and on how to compare concepts and measure the similarity between them.

In this paper we want to shift our focus in the direction of query formulation and specifically on how to use the description language, Ontolog, to formulate queries. Over the years most users have learned to formulate queries as sets of words, as handled by most search engines. This works very well for many purposes, especially with the new generation of search engines developed during the last couple of years. One major problem with query formulation as sets of words is the limited expressiveness, and therefore the lack of opportunities for rewriting or expanding queries. If the first try does not work, it can be difficult to find alternatives.

In ontology-based information retrieval systems, queries are formulated using concepts, spanning from simple sets of concepts, like the sets of terms used in classical information retrieval, to sets of compound concepts extracted from natural language by some kind of natural language processing.

Even if the query formulation is done using natural language then rewriting, for most people, is extremely difficult without detailed knowledge about the domain and the evaluation principle used by the querying system. One possible solution is for the system to perform query expansion using a ontology-based similarity measure.

Query expansion of this type is typically hidden from the end-user and one could argue that experienced users, domain experts and knowledge engineers have a specific need for more sophisticated control over the query expression in order to query both the text objects in the database, but also to pose queries directly to the structure of the ontology. This facilitates the need for the introduction of a conceptual query language with enhanced expressiveness. We propose to achieve this by extending the concept language ONTOLOG used for description of queries and conceptual indexing of text objects in the database, to form a query language.

The paper is organized as follows. Firstly we introduce and formally define the concept language used for describing the semantics of queries and objects in the database. Secondly we describe the ontology-based similarity measure in use for comparing concepts. And finally we extend the concept language for use as a query language, based on an analysis identifying the need for enhanced expressiveness. Conclusions are given in Section 5.

2 Concept Language

The purpose of the ontology is to define and relate concepts that can be used in descriptions. The ontology framework is generative in the following sense. A basic ontology defines a set of atomic concepts and situates these in a concept inclusion lattice, which basically is a taxonomy over single or multi-word concepts that are treated as atomic in the modelling of the domain. In combination with a given basic ontology, a concept language (description language) defines a set of well-formed concepts.

The concept language in focus here, ONTOLOG[9], defines a set of semantic relations which can be used for "attribution" (feature-attachment) to form compound concepts. The suitable number of available relations may vary with different domains, but among the more general relations that probably will be present in most domain modelings are WRT (With-respect-to), CHR (Characterized-by), CBY (Caused-by), TMP (Temporal), LOC (Location).

Expressions in ONTOLOG are descriptions of concepts situated in an ontology formed by an algebraic lattice with concept inclusion (ISA) as the ordering relation.

Attribution of concepts, combining atomic concepts into compound concepts by attaching attributes, can be written as a feature structures. Simple attribution of a concept c_1 with relation r and a concept c_2 is denoted $c_1[r: c_2]$.

We assume a set of atomic concepts $\mathbf{A}$ and a set of semantic relations $\mathbf{R}$, as indicated with $\mathbf{R}$={WRT, CHR, CBY, TMP, LOC, ...}. Then the set of well-formed concepts $\mathbf{L}$ of the ONTOLOG language is recursively defined as follows.

- if $x \in \mathbf{A}$ then $x \in \mathbf{L}$
- if $x \in \mathbf{L}$, $r_i \in \mathbf{R}$ and $y_i \in \mathbf{L}, i = 1, \dots, n$ then $x[r_1 : y_1, \dots, r_n : y_n] \in \mathbf{L}$

It appears that compound concepts can be built from nesting, for instance $c_1[r_1\colon c_2[r_2\colon c_3]]$ and from multiple attribution as in $c_1[r_1\colon c_2, r_2\colon c_3]$. The attributes of a multiple attributed term $T = x[r_1\colon y_1, \ldots, r_n\colon y_n]$ is considered as a set, thus we can rewrite T with any permutation of $r_1\colon y_1, \ldots, r_n\colon y_n$.

The basis for the ontology is a simple taxonomic concept inclusion relation ISA_{KB}, which is atomic in the sense that it defines a relation over the atomic concepts **A**. It is considered as domain or world knowledge and may for instance express the view of a domain expert. We distinguish this (knowledge base) relation ISA_{KB} because concepts are assumed to be related by specific knowledge over the domain. For that reason we cannot expect the relation ISA_{KB} to be transitively closed or reduced and therefore define the relation ISA as the transitive closure of ISA_{KB} and the relation $\text{ISA}_{\text{REDUC}}$ as the transitive reduction of ISA_{KB}.

Based on ISA, the transitive closure of ISA_{KB}, we can generalize into a relation over all well-formed concepts of the language **L** by the following.

- if x ISA y then $x \le y$
- if $x[\ldots] \le y[\ldots]$ then also

  $x[\ldots, r\colon z] \le y[\ldots]$, and
  $x[\ldots, r\colon z] \le y[\ldots, r\colon z]$,
- if $x \le y$ then also

  $z[\ldots, r\colon x] \le z[\ldots, r\colon y]$

where repeated ... in each inequality denotes identical lists of zero or more attributes of the form $r_i\colon w_i$.

Take as an example the sentence: "*the black dog is making noise*" which can be translated into this semantic expression *noise[*CBY*: dog[*CHR*: black]]*.

Descriptions of text expressed in this language describe semantics and goes beyond simple keyword descriptions. A key question in the framework of querying is of course the definitions of similarity or nearness of terms, now that we no longer can rely on simple matching of keywords.

3 Similarity

For the purpose of devising a ontology-based similarity measure, we can utilize the intuitive notion that similar concepts have much in common, and thereby derive similarity as being proportional to how much concepts share or how close they are in the ontology. In doing so we have to consider computational complexity, as is required by any large scale query evaluation environment. We therefore produce a similarity measure by firstly restricting the possible number of concepts we have to take in consideration when comparing any given pair of concepts, and secondly we derive similarity from "reasoning" within this restricted set of concepts.

3.1 Similarity Graphs

The restriction on the possible number of concepts to be considered, is established by constructing a so-called similarity graph [8] covering the subset of the ontology that contributes to overall similarity between the concepts being compared. A similarity graph can be viewed as a subpart of the ontology represented as a graph with a subset of concepts as nodes and relations connecting these as edges.

We can therefore define a similarity graph for any set of one or more concepts and specifically use this notion as a basis for similarity based on graph computations. The similarity between two concepts can thus be derived from a similarity graph covering these concepts.

To include in the definition of similarity graphs we first define the term-decomposition $\tau(c)$ and the upwards expansion $\omega(c)$ of a concept term c. The term-decomposition is defined as the set of all terms appearing in c. If we for a concept $c = c_0[r_1 : c_1, \ldots, r_n : c_n]$, where c_0 is the atom attributed in c and $c_1, \ldots, c_n$ are the attributes (which are atoms or further compound concepts), define:

$$subterm(c) = \{c_0, c_1, \ldots, c_n\}$$

and straightforwardly extend *subterm* to be defined on a set of concepts $C = \{c_1, \ldots, c_n\}$, such that

$$subterm(C) = \cup_i subterm(c_i)$$

then we can obtain the term-decomposition of c as the closure by subterm, that is, by repeatedly applying *subterm*:

$$\tau(c) = \{c\} \cup \{x | x \in subterm^k(c) \text{ for some k}\}$$

As an example the term $noise[\textsc{cby}\colon dog[\textsc{chr}\colon black]]$ decomposes to the following set of concepts:

$$\tau(noise[\textsc{cby}\colon dog[\textsc{chr}\colon black]]) =$$
$$\{noise[\textsc{cby}\colon dog[\textsc{chr}\colon black]],$$
$$noise, dog[\textsc{chr}\colon black], dog, black\}$$

The upwards expansion $\omega(C)$ of a set of terms C is then the transitive closure of C with respect to $\textsc{isa}_{\text{KB}}$.

$$\omega(C) = \{x | x \in C \vee y \in C, y \text{ ISA } x\}$$

This expansion thus only adds atoms to C.

Now a similarity graph $\gamma(C)$ is defined for a set of concepts $C = \{c_1, \ldots, c_n\}$ as the graph that appears when decomposing C and connecting the resulting set of terms with edges corresponding to the $\textsc{isa}_{\text{KB}}$ relation and to the semantic relations used in attribution of elements in C. We define the triple (x, y, r) as the edge of type r from concept x to concept y.

$$\gamma(C) = \begin{array}{l} \{(x, y, \text{ISA}) | x, y \in \omega(\tau(C)), x \ \text{ISA}_{\text{REDUC}} \ y\} \\ \cup \\ \{(x, y, r) | x, y \in \omega(\tau(C)), r \in \mathbf{R}, x[r\colon y] \in \tau(C)\} \end{array}$$

Figure 1 shows an example of a similarity graph covering two terms.

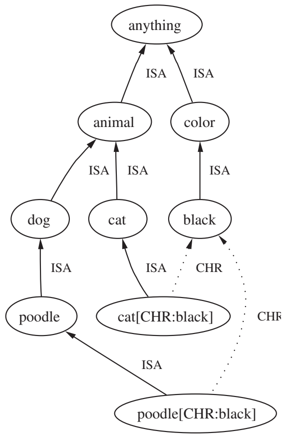

Fig. 1. An example of a similarity graph for the concepts ***cat*[chr: *black*] and *poodle*[chr: *black*]**

The primary purpose of a similarity graph is thus to form the basis for measuring similarity on the restricted set of concepts that contributes to overall similarity between a given set of concepts.

3.2 Evaluation

A time-honored and obvious approach to calculating the similarity between concepts is to reflect what connects them in the ontology. Rada et al. [10] measures similarity as proportional to the number of ordering (ISA) edges between the concepts in question in the ontology. Resnik [11] uses the same taxonomic structure as starting point, but augments the nodes by empirical probability estimates to compensate for non-uniform distance in the ontology, and points out that similarity can be derived using for example another partial (part-of) ordering relation. Another approach, as presented in previous work [6] calculates similarity by using a shortest path, considering both ordering and semantic edges.

As raised in [7,4] the shortest path approach to similarity lacks the influence of an important aspect that has to do with multiple connections between concepts. We may for instance have concepts connected directly through inclusion and in addition through an attribute dimension, as $cat[CHR : black]$ and $poodle[CHR : black]$. Taking all possible paths connecting two concepts x and y

solves this problem, but involves an increase in complexity. If we can reflect the multiple connections phenomenon without traversing all possible paths, we may have a more realistic means of similarity derivation. One option in this direction is to put emphasis on the nodes "shared" by x and y in the similarity graph covering both x and y.

With $\alpha(x)$ [7] as the set of nodes (upwards) reachable from x, we have $\alpha(x) \bigcap \alpha(y)$ as the reachable nodes shared by x and y, which thus obviously is an indication of what's common between x and y.

This notion can be transformed into a normalized measure, as described in [5], by using a set of major properties that improves a given function's accordance with the semantics of the ontology and used these to guide the choice of function.

The selected similarity function, as described in [5] is a weighted average, where where $\rho \in [0, 1]$ determines the degree of influence of both the nodes reachable from both x and y.

$$sim(x, y) = \rho \frac{|\alpha(x) \cap \alpha(y)|}{|\alpha(x)|} + (1 - \rho) \frac{|\alpha(x) \cap \alpha(y)|}{|\alpha(y)|}$$

As illustration consider the subontology in fig. 1. The similarities for $poodle[CHR : black]$ and the other concepts included in the subontology are, when collected in a fuzzy subset of similar concepts (with $similar(x) = \Sigma sim(x, y)/y$) and $\rho = \frac{4}{5}$ the following:

$$similar(poodle[CHR : black]) =$$
$$1.00/poodle[CHR : black] + 0,66/poodle + 0,59/cat[CHR : black] +$$
$$0,54/dog + 0,54/black + 0,43/animal + 0,43/color + 0,36/cat + 0.31/anything$$

The purpose of similarity measures in connection with querying is of course to look for similar rather than for exactly matching values, that is, to introduce soft rather than crisp evaluation.

In addition to the problem of finding a useful measuring principle, a challenge is to device a principle of similarity-based evaluation that is realistic in connection with query processing.

To this end the principle of similarity expansion is an obvious improvement. Instead of calculating similarities in connection with every matching of two values during evaluation, one of these can be expanded and similarity matching becomes a matter of value to set comparison. As indicated through the example above we can introduce similar values by expanding a crisp value into a fuzzy set including also similar values.

4 Concepts, queries and answers

The framework shortly described above and in more detail covered in [5], is initially aimed at a query-answering mechanism where NL-queries are posed to a base of text objects and where ontology-based descriptions, that are expressions in a concept language, are derived through shallow NL parsing and matched by means of ontology-based similarity. In this context concepts, conceptual-based descriptions, ontology, and similarity are (or may be) all hidden from the user who poses a textual query and receives a textual answer. While this may well serve the naive end-users needs it is probably not satisfactory for skilled users such as domain experts, knowledge engineers, and super-users. Typically, skilled users prefer control over the query expression rather than the ease of automatic evaluation from textual expressions. Furthermore, domain and ontology engineers obviously need to pose queries for evaluation against the ontology rather than against the base of text objects.

In other words, there is a need for an optional evaluation of queries against the ontology rather than against the text base, and for a conceptual query language. We discuss below aspects concerning evaluation against the ontology - intensional evaluation - and introduce a conceptual query language.

Obviously, the presented framework can be extended to support the concept language, applied in descriptions, used directly as a query language since queries are transformed into descriptions. A query such as "*large dog*" maps to a description such as "*dog[CHR:large]*" and the latter can as well be used as a query.

4.1 Querying the ontology

An obvious extension to a framework where evaluation of queries to database objects involves interpretation of the query based on an ontology, is a means for querying the ontology concepts directly rather than the database objects. Evaluation of an ontology query can namely be considered as an intermediate step in the evaluation of queries for objects.

The answer to a single concept query [1] can be given an interpretation simply by the function *similar* defined above. So for a given concept ϕ we can set $queryanswer(\phi) = similar(\phi)$, so with the ontology described in figure 1 we have (for a given similarity setting):

$$queryanswer(poodle) = \\ 1.00/poodle + 0,91/poodle[CHR : black] + 0,80/dog + 0,60/animal + \\ +0,53/cat + 0,47/cat[CHR : black] + 0,30/color$$

Aggregation for multiple concept queries can be done in several ways with one simple option being fuzzy union. This topic is further discussed in [1,2,12].

When considering ontology queries, the question is whether an interpretation purely against the generative ontology is the most obvious. For any query the result of the ontology interpretation can be further evaluated in the database whereby the result of the text object interpretation will be obtained. Thus the text object interpretation reflects the database while the ontology interpretation does not.

An alternative to evaluation in the generative ontology is an evaluation in the "extensional ontology" reflecting the database.

[1] A query with a single concept description

We define the extensional ontology as the subontology O_{DB} reflecting the database extension as follows. O_{DB} is the subontology of the generative ontology covering the set of (atomic and compound) concepts used or referred to in the descriptions in the database and any generalizations of these. Thus O_{DB} is a precise reflection of the database content, it is finite and it is non generative.

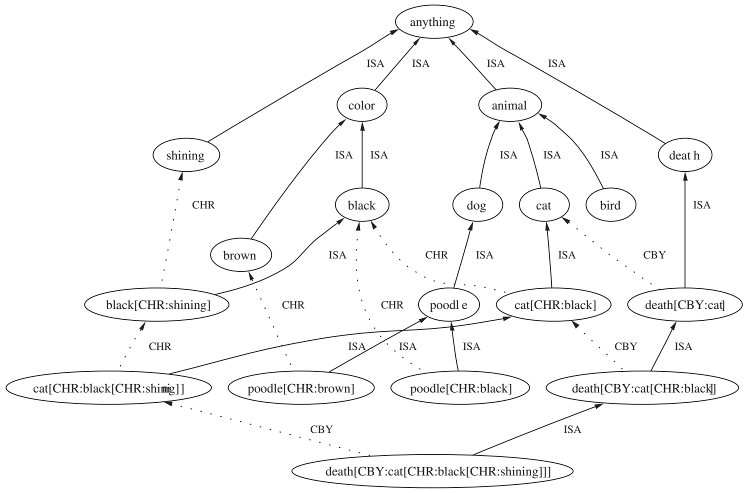

Fig. 2. An example of an extensional ontology for a database with object descriptions listing the concepts ***bird***, ***poodle*****[chr:** ***brown*****]**, ***poodle*****[chr:** ***black*****]** and ***death*****[cby:*****cat*****[chr:** ***black*****[chr:** ***shining*****]]]**

Consider as an example a database consisting of the text objects:

- The two dogs are a black poodle and a brown poodle
- The bird and the death caused by the shining black cat

The descriptions for these objects may be respectively {*poodle*[CHR : *brown*], *poodle*[CHR : *black*]} and {*bird*, *death[*CBY*: cat[*CHR*: black[*CHR*: shining]]]*}. Assuming a set of more general concepts, an example of an extensional ontology corresponding to this database is shown in figure 2.

The advantage of interpreting concepts in the extensional ontology is that the evaluation does not include any concepts not indexed in the base, and we thereby restrict the concepts we have to take into consideration when evaluating queries.

4.2 Querying by description expressions

Another obvious extension to the framework is to allow queries posed directly as expressions in the concept language (using the concept language as query language). Thus in place of an NL-query "Some black cat" to support a description query "*cat[CHR:black]*".

Let us consider ontology queries and assume an extensional ontology interpretation. Thus the answer to a single concept query is a set of concepts appearing in the (descriptions of) objects that are most similar to the query concept.

Take as an example the query Q = "*cat[CHR:black]*" evaluated using $similar_\alpha(Q)$ where α is a threshold limiting the set of similar concepts to those with a membership grade (i.e. similarity) $\geq \alpha$. The query is evaluated in the extensional ontology shown in figure 2, resulting in the following similar concepts.

$$similar_{0.6}(cat[CHR : black]) =$$
$$1.00/cat[CHR : black] + 0.93/cat[CHR : black[CHR : shining]] +$$
$$0.93/death[CBY : cat[CHR : black]] +$$
$$0.89/death[CBY : cat[CHR : black[CHR : shining]]] +$$
$$0.65/poodle[CHR : black] + 0.60/black + 0.60/cat$$

This type of querying may by applicable in cases where the user knows the ontology and the database content well and has a rather specific intention. Without knowledge about the ontology, however, it may be difficult in any case to pose concept queries and likewise with only brief knowledge about the database content posed queries will probably often give unsatisfactory or empty answers.

Posing an NL-query is leaving the control and responsibility for a satisfactory conceptual representation of the query intention to the system. With a concept query the user gains control, but may face problems exploiting this control.

Take as an example the query where the user is interested objects about "colored dogs in general or any specific type of colored dog". This query is problematic because it cannot be expressed using pure concept language.

With special attention to experienced users and domain experts it appears that there is a need for a query language with a broader type of encirclement in query expressions than what can be expressed with pure concept language. For instance a "very large black dog" object will only to some (if any) extent belong to the answer to the query "large pet". The explanation here is the similarity based evaluation of queries, which in turn motivates the introduction of more expressiveness to the query language.

4.3 Query language

Now a concept as a query maps to a set of similar concepts, and similarity is influenced by distance in the ontology. The extension to the concept language introduced here is specialization/generalization operators to cope with a quite useful notation for disjunctions along specialization and/or generalization, and thus avoid reduced similarity over paths of specialization and/or generalization.

Given the concept language **L** based on the set of atomic concepts **A** and the set of semantic relations **R**, as described above, we define an extension of **L** to a query language **QL** as follows.

- $\mathbf{L} \subseteq \mathbf{QL}$
- $* \in \mathbf{QL}$

– if $c \in \mathbf{L}$ then $c_> \in \mathbf{QL}$ and $c_< \in \mathbf{QL}$
– if $c \in \mathbf{QL}$, $r_i \in \mathbf{R}$ and $c_i \in \mathbf{QL}, i = 1, \ldots, n$
  then $c[r_1 : c_1, \ldots, r_n : c_n] \in \mathbf{QL}$

The interpretation of this extended language is the following. $*$ denotes any well-formed concept in $\mathbf{L}$. $c_>$ denotes any specialization of c while $c_<$ denotes any generalization of c. A query involving the operators $<, >$ and $*$ can be considered a disjunctive query over the set of denoted concepts.

With the ontology in figure 2, we have that $dog_<$ denotes all of $\{dog$, $animal$, $anything\}$ and $dog_>$ denotes all of $\{dog, poodle, poodle[CHR : brown], poodle[CHR : black]\}$. The set of denoted concepts for a query is obviously the crisp answer to the query when evaluated in the extensional ontology. Thus a query like "Do we have dogs", with the interpretation "Give me a dog or some specialization of that" can be expressed in the query $dog_>$ and the answer provides a conceptual description of the kinds of dogs that are currently contained in the database without specification of actual dogs and without cardinalities. The reading in the answer is something like "We have poodles here in colors black and brown".

Also with the ontology in figure 2, we have that $cat[CHR : black_<]$ denotes all of $\{cat[CHR : black]$, $cat[CHR : color]$, $cat[CHR : anything]$ while $cat[CHR : black_>]$ denotes $\{cat[CHR : black], cat[CHR : black[CHR : shining]]$.

Concepts that are not part of the extensional ontology such as $animal[CHR : black]$ can of course also be used in queries. We have that $animal[CHR : black_<]$ denotes $\{animal[CHR : black], animal[CHR : color], animal[CHR : anything]\}$ and $animal[CHR : black_>]$ denotes $\{animal[CHR : black], animal[CHR : black[CHR : shining]]\}$.

In some cases the combination of disjunction from expression as above with similarity may confuse the picture. If for instance we want a conceptual answer to $dog_>$ we probably prefer to reveal only what are the different kinds of dogs and have no interest in adding what may be similar to some kind of dogs. On the other hand, if we consider for instance $cat[CHR : black]$ we may prefer an interpretation where specializations of the property *black* are considered equally fulfilling properties so that we get an answer corresponding to

$$similar_{0.6}(cat[CHR : black_>]) =$$
$$1.00/cat[CHR : black] + 1.00/cat[CHR : black[CHR : shining]] +$$
$$0.93/death[CBY : cat[CHR : black]] + 0.89/death[CBY : cat[CHR : black[CHR : shining]]]$$
$$0.65/poodle[CHR : black] + 0.60/black + 0.60/cat$$

One reasonable question related to the introduction of specialization/generalization-queries is to what extend such aspects are already covered by the pure concept language. What is the need of an expression such as $animal[CHR : *]$ to represent "Animal characterized by just anything"[2] when we already can express $animal[CHR : top]$ which basically denotes the same thing. The most important argument for the extension is that we have to cope with side-effects from

[2] notice that $animal[CHR : *]$ and $animal[CHR : Top_>]$ are equivalent

introducing similarity and especially also consider graduate decrease in similarity over longer paths of specialization. We consider *animal* to have *dog* as a more similar concept and $poodle[CHR : brown]$ as less similar concept.

5 Conclusion

One important aspect in working with ontology-based information retrieval systems, concerns the utilization of knowledge from domain-specific ontologies to obtain better and closer answers on a semantical basis.

In this context a major problem concerns query formulation in general concerns, due to the limited expressiveness of the query language, and therefore the lack of opportunities for rewriting or expanding queries. If the first try does not work, it can be difficult to find alternatives. Rewriting of queries is therefore extremely difficult without detailed knowledge about the domain and the evaluation principle used by the querying system. One possible solution is to enhance the control the end-user has over the query language.

We have therefore introduced additions, in the form of new operators, to the concept language Ontolog for use as query language. This has introduced the possibility for querying both objects in the base but also mechanisms for direct querying of the ontology. Experienced users, domain experts, and knowledge engineers can thereby browse the structure the ontology for use in both querying the system, but also for use in ontology engineering and re-construction. The additional operators can furthermore be used to guide the expansion of concepts in the ontology, and can thereby allow the end-user to tailor the evaluation principle to favor more general or more specific answers to a given query. The use of the concept language Ontolog as query language therefore seems to indicate a usable theoretical and practical foundation for querying and browsing of ontology-based querying systems.

Acknowledgments

The work described in this paper is part of the OntoQuery[3][3] project supported by the Danish Technical Research Council and the Danish IT University.

References

1. Andreasen, T.: On knowledge-guided fuzzy aggregation, 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'2002, Annecy, France, July 1-5 , 2002, Proceedings
2. Andreasen, T.: Query evaluation based on domain-specific ontologies. 20th IFSA / NAFIPS International Conference Fuzziness and Soft Computing, NAFIPS'2001, pp. 1844-1849, Vancouver, Canada, 2001, Proceedings

[3] The project has the following participating institutions: University of Copenhagen, The Technical University of Denmark, Copenhagen Business School, and Roskilde University.

3. Andreasen, T., Jensen, P. Anker, Nilsson, J. Fischer, Paggio, P., Pedersen, B. Sandford & Thomsen, H. Erdman: Ontological Extraction of Content for Text Querying, NLDB'2002, Stockholm, Sweden, 2002.
4. Andreasen, T., Bulskov, H. and Knappe, R.: On ontology-based querying, to appear in Eighteenth International Joint Conference on Artificial Intelligence, August 9-15, Acapulco, Mexico, Proceedings
5. Andreasen, T., Bukskov, H., and Knappe, R.: Similarity from Conceptual Relations, pp. 179-184 in Ellen Walker (Eds.): 22nd International Conference of the North American Fuzzy Information Processing Sociaty, NAFIPS 2003, Chicago, Illinois USA, July 24-26, 2003, Proceedings
6. Bulskov, H., Knappe, R. and Andreasen, T.: On Measuring Similarity for Conceptual Querying, LNAI 2522, pp. 100-111 in T. Andreasen, A. Motro, H. Christiansen, H.L. Larsen (Eds.): Flexible Query Answering Systems 5th International Conference, FQAS 2002, Copenhagen, Denmark, October 27-29, 2002, Proceedings
7. Knappe, R., Bulskov, H. and Andreasen, T.: On Similarity Measures for Content-based Querying, International Fuzzy Sysytems Association, World Congress, Istanbul, Turkey, June 29-July 2, 2003, Proceedings
8. Knappe, R., Bulskov, H. and Andreasen, T.: Similarity Graphs, LNAI 2871, pp. 668-672 in N. Zhong, Z.W. Ras, S. Tsumoto, E. Suzuki (Eds.): 14th International Symposium on Methodologies for Intelligent Systems, ISMIS 2003, Maebashi, Japan, October 28-31, 2003, Proceedings
9. Nilsson, J. Fischer: A Logico-algebraic Framework for Ontologies - ONTOLOG, in Jensen, P. Anker & Skadhauge, P. (eds.): Proceedings of the First International OntoQuery Workshop - Ontology-based interpretation of NP's, Department of Business Communication and Information Science, University of Southern Denmark, Kolding, 2001
10. Rada, Roy, Mili, Hafedh, Bicknell, Ellen, Blettner, Maria: Development and Application of a Metric on Semantic Nets, IEEE Transactions on Systems, Man, and Cybernetics, Volume 19, 1989
11. Resnik, Philip: Semantic Similarity in a Taxonomy: An Information-based Measure and its Application to Problems of Ambiguity in Natural Language, Journal of Artificial Intelligence, pp. 95-130, 1999
12. Yager, R.R.: A hierarchical document retrieval language, in Information Retrieval vol 3, Issue 4, Kluwer Academic Publishers pp. 357-377, 2000.