Ontological Profiles in Enterprise Search

Geir Solskinnsbakk and Jon Atle Gulla

Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
{geirsols, jag}@idi.ntnu.no

Abstract. Ontology-driven search applications use ontological concepts either to index documents or to guide and understand the users. Since ontologies by nature are domain-dependent and application-independent, though, there is no guarantee that their concepts are efficient in categorizing and retrieving information from a specific document index. This paper explains the idea of ontological profiles, which is an ontology adapted to the actual language used in a document collection or among the users. A method for constructing ontological profiles from petroleum documents is presented, as well as a search application that makes use of profiles to interpret users queries. Testing on real documents with a 20,000 concepts petroleum ontology reveals that the approach is useful in situations where recall is more critical than precision.

Keywords: ontological profiles, feature vectors, information retrieval, query expansion

1 Introduction

Normal search or information retrieval applications use statistics to calculate similarities between documents and queries and to rank documents with respect to given queries. Documents are viewed as vectors of normalized term frequencies, without paying much attention to what these terms really mean. The approach is often referred to as syntactic search, or morpho-syntactic search if stemming or lemmatization is used on the documents before indexing.

In ontology-driven search applications we attempt to use ontological concepts or structures to address the content -the semantics- of documents and queries. An ontology defines the vocabulary that is later used both in document descriptions and queries. As a consistent and well-defined set of terms is used both to index and retrieve documents, and these terms are semantically related in the ontology, we may better match the users information needs with the real content of the documents. Reasoning with ontological structures may also be applied in cases where there is no direct match between the query and the documents. In practice, ontology-driven or semantic search can be achieved with semantic indices, semantic annotations, query interpretation, or a combination of the above.

Current ontology-driven search applications have however been only moderately successful. Many applications use ontological structures to reformulate queries with more generalized or specialized terms, producing results that are comparable to thesaurus- or dictionary-supported search solutions. Others have tried to encourage users to add semantic annotations to documents, which has led to substantially more manual work and only limited improvement of precision and recall. Even more problematicis the fact that ontology-driven search seems to perform better with tailored ontologies, while ontologies in principle should be application-independent. Application-dependent ontologies is unfortunately also problematic, since they both complicate the idea of application integration and necessitates additional structures or bodies for maintenance and updating.

In this paper we propose an enriched ontology -an ontological profile- that provides an understanding of the ontology in terms of the language used in the document collection. We can generate ontological profiles automatically with text mining techniques, though we may imagine that this profile is gradually adapted to the application with machine learning and user modeling. The profile helps us relate both the document content and the query to the ontological concepts defined, giving us a way to search semantically even if the authors and searchers have no knowledge of the ontology. We argue that the proposed ontological profile is generic to the search process, in the sense that either the ontology or the document collection can be exchanged so that the search platform is usable for other domains. Following this argument, the search platform does not rely on a specific ontology or document collection. Both the indexing process and the search process are unchanged from traditional syntactic search, but a mapping to semantic concepts is done in the background with the ontological profile.

The structure of the paper is as follows: Section 2 gives a brief overview of related work, while Section 3 explains the concept of ontological profiles. In Section 4 we present our method for automatic construction of such profiles, followed by Section 5 which shows how the approach has been used to interpret queries semantically in the petroleum domain. A discussion of results is given in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

The related work for this paper comes from several areas, such as ontology alignment (mapping) and semantic information retrieval. Su [4] describes an approach to ontology mapping that uses feature vectors. However, we will concentrate on work that is directed at semantic search applications.

Rocha et al. [6] present an approach to semantic search based on a combination of text search and spread activation. In the first step the users query terms are used as a query into the KB (consisting of metadata for concept instances) and returns a set of concepts, denoted as start nodes for the spread activation step. Next spread activation is applied to a graph where the concepts are represented as nodes and the relations are represented as edges. The search result returned to the user contains nodes (instances) of the ontology that have

a semantic similarity to the query, although the similarity may not explicitly be provided through the users query terms.

In [7] Pinheiro et al. present an approach to semantic search based on contextualizing the user query by adding terms that disambiguate the query. An ontology is used, and concepts from the ontology are added to the query based on the relationships with the concepts of the original query terms. The query is executed by the Google[1] search API and the results displayed to the user.

Lei et al. [8] present a system that lets the user specify queries as keywords, and hides the actual semantics from the user. The keyword query is first disambiguated by trying to find the semantic meaning of the query terms. This is done by performing a text search on the semantic entities (concepts, relations etc.), returning all matching entities. These are next used to construct formal (SeRQL [2]) queries which are queried against the semantic repository. Finally the results are ranked and presented to the user.

Guha et al. [9] describe a system that augments traditional search results with semantic data. The traditional search results are provided through the Google API. The additional information displayed to the user is based on matching the query terms to one or more semantic entities (anchor nodes) in the semantic repository. Next, a sub graph around the anchor nodes is generated, and the information found is displayed to the user. When searching for a musician, the augmented data may contain cd's recorded, concert schedules, etc.

Sieg et al. [10] describes an approach that is in essence quite similar to our approach for constructing feature vectors (see Section 4). The system uses an ontology and builds term vectors for each of the concepts in the ontology based on pre-assigned documents. During search, the query is expanded by using positive evidence (selected concepts) and negative evidence (deselected concepts). One other system that is similar to ours is the one described by Tommassen et al. [11]. The system builds feature vectors for each concept in the ontology, and uses these for subsequent query expansion. The main difference between our system and the one described in [11] is in how the feature vectors are constructed and used during search.

3 Ontological Profiles

An ontology is informally often referred to as a set of concepts and their relationships. According to Gruber "*an ontology is an explicit specification of a conceptualization*"[12]. The ontology describes the important concepts of the domain, their properties, relationships and constraints. A fundamental assumption is that these concepts describe the domain as a whole, independently of how they might be used by people or applications. It provides a standardized vocabulary that may be used to bridge applications and help humans and computers to exchange information and services.

[1] http://www.google.com
[2] http://www.openrdf.org

However, ontologies are of little use unless humans and applications consistently refer to the concepts defined in the ontologies. This may be done by adopting the ontological structures for internal information management, or by providing mappings between internal structures and ontological structures.

For search applications we will usually not be in a position to enforce a standardized vocabulary in documents or users queries. Both documents and queries may deviate from the standards defined, as they reflect the users own personalized vocabulary and preferences. The documents language and structure express the authors line of thinking rather than some abstract conceptual model. One option would be to ask the authors explicitly to annotate all documents with ontological concepts, but this requires additional manual work and intimate knowledge of the ontology at hand. Since ontologies can easily exceed tens of thousands of concepts, it is not realistic to assume that authors will correctly and consistently annotate their documents themselves. Another problem is that the ontology is not necessarily structured according to the needs of search applications. As shown in several research projects (cf. Brasethvik [1]), the granularity of the ontology must be carefully adjusted to the nature of the document collection. Whereas a very detailed ontology may be both bad for performance and recall, a coarse-grained ontology may either be of no use or even lower the precision of search.

Take for example the christmas tree in Figure 1. According to the ISO 15926 standard used in the petroleum industry, a christmas tree is "*an artefact that is an assembly of pipes and piping parts, with valves and associated control equipment that is connected to the top of a wellhead and is intended for control of fluid from a well.*" The concept is together with around 50,000 other petroleum-related concepts defined in an ontology that is now under completion and is already used in government production reports from the Norwegian Continental Shelf. The introduction of this ontology is supposed to simplify collaboration across disciplines, phases and companies in the Norwegian petroleum industry, and it is part of an integrated operations initiative for streamlining the subsea petroleum production processes.

However, the term christmas tree may not be used consistently throughout the domain. There are of course synonyms like x-tree in use, but also more indirect references like wellhead top, tree or wellhead control. Texts about boreholes and wellhead housings may not mention christmas trees at all, even though it most likely will deal with topics relevant to christmas trees. Looking at the ontological hierarchy in Figure 1, it is also clear that many ontological concepts will never be used in natural language texts. A christmas tree is a specialization of artefact, which is again a specialization of ISO-IS_15926_inanimate_physical_object, but these terms are not likely to appear in texts about christmas trees although they are likely to be related in an ontological sense. Many concepts are needed to provide unambiguous and exhaustive domain definitions and are not normally used in documents from the domain.

The aforementioned problems makes it problematic to use ontological concepts to index documents or reformulate user queries in search applications.

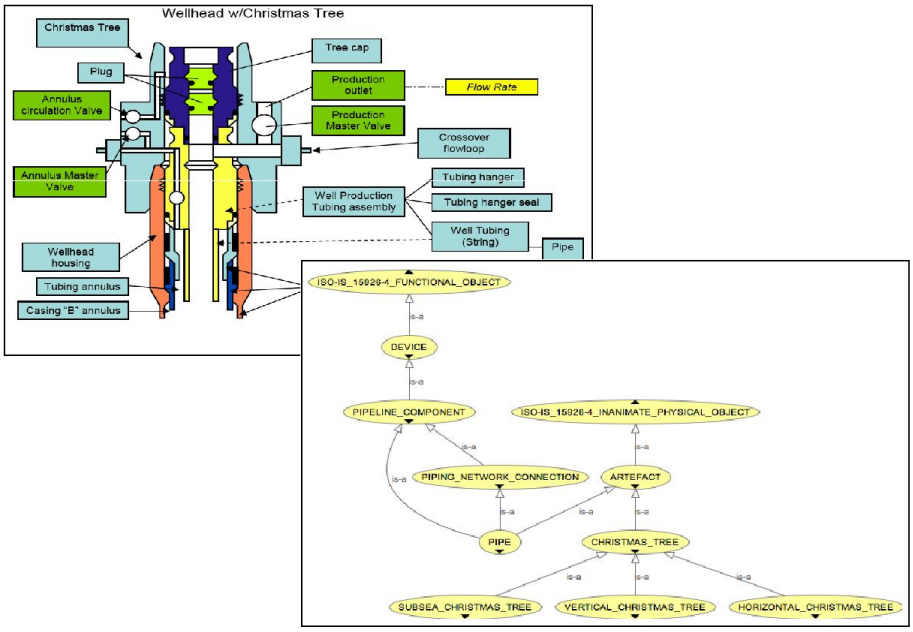

Fig. 1. (Top/left) Schematic view of a Wellhead w/Christmas Tree. (Bottom/right) Part of the IIP ontology focused around the christmas tree. Adapted from [2]

Having clearly defined concepts in a domain is useful, but only if we can relate these concepts to the everyday language used in documents and queries.

An ontological profile is an extension of an ontology, in which every concept is given a semantic vector definition. This vector provides a weighted list of terms that are semantically related to the concept itself. The term weights are given in the range 0 to 1. The higher the weight, the more central is the term with respect to the concept.

Fundamental to ontological profiles is the fact that they only reflect the language used in a particular application or the texts available from a particular information source. The profile is a semantic characterization of the application or information source in terms of well-defined domain concepts. Similarly, the profile is an interpretation of ontological concepts with respect to a particular application or information source.

4 Constructing Ontological Profiles

The ontological profiles are constructed on the basis of a document collection reflecting the vocabulary of the domain, thus linking the concepts with the actual language used in the domain.

The approach we have used to construct the ontological profiles is found in [3], based on a method described by Su [4]. In Figure 2 an overview of the approach to the construction of the ontological profiles is shown as the indexing phase part of the figure.

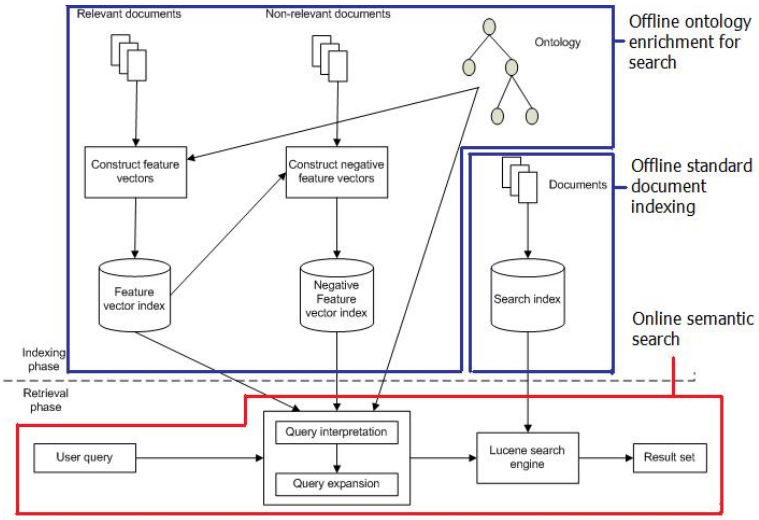

Fig. 2. Overview of the approach.

The documents used during the construction of the ontological profiles are first indexed using a standard Lucene [3] implementation based on the vector space model [5]. During this process, stop words are removed, and the terms are stemmed lightly, using the conversion $s \rightarrow \emptyset$ for all terms not ending with ss. We have chosen to build three separate indexes based on the document collection to reflect three views of the documents. The first view reflects the bag of words approach, in which all the words are in some manner related since they appear in the same document. In the second view, we partition the document into paragraphs (or paragraph documents) based on two or more consecutive line breaks. This view considers each paragraph as a semantic entity, in which the words are relatively tightly linked semantically. Our third and last view partitions each document into sentences (or sentence documents), based on the punctuations ".", "!", and "?". The sentence represents a group of words that are very tightly linked semantically. During the indexing procedure, we thus construct one index for each of the three views, and each document in the document collection is indexed based on the whole document, based on paragraphs, and based on the sentences in the document. The reason for partitioning the documents according

[3] http://lucene.apache.org/

to this schema is that we hypothesize that relations between concepts and terms are stronger when the two are found closer together.

When the appropriate index structures are created, the next step of the process is to do the actual construction of the ontological profile. In this step we use the ontology together with the indexes just created as input, and get the ontological profile as output. For each of the concepts in the ontology, we use the name of the concept as a query into the three different indexes. As many of the ontology concepts consist of multiple words, we use the concept name as a phrase query into the indexes. Since many of the concepts are rather long phrases, or even artificial in the construction (cf. ISO-IS_15926_inanimate_physical_object), this imposes a challenge when searching for the concepts. The artificial concepts may not be found at all, and longer phrases may be broken up in the text making it hard to locate them, resulting in many of the concepts not being associated with a concept vector. Each concept is assigned to it the set of relevant documents (documents, paragraph documents, and sentence documents) found during search. We now construct the feature vector for each of the concepts by differentiating the weight given to terms found in documents, paragraph documents, and sentence documents. Equation 1 shows how this is done, where $vf_{i,j}$ is the term frequency for term i in concept vector j, $f_{i,k}$ is the term frequency for term i in document vector k, D, P, S is the (possibly empty) set of documents, paragraph documents, and sentence documents assigned to j, respectively, and α, β, γ are the constant modifiers for weighting each document type. The weight is differentiated to reflect the fact that words found in the same document as the concept name is likely to by semantically related to the concept. Terms found in the same paragraph as the concept name is likely to be more semantically related to the concept, and finally terms found in the same sentence as the concept name are likely to be most semantically related to the concept name. The weight given to terms found in the same document, paragraph, and sentence is $\alpha = 0.1, \beta = 1.0$, and $\gamma 10.0$, respectively. These feature vectors are now what we refer to as raw feature vectors.

$$vf_{i,j} = \alpha \cdot \sum_{d \in D} f_{i,d} + \beta \cdot \sum_{p \in P} f_{i,p} + \gamma \cdot \sum_{s \in S} f_{i,s} \tag{1}$$

The raw feature vectors are transformed using the tfidf [5] score. The idf factor is calculated by letting the raw feature vectors act as documents. The calculation of the tfidf score is shown in Equation 2, where $tfidf_{i,j}$ is the tfidf score for term i in feature vector j, $vf_{i,j}$ is the raw term frequency for term i in feature vector j, $max(vf_{l,j})$ is the frequency of the most frequent occurring term l in feature vector j, N is the total number of feature vectors, and n_i is the number of feature vectors containing term i. After the tfidf score is calculated for each of the concept vectors, each vector is normalized to unit length in order to let search reflect the prominence of each term within the vector. The last part of the construction process is to index the feature vectors so that the contents of the feature vectors are searchable.

$$tfidf_{i,j} = \frac{vf_{i,j}}{max(vf_{l,j})} \cdot \log \frac{N}{n_i} \quad (2)$$

Next we introduce the concept of negative feature vectors. For each concept that we successfully built a feature vector, we also attempt to create a negative feature vector. The same principles as above have been applied, with exception to three important details. The first is that the documents used as a basis are not relevant for the domain of the ontology. This means that we build a negative feature vector containing terms that are not normally associated with the domain. Take for instance the concept christmas tree which has been explained earlier. Typical terms contained in the negative feature vector for christmas tree would be pine, decoration, turkey etc. The second difference is that the query used in the construction process is not directly based on the concept name, but rather on the top 5 terms for each feature vector. We have chosen to use this approach as using the concept name as a query into the index would not get many hits, as quite many of the concepts are domain specific phrases. In addition we have chosen to filter the terms used in the negative feature vector by removing all terms that are found in the "parent" feature vector, and all terms that are found in more than 5% of the concept feature vectors. Third, the top 15 terms for each of the negative feature vectors are indexed in a separate negative feature vector index. The main purpose of the negative feature vector is to help filter results that are not at all relevant to the domain, even though they may contain terms found in the concept feature vectors. The weights are not included in the index, and the reason for this will be explained in the next section.

5 Interpreting Queries with Ontological Profiles

Before we go into more detail about the query interpretation, we will give a brief overview of how the process is done. The interpretation and expansion of the user entered query is implemented as a module that is placed between the user interface and the search core based on Lucene. The purpose of this module is to map the user query to one or more concepts from the ontology using the ontological profile. The user entered query is subsequently expanded using the semantic representation of the concept given by the ontological profile. Figure 3 shows a conceptual view of the process. The user enters the query horizontal tree and the semantic layer, represented in the module by the query interpretation process, interprets the query semantically.

The query terms are used as a query into the feature vector index, returning a ranked list of concepts based on the weight of the term with respect to the concepts. This mapping reflects the vocabulary used in the document collection used to construct the ontological profile. The next step is to reformulate the original query by the terms found in the feature vector for the concept(s) chosen for expansion.

We have implemented four different approaches for query interpretation described in the following.

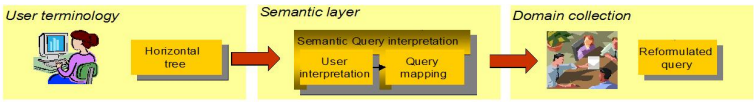

Fig. 3. Conceptual view of the query interpretation and expansion process.

Simple query interpretation This approach is what we consider the naive approach, as it maps each of the query terms to a single concept. Each of the user entered query terms is used as a query into the feature vector index, returning the concept with the highest tfidf score for the term. This concept is picked as a semantic representation of the query term and subsequently used in the expansion process.

Best match query interpretation The simple approach is naive in the sense that it does not recognize the relation between the query terms entered by the user. Concepts are chosen for expansion solely based on the fact that a concept is the best semantic representation for the specific query term. This approach is an attempt at improving the mapping between the query terms and the concepts by assuming that the query terms entered by the user are related. We thus try to map the query terms to a single concept that has a good representation for the query terms collectively. This is done by requiring the candidate concepts to contain all the query terms. Equation 3, where $t_{i,c}$ is the weight of query term i in concept c, shows the metrics for query term to concept mapping. The mapping with the highest score, $score_c$, is chosen as a semantic representation for the user entered query.

$$score_c = t_{0,c} + t_{1,c} + \ldots + t_{n-1,c} \tag{3}$$

Cosine similarity query interpretation The following two approaches are attempts at disambiguating the query. The simple approach naively maps the terms to the best corresponding concepts, not recognizing that there may exist a relation between the best concept for $queryterm_1$ and the second best concept for $queryterm_2$. This approach is an attempt at recognizing such relations by taking the cosine similarity between the concept feature vectors into account during the query to concept mapping phase. Due to computational complexity this approach (together with the ontology structure interpretation) is only implemented for user queries consisting of two terms. For each of the two terms a ranked list (top 15) of concepts based on the tfidf score is used as a basis for the calculation. For each pair of concepts a score is calculated and the pair of concepts resulting in the highest score is chosen as a semantic representation of the original query. Equation 4 shows the calculation, where ckl represents the concept ranked as l with respect to query term k, $t_{k,l}$ represents the weight of the query term, k, in the concept ranked as l with respect to k and cos_sim is the function that

calculates the cosine similarity between two concepts. Note that the approach may result in a single concept being chosen in the case where both query terms are present with a high score within a single feature vector.

$$score_{cin,cjm} = t_{i,n} * t_{j,m} * cos\_sim(cin, cjm) \tag{4}$$

Ontology structure query interpretation The last approach, ontology structure query interpretation, is just as the cosine based approach an attempt at finding the pair of concepts with largest semantic coherence with respect to the original query. Just as with the cosine based approach, the first step is to generate a ranked list of the top 15 concept feature vectors based on the tfidf score for each of the terms in the original query. In this approach the structure of the ontology itself is used as a basis for the interpretation of the original query. A graph representation of the ontology is built, in which the concepts are nodes and the edges represent the child/parent relations in the ontology. For each pair of concepts related to the query terms a score is calculated based on Equation 5, where ckl, and $t_{k,l}$ are identical to ckl and $t_{k,l}$ in Equation 4 and $path(i, j)$ is a function that calculates the length of the path between two concepts of the ontology. The score is based on the distance between the concepts in the ontology, using the assumption that concepts that are close within the ontology have a stronger semantic relation than concepts that are far apart in the ontology. The pair of concepts with the highest score is chosen as the semantic representation of the original query. Also in this approach we note that a single concept may be chosen as the semantic representation due to high score for both terms within a single concept.

$$score_{cin,cjm} = t_{i,n} * t_{j,m} * \frac{1}{path(cin, cjm)} \tag{5}$$

Query Expansion Once the query interpretation phase is done, the original query is expanded by the concepts chosen as a semantic representation for the original query. For each of the concept(s) chosen for expansion the top 15 terms are used as a semantically reformulated query. The weight of the terms within the feature vectors are included so that the relative importance of each term within the concept representation is conserved. Original query terms are boosted to reflect the importance of them as they were selected by the user. In case a term is not found in the feature vector index, the term itself is used for querying (no semantic representation of the term other than itself). In addition the top 15 terms in each of the corresponding negative feature vector(s) (if they exist) are added as a NOT query, prohibiting any of the obtained search results from containing any terms from the negative feature vectors. The new query consists of two parts, the conceptual weighted representation of the terms, and the NOT query based on atypical terms for the domain.

6 Experiment

The domain in which the approach was evaluated was subsea petroleum equipment and installations. We used the IIP [2] core ontology containing 18,675 concepts. During the construction of the ontological profile, 2195 concept feature vectors were built together with 2006 negative feature vectors. The reason for this low number may be that many of the concepts were not mentioned in the text, that they are not commonly used in "daily" speak, or the phrases may have been broken up in the text. The document collection used to build the feature vectors is the Schlumberger Oilfield Glossary [4], consisting of 4132 files totaling 2.2 MB of text. The document collection used to build the negative feature vectors consisted of 82 files found using the Google Search API. The search index contained the Schlumberger Oilfield Glossary together with 130 documents (4 MB), assumed to be relevant, found using the Google Search API and 99 documents (0.5 MB) that were assumed to be non-relevant for the domain.

The prototype was evaluated using 7 queries, and 5 test subjects. The search using the reformulated query is applied to an index based on paragraphs of minimum length 1200 characters rather than the full documents. The reformulated query was evaluated against standard keyword search (the original query) based on both paragraph indexing and document indexing. Each of the test subjects were asked to score the top 10 hits for each of the queries and search strategy with a score from 0 to 2, where 0 designates the hit as being totally irrelevant, 1 related to the domain, and 2 relevant for the query.

Figure 4 shows the average score for each of the search strategies over all 7 queries and all test subjects. From the figure we can note that our four reformulation strategies seem to perform quite equally. It is hard to point out one of them as being superior to any of the others. The reason for the equality in performance may be that the parameters used for query expansion do not differentiate that much between the different strategies, leading to many of them expanding similar concepts. We also see that compared to the standard keyword based search based on paragraph indexing (Lucene par) and documents (Lucene doc) our reformulated queries perform significantly better.

We will now show the results of three of the seven queries tested. First we look at Figure 5, which shows the results for the query chrstmas tree. From Table 1, which shows the concepts expanded by each of the reformulation strategies for the three chosen queries, we see that all the queries expand the concept "Christmas Tree", and that two of them in addition expand the concept "Tree". The equality in expanded concepts explains why the results for the reformulated queries perform equal. What is the reason for the poor performance for the keyword search? Recall that the document collection searched includes some christmas holiday type documents. Although there are not many of them in the collection, these are ranked high in the keyword based approach, due to a relatively high concentration of the terms christmas and tree. In addition

[4] http://www.glossary.oilfield.slb.com

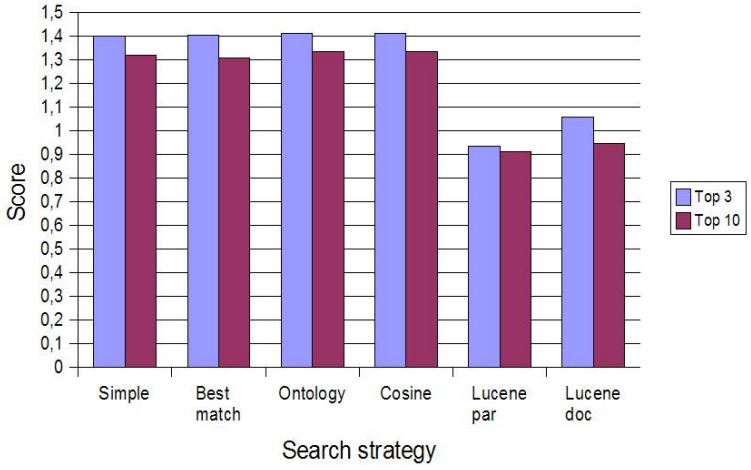

Fig. 4. The average score for each of the search strategies calculated over all 7 queries and 5 test subjects.

we note that the reformulated queries in general retain a quite high score of approximately 1.7 showing that the results are quite precise for the petroleum domain. We explain this by a combination of the added semantics to the query together with the negative feature vectors which filter the irrelevant documents.

Table 1. The concept expansions made by different reformulation strategies.

Strategy\Query	`christmas tree`	`valve control`	`tree pipe`
Simple	"christmas tree" "tree"	"on off valve" "on off control"	"tree" "pipe"
Best match	"christmas tree"	"on off valve"	"pipe"
Ontology	"christmas tree"	"valve control"	"christmas tree" "pipe"
Cosine	"christmas tree" "tree"	"on off valve" "on off control"	"tree" "pipe"

Figure 6 shows the score given by the test subjects for the query valve control. The figure shows that the reformulated queries for the strategies Simple, Best Match, and Cosine perform quite equally. Looking at the concept expansions in Table 1, we see that these strategies expand a combination of two different concepts. This helps explain why the strategies perform so similarly. The last strategy, Ontology, expands to a different concept, Valve control, and is perceived by the test subjects to give the best score. We also note that in this particular

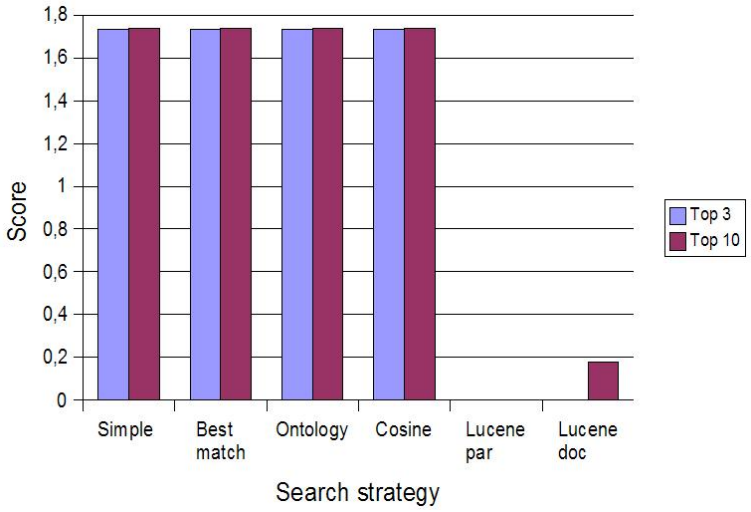

Fig. 5. Average score for the top 3 and top 10 hits for each of the search strategies for query *christmas tree*.

query the reformulated queries perform approximately equal to or slightly poorer than the keyword based approaches. One reason for this may be that the initial query is precise.

Figure 7 shows the results for the query tree pipe, and we note that our reformulated queries perform significantly better than the keyword based queries with exception to the top 3 hits for the Lucene document based search. The strategies that seem to perform best are the Simple, and the Cosine based reformulated queries, which both actually expand the concepts "Tree" and "Pipe", correctly identifying the concepts of the query.

Table 2 shows the total overlap for the paragraphs retrieved and does obviously not include the Lucene doc. The table is read as follows; the search strategy labeled by the columns found $X\%$ of the paragraphs found by the row labeled strategies. From the table we note that the four reformulation strategies have quite high overlap, which may explain to some degree why their performance does not vary so much. We note that we have no measure for where in the ranked list the overlap occurs. On the other hand we note that the reformulated queries only locate in the range of 56% - 64% of the paragraphs found by the keyword search. This result may in our opinion be caused by the negative feature vectors filtering certain hits.

Table 3 shows the total hits made by the four reformulation strategies and the Lucene paragraph based search. This shows that the number of hits for the reformulated queries is significantly higher than for the keyword based search. Of course, this is expected as the reformulated queries expand the original query resulting in more hits.

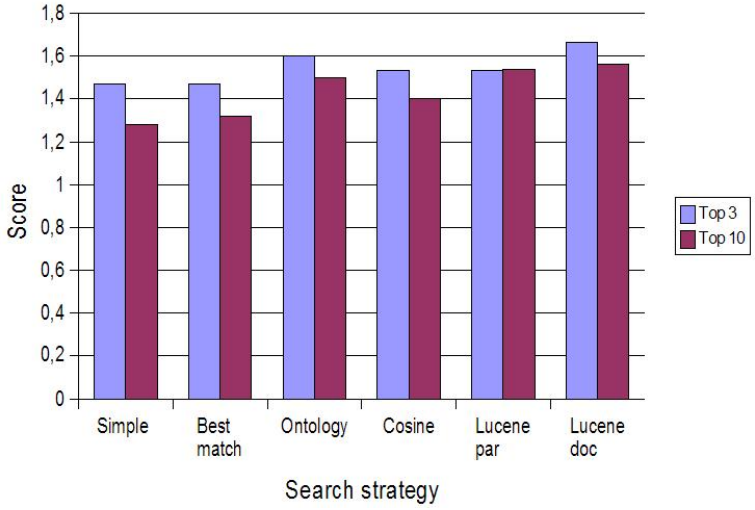

Fig. 6. Average score for the top 3 and top 10 hits for each of the search strategies for query *valve control*.

Table 2. Total overlap for the paragraphs retrieved.

	Simple	Best match	Ontology	Cosine	Lucene par
Simple	100,00%	76,69%	75,25%	90,15%	23,53%
Best match	91,64%	100,00%	84,32%	95,07%	32,10%
Ontology	92,31%	86,56%	100,00%	95,84%	30,80%
Cosine	96,91%	85,53%	83,98%	100,00%	25,55%
Lucene par	56,48%	64,48%	60,26%	57,06%	100,00%

The experimentation has shown that there is not that much difference in the results among the four strategies that we have proposed, and it is certainly not possible to point out one of the as the best both generally and for certain types of queries. We will in the future examine further the mapping of query terms to concepts, to try to find an approach that does achieve more differentiation between the strategies. Further, the approach relies on quite many different variables which have not been researched yet, so we will do more research to find out how these variables will influence the search results. The number of terms from the concept feature vector to use in the expansion of the query has been set to 15, but has not been researched to find the optimal number. This number may rely on the quality of the vectors and the type of query. Regarding the handling of the negative feature vectors, our opinion is that the terms found in them are handled too strict, removing any document from the result set if it contains terms from the negative feature vector. We propose to use less strict

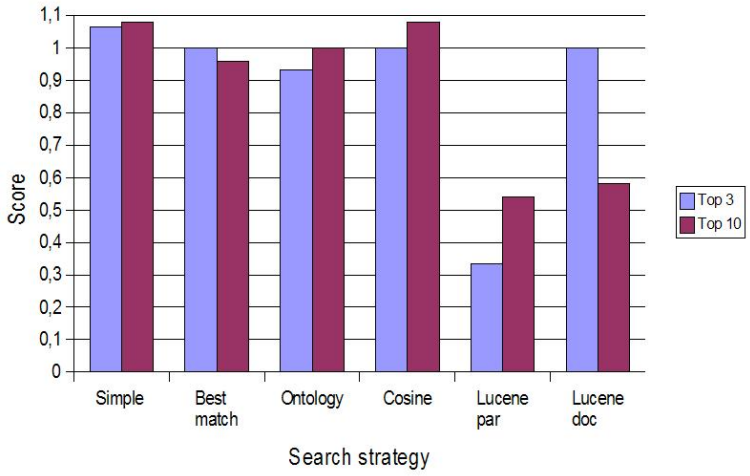

Fig. 7. Average score for the top 3 and top 10 hits for each of the search strategies for query *tree pipe*.

Table 3. Total number of hits for each search strategy.

	Simple	Best match	Ontology	Cosine	Lucene par
Total paragraph hits:	16616	13905	13545	15457	6923
Total document hits:	11385	8714	8509	10419	2574
Average paragraphs per document:	1.46	1.60	1.59	1.48	2.69

handling, in which documents are penalized for containing such terms, rather than removing them from the result set altogether.

Lastly we must point out that the evaluation of the prototype has not been a full blown evaluation with statistical significant results. However, we deem the evaluation results good enough to give an indication that the search approach based on ontological profiles has strengths that are worth pursuing further.

7 Conclusions

This paper has explored an approach for constructing a semantic search application based on ontological profiles. We have seen that the general approach is promising, giving results that are better than for basic keyword search. One surprising result is that the four strategies we suggested perform on average quite equally. We had an initial thought that the different strategies might be strong for certain types of queries and weaker for others, but we did not see any such

trends. We will continue to work on the approach to find if there are any parameter settings that may give certain strategies benefits for certain queries. We also plan to look into how reasoning in the ontology may be used to improve the search, as well as find answers to queries that are found directly in the ontology.

Acknowledgment. This research was carried out as part of the IS_A project, project no. 176755, funded by the Norwegian Research Council under the VERDIKT program.

References

1. T. Brasethvik. Conceptual modeling for domain specific document description and retrieval. PhD Thesis, Norwegian University of Science and Technology, 2004.
2. J.A. Gulla, S.L. Tomassen, D. Strasunskas. Semantic interoperability in the norwegian petroleum industry. In D. Karagiannis, H.C. Mayer, (Eds.), 5th International Conference on Information Systems Technology and its Applications (ISTA 2006), volume P-84 of Lecture Notes in Informatics (LNI), pages 81-94. Köllen Druck Verlag GmbH, Bonn, Klagenfurt Austria, 2006.
3. G. Solskinnsbakk. Extending Ontologies with Search-Relevant Weights. Technical report, Norwegian University of Science and Technology, Trondheim, Norway, 2006.
4. X. Su. Semantic Enrichment for Ontology Mapping. PhD thesis, Norwegian University of Science and Technology, 2004.
5. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval, ACM Press, New York, 1999.
6. C. Rocha, D. Schwabe, M.P.d. Arago. A Hybrid Approach for Searching in the Semantic Web. In Proceedings of the Thirteenth International Conference on World Wide Web (WWW'04). New York, NY, USA. p. 374-383. 2004.
7. W. A. Pinheiro, A. M. d. C. Moura. An Ontology Based-Approach for Semantic Search in Portals. In Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA'04) , IEEE Computer Society. 2004.
8. Y. Lei, V. Uren, E. Motta. SemSearch: A Search Engine for the Semantic Web. 15th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks (EKAW 2006). Podebrady, Czech Republic.
9. R. Guha, R. McCool, E. Miller. Semantic Search. WWW 03: Proceedings of the Twelfth International Conference on World Wide Web. Budapest, Hungary. 2003.
10. A. Sieg, B. Mobasher, R. Burke, G. Prabu, S. Lytinen. Representing User Information Context with Ontologies. In Proceedings of the 3rd International Conference on Universal Access in Human-Computer Interaction, HCI International 2005, Las Vegas, NV, July 2005.
11. Tomassen, S.L., Gulla, J.A., Strasunskas, D.: Document Space Adapted Ontology: Application in Query Enrichment. In: 11th International Conference on Applications of Natural Language to Information Systems (NLDB 2006), Vol. 3999. Springer-Verlag, Klagenfurt, Austria (2006) 46-57
12. Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2):199-220, 1993.