An Ontology-Based Information Retrieval Model

David Vallet, Miriam Fernández, and Pablo Castells

Universidad Autónoma de Madrid
Campus de Cantoblanco, c/ Tomás y Valiente 11, 28049 Madrid
{david.vallet, miriam.fernandez, pablo.castells}@uam.es

Abstract. Semantic search has been one of the motivations of the Semantic Web since it was envisioned. We propose a model for the exploitation of ontology-based KBs to improve search over large document repositories. Our approach includes an ontology-based scheme for the semi-automatic annotation of documents, and a retrieval system. The retrieval model is based on an adaptation of the classic vector-space model, including an annotation weighting algorithm, and a ranking algorithm. Semantic search is combined with keyword-based search to achieve tolerance to KB incompleteness. Our proposal is illustrated with sample experiments showing improvements with respect to keyword-based search, and providing ground for further research and discussion.

1 Introduction

The use of ontologies to overcome the limitations of keyword-based search has been put forward as one of the motivations of the Semantic Web since its emergence in the late 90's. While there have been contributions in this direction in the last few years, most achievements so far either make partial use of the full expressive power of an ontology-based knowledge representation, or are based on boolean retrieval models, and therefore lack an appropriate ranking model needed for scaling up to massive information sources.

In the former case, ontologies provide a shallow representation of the information space, equivalent in essence to the taxonomies and thesauri used before the Semantic Web was envisioned [3,6,7,15]. Rather than an instrument for building knowledge bases, these light-weight ontologies provide controlled vocabularies for the classification of content, and rarely surpass several KBs in size. This approach has brought improvements over classic keyword-based search through e.g. query expansion based on class hierarchies and rules on relationships, or multifaceted searching and browsing. It is not clear though that these techniques alone really take advantage of the full potential of an ontological language, beyond those that could be reduced to conventional classification schemes.

Other semantic search techniques have been developed that do exploit large knowledge bases in the order of GBs or TBs consisting of thousands of ontology instances, classes and relations of arbitrary complexity [1,2,4,12]. These techniques typically use boolean search models, based on an ideal view of the information space as consisting of non-ambiguous, non-redundant, formal pieces of ontological knowledge. In this view, the information retrieval problem is reduced to a data retrieval task. A knowledge item is either a correct or an incorrect answer to a given information request, thus search results are assumed to be always 100% precise, and there is no notion of approximate answer to an information need. This model makes sense when the whole information corpus can be fully represented as an ontology-driven knowledge base, so that search results consist of ontology entities.

However, there are limits to the extent to which knowledge can or should be formalized in this way. First, because of the huge amount of information currently available to information systems worldwide in the form of unstructured text and media documents, converting this volume of information into formal ontological knowledge at an affordable cost is currently an unsolved problem in general.

Second, documents hold a value of their own, and are not equivalent to the sum of their pieces, no matter how well formalized and interlinked. The replacement of a document by a bag of information atoms inevitably implies a loss of information value: the thread of thought behind the order of the sentences in free text, the choice of the words, etc., are a valuable, relevant, and necessary part of the conveyed message. Therefore, although it is useful to break documents down into smaller information units that can be reused and reassembled to serve different purposes, it is yet often appropriate to keep the original documents in the system.

Third, wherever ontology values carry free text, boolean semantic search systems do a full-text search within the string values. In fact, if the string values hold long pieces of free text, a form of keyword-based search is taking place in practice beneath the ontology-based query model since, in a way, unstructured documents are hidden within ontology values, whereby the “perfect match” assumption starts to become arguable, and search results may start to grow in size. While this may be manageable and sufficient for small knowledge bases, the boolean model does not scale properly for massive document repositories where searches typically return hundreds or thousands results. Boolean search does not provide clear ranking criteria, without which the search system may become useless if the search space is too big.

In this paper we propose an ontology-based retrieval model meant for the exploitation of full-fledged domain ontologies and knowledge bases, to support semantic search in document repositories. In contrast to boolean semantic search systems, in our perspective full documents, rather than specific ontology values from a KB, are returned in response to user information needs. The search system takes advantage of both detailed instance-level knowledge available in the KB, and topic taxonomies for classification. To cope with large-scale information sources, we propose an adaptation of the classic vector-space model [16], suitable for an ontology-based representation, upon which a ranking algorithm is defined.

The performance of our proposed model is in direct relation with the amount and quality of information within the KB it runs upon. The latest advances in automating ontology population and text annotation are promising [5,9,11,14]. While, if ever, ontologies and metadata (and the Semantic Web itself) become a worldwide commodity, the lack or incompleteness of available ontologies and KBs is a limitation we shall likely have to live with in the mid term. In consequence, tolerance to incomplete KBs has been set as an important requirement in our proposal. This means that the recall and precision of keyword-based search shall be retained when ontology information is not available or incomplete.

We have implemented our model and done some low-scale experimentation with real documents and data from a digital news archive from a local Spanish newspaper. The experiments build upon previous work in the Neptuno project [1], where an ontology and a knowledge base were built for the description of archive news.

The rest of the paper is organised as follows. An overview of related work is given in Section 2. After this, our scheme for semantic annotation is described. Section 4 explains the retrieval and ranking algorithms. Some initial experiments with our techniques are reported in Section 5 The strengths, weaknesses, and significance of our approach are summarized in Section 6, after which some closing conclusions are given.

2 State of the Art

Our view of the semantic retrieval problem is very close to the latest proposals in KIM [11,14]. While KIM focuses on automatic population and annotation of documents, our work focuses on the ranking algorithms for semantic search. Along with TAP [8], KIM is one of the most complete proposals reported to date, to our knowledge, for building high-quality KBs, and automatically annotating document collections at a large scale. In their latest account of progress [11] a ranking model for retrieval is hinted at but has not been developed in detail and evaluated. In fact, KIM relies on a keyword-based IR engine for this purpose (indexing, retrieval and ranking). Our work complements KIM with a ranking algorithm specifically designed for an ontology-based retrieval model, using a semantic indexing scheme based on annotation weighting techniques.

TAP [8] presents a view of the Semantic Web where documents and concepts are nodes alike in a semantic network, whereby the separation of contents and metadata is not as explicit as we propose here. The two main problems addressed by TAP are a) the development of a distributed query infrastructure for ontology data in the Semantic Web, and b) the presentation of query execution results, augmenting query answers with data from surrounding nodes. These issues are complementary to the ones addressed in this paper. However the expressive power of the TAP query language is fairly limited compared to ontology query languages such as RDQL, RQL, etc. The supported search capability is limited to keyword search within the “title properties” of instances, and no ranking is provided.

Mayfield and Finin [13] combine ontology-based techniques and text-based retrieval in sequence and in a cyclic way, in a blind relevance feedback iteration. Inference over class hierarchies and rules is used for query expansion, and extension of semantic annotations of documents. Documents are annotated with RDF triples, and ontology-based queries are reduced to boolean string search, based on matching RDF statements with wildcards, at the cost of losing expressive power for queries. We share with Mayfield et al the idea that semantic search should be a complement of keyword-based search as long as not enough ontologies and metadata are available. Also, we believe that inferencing is a useful tool to fill knowledge gaps and complete missing information (e.g. transitivity of the *locatedIn* relationship over geographical locations).

Semantic Portals [1,2,4,12] typically provide simple search functionalities that may be better characterised as semantic data retrieval, rather than semantic information retrieval. Searches return ontology instances rather than documents, and no ranking method is provided. In some systems, links to documents that reference the instances are added in the user interface, next to each returned instance in the query answer [4], but neither the instances, nor the documents are ranked. Maedche et al do provide a criterion for query result ranking in the SEAL Portal [12], but the principles on which the method is based – a similarity measure between query results and the original KB without axioms, is not clearly justified, and no testing of the method is reported.

The ranking problem has been taken up again in [19], and more recently [15]. Rocha et al propose the expansion of query results through arbitrary ontology relations starting from the initial query answer, where the distance to the initial results is used to compute a similarity measure for ranking [15]. This method has the advantage of allowing the user to express information needs with simpler, keyword-based queries but, from our perspective, it introduces an unnecessary loss of precision, since a more accurate result expansion can be achieved by including ontology relations explicitly in a structured query. From our point of view, Rocha's techniques would be appropriate in a more browsing-oriented information seeking context. Stojanovic et al propose a sentence ranking scheme based on the number of times an instance appears as a term in a relation type, and the derivation tree by which a sentence is inferred [19]. Whereas these works are concerned with ranking query answers (i.e. ontology instances), we are concerned with ranking the documents annotated with these answers. Since our respective techniques are applied in consecutive phases of the retrieval process, it would be interesting to experiment the integration of the query result relevance function proposed by Stojanovic et al into our document relevance measures.

3 Knowledge Base and Document Base

In our view of semantic information retrieval, we assume a knowledge base has been built and associated to the information sources (the document base), by using one or several domain ontologies that describe concepts appearing in the document text. The concepts and instances in the KB are linked to the documents by means of explicit, non-embedded annotations to the documents.

While we do not address here the problem of knowledge extraction from text [4,5,9,10,11,14], we provide a vocabulary and some simple mechanisms to aid in the semi-automatic annotation of documents, once ontology instances have been created (manually or automatically). These are described in Subsection 3.2. Our system can work with any arbitrary domain ontology with essentially no restrictions, except for some minimal requirements that are explained next.

3.1 Root Ontology Classes

Our system requires that the knowledge base be constructed from three main base classes: *DomainConcept*, *Taxonomy*, and *Document*. *DomainConcept* should be the root of all domain classes that can be used (directly or after subclassing) to create instances that describe specific entities referred to in the documents. For example, in the Arts domain, classes like *Artist*, *Sculptor*, *ArtWork*, *Painting*, and *Museum* should be defined as (probably indirect) subclasses of *DomainConcept*. A small set of upper-level open-domain classes like *Person*, *Building*, *Event*, *Location*, etc., is included in the base concept ontology, to be extended for specific domains.

Document is used to create instances that act as proxies of documents from the information source to be searched upon. Two subclasses, *TextDocument* and *MediaContent*, are provided, which can be further subclassed, if appropriate for a particular application domain, to provide for different types of documents, such as *Report*, *News*, *PurchaseOrder*, *Invoice*, *Message*, etc., with different fields (e.g. *title*, *date*, *subject*, *price*, *sender*). The class *MediaContent* is provided in anticipation of future extensions for multimedia retrieval, which we have not developed yet. *Document* has a *location* property that holds a dereferenceable physical address from which the actual document contents can be retrieved. In our current implementation, this consists of a URL.

Taxonomy is the root for class hierarchies that are merely used as classification schemes, and are never instantiated. These taxonomies are expected to be used as a terminology to annotate documents and concept classes, using them as values of dedicated properties. For instance, in a KB for news, classes like *Culture*, *Politics*, *Economy*, *Sports*, etc. (after the IPTC Subject Reference System standard[1]), could be used as values of a (probably multivalued) *topic* property of the *News* class. Furthermore, concept classes like *Athlete* and *Tournament* could also have the *topic* property, in this case with the value *Sports*, i.e. concepts can also be classified under the same scheme as documents. Several separate taxonomies can be used simultaneously on the same documents, thus providing for multifaceted classification.

The distinction between the three root classes *DomainConcept*, *Taxonomy*, and *Document*, arises from our own experience in previous Semantic Web projects [1,2], and many other observed information systems where this (or a similar distinction) seems to be natural, useful and recurrent (see e.g. [17]). In our system, we exploit taxonomies for multifaceted search, and to solve word ambiguities, as will be described later.

3.2 Document Annotation

The predefined base ontology classes described above are complemented with an annotation ontology that provides the basis for the semantic indexing of documents with non-embedded annotations. In many respects, our scheme for semi-automatic annotation is similar to the one recently reported in [11].

Documents are annotated with concept instances from the KB by creating instances of the *Annotation* class, provided for this purpose. *Annotation* has two relational properties, *instance* and *document*, by which concepts and documents are related together. Reciprocally, *DomainConcept* and *Document* have a multivalued *annotation* property.

Annotations can be created manually by a domain expert, or semi-automatically. The subclasses *ManualAnnotation* and *AutomaticAnnotation* are used respectively, to differentiate each case. We have found this distinction useful for the system at least

[1] http://www.iptc.org/NewsCodes

because a) manual annotations are more reliable than automatic ones, and when available should prevail, and b) while automatic annotations can be deleted for recalculation, manual annotations should be preserved.

Our system provides a simple facility for semi-automatic annotation, which works as follows. *DomainConcept* instances use a *label* property to store the most usual text form of the concept class or instance. This property is multivalued, since instances may have several textual lexical variants. Close equivalents of our *label* property are used in systems like KIM [11] and TAP [8]. In our current experiments, the value of this property is set by hand by the ontology designer, but it could be set by automatic means, if an external instance generation system was plugged to our system. Similarly to KIM, instance labels are used by the automatic annotator to find potential occurrences of instances in text documents. Whenever the label of an instance is found, an annotation is created between the instance and the document. In our system, documents can be annotated with classes as well, by assigning labels to concept classes.

This basic mechanism is complemented with heuristics to cope with polysemia, i.e. label coincidence between different instances or classes. First the system always tries to find the longest label, e.g. “Real Madrid” is preferred to “Madrid”. Second, classification taxonomies are used as a source of semantic scope for disambiguation: a similarity measure is defined to compare the respective classification of the document and candidate synonym instances for annotation, so that the instance that has the closest classification to the document is chosen. For example, the word “Irises” in a document classified under *Arts* would be linked to an instance of *Painting* that represents Van Gogh’s famous work, rather than a subclass of *Flower*, provided that the painting instance exists in the knowledge base and has been correctly classified under *Arts*, or a taxonomic subclass thereof, and assuming that *Flower* is classified under a different taxonomic branch such as *Botany* or the like. Of course, if the *Painting* instance does not exist, our system fails because it would incorrectly annotate the document with the botanic sense.

Our semi-automatic annotation mechanisms can be further improved, but this is out of the extent of our undergoing research. More sophisticated annotation techniques, as have been reported in the literature [5,9,11,14], would be complementary and beneficial to our system.

3.3 Weighting Annotations

The annotations are used by the retrieval and ranking module, as will be explained in Section 4. The ranking algorithm is based on an adaptation of the classic vector-space model [16]. In the classic vector-space model, keywords appearing in a document are assigned weights reflecting that some words are better at discriminating between documents than others. Similarly, in our system, annotations are assigned a weight that reflects how relevant the instance is considered to be for the document meaning. Weights are computed automatically by an adaptation of the TF-IDF algorithm [16], based on the frequency of occurrence of the instances in each document. More specifically, the weight $w_{i,j}$ of instance I_i for document D_j is computed as:

$$w_{i,j} = \frac{freq_{i,j}}{\max_k \ freq_{k,j}} \times \log \frac{N}{n_i}$$

Where $freq_{i,j}$ is the number of occurrences of I_i in D_j, $\max_k freq_{k,j}$ is the frequency of the most repeated instance in D_j, n_i is the number of documents annotated with I_i, and N is the total number of documents in the search space.

The number of occurrences of an instance in a document is primarily defined as the number of times the label of the instance appears in the document text, if the document is annotated with the instance, and zero otherwise. We realised in our first experiments that quite a number of occurrences were missed in practice with this approach, since pronouns, periphrasis, metonymy, and other deixis abound in regular written speech. Finding all the references to an individual in free text is a very complex natural language processing problem far beyond the scope of our current research. Nonetheless we have achieved significant improvements by extending our labeling scheme and exploiting class hierarchies, as follows.

First, further instance occurrences are found by adding more labels to instances. However, the proliferation of labels tends to introduce further polysemic ambiguities that lead to incorrect annotations. To avoid this negative effect, our system provides a separate *keyword* property to be used, in addition to *label*, for instance frequency computation, but not for automatic annotation. As a general rule, *label* should be reserved to clearly instance-specific text forms, leaving more ambiguous ones as *keyword*s. Since instance occurrences are only computed in the presence of an annotation, very few or no ambiguities are caused in practice.

Also, synecdoche is a frequent rhetoric figure used to avoid repetition, where an individual is referred to by its class (e.g. "the painter"), after the individual (e.g. "Picasso") has already appeared in the text. To cope with this, the list of textual forms (labels and keywords) of an instance is automatically expanded (just for the computation of occurrences) with the textual forms of its direct and indirect classes. This introduces a slight occurrence counting imprecision when more than one instance of the same class are annotating the same document, because the same class references are counted once for each instance. However, in our experiments the improvements obtained with this technique outweight the effect of the imprecision.

4 Processing Queries

Our approach to ontology-based information retrieval can be seen as an evolution of classic keyword-based retrieval techniques, where the keyword-based index is replaced by a semantic knowledge base. The overall retrieval process is illustrated in Fig. 1. Our system takes as input a formal RDQL query. This query could be generated from a keyword query, as in e.g. [8,15,18], a natural language query [4], a form-based interface where the user can explicitly select ontology classes and enter property values [1,11,12], or more sophisticated search interfaces [7]. A number of research works have undertaken the construction of easy to use user interfaces for ontology query languages, and we do not address this problem here. The RDQL query is executed against the knowledge base, which returns a list of instance tuples that satisfy the query. Finally, the documents that are annotated with these instances are retrieved, ranked, and presented to the user.

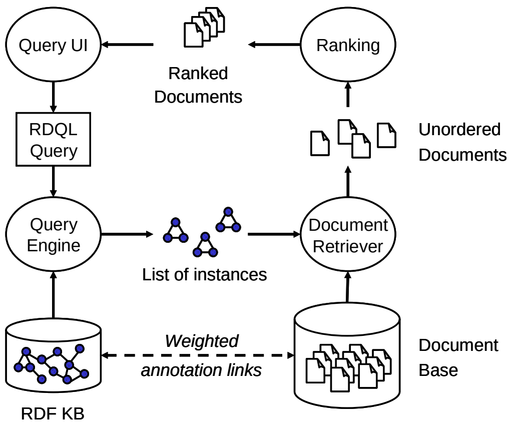

Fig. 1. Our view of ontology-based information retrieval

4.1 Query Encoding and Document Retrieval

The RDQL query can express conditions involving domain ontology instances, document properties (such as *author*, *date*, *publisher*, etc.), or classification values. E.g. "cultural articles published by the Le Monde newspaper about European movies with Canadian actors in the cast."

In classic keyword-based vector-space models for information retrieval, the query keywords are assigned a weight that represents the importance of the concept in the information need expressed by the query. Analogously, in our model, the variables in the SELECT clause of the RDQL query can be weighted to indicate the relative interest of the user for each of the variables to be explicitly mentioned in the documents. For instance, in the previous example, the user might be interested that both the movies and the Canadian actors are mentioned in the articles, or have a higher priority for either the movies or the actors. The weights can be set explicitly by the user, or be automatically derived by the system, e.g. based on frequency analysis, personalisation techniques, or other strategies [6].

Our system uses inferencing mechanisms for implicit query expansion based on class hierarchies (e.g. organic pigments can satisfy a query for colorants), and rules such as one by which the winner of a sports match might be inferred from the scoring. In fact, in our current implementation, it is the KB which is expanded by adding inferred statements beforehand.

The query execution returns a set of tuples that satisfy the query. It is the document retriever's task to obtain all the documents that correspond to the instance tuples. If the tuples are only made up of instances of domain concepts, the retriever follows all outgoing annotation links from the instances, and collects all the documents in the repository that are annotated with the instances. If the tuples contain instances of document classes (because the query included direct conditions on the documents), the same procedure is followed, but restricted to the documents in the result set, instead of the whole repository.

4.2 Ranking Algorithm

Once the list of documents is formed, the search engine computes a semantic similarity value between the query and each document, as follows. Let $O = \{I_i\}_{i=1}^{M}$ be the set of all instances in the ontology, and $\{D_i\}_{i=1}^{N}$ be the set of all documents in the search space. Let $(v_1,...,v_k)$ be the weights of the variables in the SELECT clause of the RDQL query *Q*, and let $T = \{T_i\}_{i=1}^{n}$ be the list of tuples in the query result set, where $T_i = \{T_{i,j}\}_{j=1}^{k}$, with $T_{i,j} \in O$.

We define the *document vector* of D_i as $\vec{d}_i = (d_{i,1}, \ldots, d_{i,M})$, where $d_{i,j}$ is the weight of the annotation of document D_i with I_j, if such annotation exists, and zero otherwise. We define the *extended query vector* as $\vec{q} = (q_1,...,q_M)$, where $q_l = \sum_{\exists i | I_l = T_{i,j}} v_j$, i.e. the query vector element corresponding to I_l is added the variable weight v_j if I_l is a value of the variable *j* in some tuple T_i that satisfies the query *Q*. Note that the sum rarely has more than one term, since this would mean that the same instance appears more than once in the same result set tuple. If I_l does not appear in any tuple, $q_l = 0$.

Now the similarity measure of D_i for the query *Q* is computed as:

$$sim(D_i, Q) = \frac{\vec{d}_i \bullet \vec{q}}{|\vec{d}_i| \times |\vec{q}|}$$

Because of the way $\vec{q}$ is constructed, $|\vec{q}|$ is usually quite large, and as a consequence the values of the similarity function are too low. For example, if the user queries for special offers for summer holidays in the Aegean Islands, a document that shows one such offer will get a similarity value in the order of 1/*n*, where *n* is the total number of registered offers in the knowledge base that match the query. Only a document that displays nearly all offers could get close to similarity 1. To compensate for this, in practice we use the following normalization factor instead of $|\vec{q}|$: $\sqrt{\sum_{j=1}^{k}\left(v_j^{\,2} \times \max_l \#T_j^l\right)}$, where $T_j^l = \left\{ I \in O \mid I \text{ annotates } D_l \wedge \exists i,\ I = T_{i,j} \right\}$.

If the knowledge in the KB is incomplete (e.g. there are documents about travel offers in the knowledge source, but the corresponding instances are missing in the KB), the semantic ranking algorithm performs very poorly: RDQL queries will return less results than expected, and the relevant documents will not be retrieved, or will get a much lower similarity value than they should. As limited as might be, keyword-based search could perform better in these cases. To cope with this, our ranking model combines the semantic similarity measure with the similarity measure of a keyword-based algorithm. The final value for ranking is computed as $t \times sim\,(D_i,Q) + (t - 1) \times ksim\,(D_i,Q)$, where *ksim* is computed by a keyword-based algorithm. We have taken $t = 0.5$, which seems to perform well in our experiments. As a further adjustment, if *ksim* returns 0, we take *t* = 1, and if *sim* returns 0, we take *t* = 0.2. For further testing, we have implemented a user interface where this parameter can be freely set by the user with a slider after the search has been executed, so that the user can see dynamically how the results are reranked as the value of *t* is moved.

The keywords for the *ksim* algorithm could be extracted directly from the user query, if a keyword-based or even natural language interface was used. In our current implementation we extract the keywords from the RDQL query, which is suitable for testing, and would be appropriate for a form-based interface as well. More specifically, the value of the *label* property of a) the class of all query variables for which a *rdf:type* clause is included in the query, and b) any instances explicitly appearing within the RDQL query, are taken as query keywords. In practice, since the *label* property can be multivalued, a separate property is used, which stores one of the label values, designated as a unique most common lexical form. For example, the following query:

```
SELECT  ?player ?team
WHERE   (?player rdf:type sports:Player)
        (?player sports:sport sports:Basketball)
        (?player general:nationality geog:USA)
        (?player sports:playsIn ?team)
        (?team rdf:type sports:SportsTeam)
        (?team geog:locatedIn geog:Catalonya)
```

would yield the query keywords "player", "basketball" , "USA", "team", "Catalonia".

In sum, our method improves keyword-based search (actually outperforms it, as is shown in the next section) when the relevant information is available in the KB, and relies on keyword-based search otherwise.

5 Early Experiments

We have tested our system with a document base taken from an online newspaper archive [2]. For this application, the document class hierarchy includes *News* (subclass of *TextDocument*), *Photograph* and *CustomGraphic* (subclasses of *MediaDocument*). Only one classification taxonomy is used, based on the IPTC Subject Reference System, with which all documents and domain classes are classified, as explained in Section 3.1. Our current implementation is compatible with both RDF and OWL.

Building appropriate domain ontologies and a complete KB for a newspaper archive is an enormous undertaking, or would need very advanced semi-automatic knowledge extraction techniques that are not available yet in current state of the art. However, as stated in previous sections, our system tolerates incomplete ontologies and KBs. We have built three reduced domain ontologies for testing purposes, corresponding to the Culture, Economy, and Sports domains, with classes such as *Artist*, *Painter*, *Monument*, *Company*, *Bank*, *Sportsman*, *SportsTeam*, *Stadium*, etc., and a few instances of each class. These ontologies were built by reading 200 news articles, and defining classes and instances by hand for concepts found in the documents. In total, 143 domain classes and 1,144 instances were created. We have also manually set labels and keywords for concept classes and instances. Then we have run the automatic annotation and weighting algorithm over a subset of the archive comprising 2,039 news articles, which generated 3,471 annotations, of which 349 were manually created.

Once the KB was built, we tested the retrieval algorithm with some examples, and compared it to a keyword-only search, using the Jakarta Lucene library.[2] We report next the observed results in four examples, showing different levels of performance of our method in different cases. The metrics are based on a manual ranking of all documents for each query, on a scale from 0 to 5. In the experiments, all the query variables were given a weight of 1. The measurments are subjective and limited, yet indicative of the degree of improvement that can be expected, and in what cases, with respect to a keyword-based engine. The retrieval times are too low to draw any significant observation regarding efficiency. The results are shown in Fig. 2.

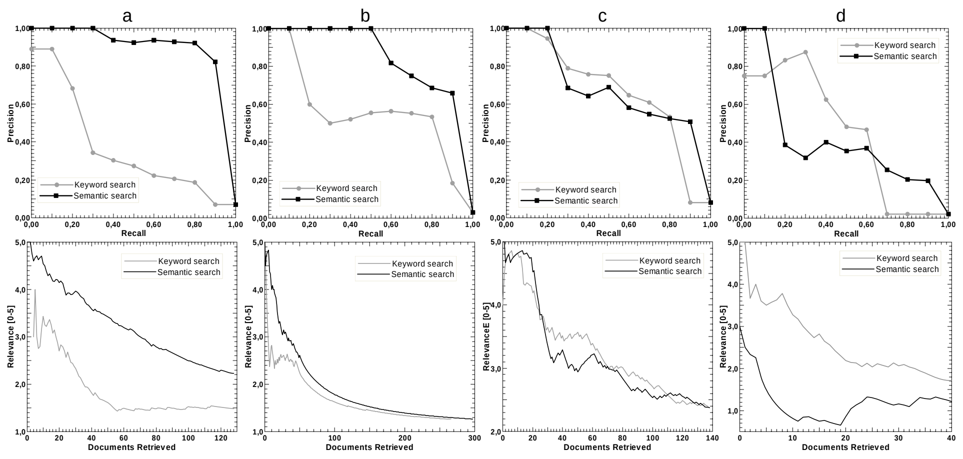

Fig. 2. Evaluation of ontology-based search (combined with keyword-based) against keyword-based only. The performance of both algorithms are shown for four different queries a, b, c, and d. The graphics on top show the precision vs. recall figures (as defined in e.g. [16]), and the graphics below show the average relevance at different document cutoff values, for each query

Query a. “News about players from USA playing in basketball teams of Catalonia.”
In this example the semantic retrieval algorithm outperforms keyword-based search because the KB contains many instances of basketball players and teams, some of which match the query. Keyword-based search only recognizes a document as relevant if it contains words like “player”, “USA”, “Catalonia”, whereas the semantic search retrieves news about players and teams as soon as the name of the player or the team are mentioned in the document.

These are typical results when a search query involves a region of the ontology with some degree of completeness in terms of instances and annotations. These cases yield a high precision up to almost maximum recall. On the other hand, the relevance graph shows that here the semantic search gives high ranks to the relevant documents. For instance, the top 20 retrieved documents have a mean relevance value of 4.2 upon 5, versus 2.7 in the keyword search.

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

However, the KB does not contain *all* teams and players, which explains the collapse of the precision at 100% recall. If more instances were created, precision values would stand at high levels for all the recall values.

Query b. "News about sports team presidents."
In this example, the ontology KB has only a few instances of sports team presidents, so not all documents relevant to the query are annotated. This causes precision to drop to lower values when recall increases. Although the total recall of semantic search is low, it still has a good precision for the top-ranked documents, which are the few ones annotated with instances in the KB. A few more documents where semantic search alone fails are still given a high ranking thanks to the combination with keyword-search, which shows here a comparable behavior to example a.

Query c. "News about basketball players."
In this case the performance of the two algorithms is similar. For this example, we have intentionally removed most instances of players from the KB, leaving a relatively low number. Moreover, we have removed all lexical variants in the *label* and *keyword* properties of player instances, except the player's surname. As a consequence, many annotations are missing. Under these conditions, the semantic model alone performs much worse than keyword-based search. However, the combined search yields a similar final behavior to keyword-based search.

Query d. "News about the European Union."
This example shows a case where our method fails. The KB contains an instance for the European Union, with all the possible syntactic variants (in Spanish "UE", "U.E.", etc.). The problem is that many Catalan sports teams have the word "UE" (acronym for "Sports Union" in Catalan) in their names. If the KB contained these teams, the disambiguation algorithm would solve the problem by favoring the sports interpretation, whenever appropriate, because "UE" is part of a longer matching string (the team name). In other examples where the labels could be totally coincident, the system would use the classification of news and instances as context information for disambiguation. But because many such teams are missing in the KB, the automatic annotator incorrectly annotates the sports news with the European Union concept, and the retriever returns them. So far, the keyword-based search behaves similarly. But the semantic ranking places these wrong documents in a top position, whereas the keyword-based model does not rank them particularly higher than the correct documents.

It can be seen that it is the automatic annotator, and not the retrieval system, which is failing here in the absence of the appropriate instances needed to solve ambiguities. One way to reduce the negative impact of incorrect annotations would be to introduce a factor in the automatic weighting algorithm that accounts for the proximity of the respective classifications of the documents and the instances. In this example, although it is difficult to avoid annotating with the European Union concept the news about Catalan teams whose name contains "UE" (in fact, some sports news could properly mention the EU), at least the weight of the annotation would be reduced because the classifications (*Geography* and *Politics* vs. *Sports*) do not match. Testing this and other possible improvements to the automatic annotation strategies are one of our planned tasks for the immediate future.

6 Discussion

The added value of semantic information retrieval with respect to traditional keyword-based retrieval, as envisioned in our approach, relies on the additional explicit information – type, structure, relations, classification, and rules, about the concepts referenced in the documents, represented in an ontology-based KB, as opposed to classic flat keyword-based indices. Semantic search introduces an additional step with respect to classic information retrieval models: instead of a simple keyword index lookup, the semantic search system processes a semantic query against the KB, which returns a set of instances. This can be seen as a form of query expansion, where the set of instances represent a new set of query terms, leading to higher recall values. Further implicit query expansion is achieved by inference rules, and exploiting class hierarchies. The rich concept descriptions in the KB provide useful information for disambiguating the meaning of documents.

In summary, our proposal achieves the following improvements with respect to keyword-based search:

- Better recall when querying for class instances. For example, querying for "presidents of the Spanish government" would return documents that mention *José Luis Rodríguez Zapatero* and other former presidents, even if the words "president", "Spanish" and "government" are not present in the documents.
- Better precision by using structured semantic queries. Structured queries allow expressing more precise information needs, leading to more accurate answers. For instance, in a keyword-based system, it is not possible to distinguish a query for USA players in Catalan basket teams vs. Catalan players in USA teams, which is possible with a semantic query.
- Better precision by using query weights. Variables with low weights are only used to impose conditions on the variables which really matter. For example, the user can search for news about USA players in Catalan teams, regardless of whether the news mention the team at all, or the nationality of the player.
- Better recall by using class hierarchies and rules. For example, a query for *WaterSports* in Spain would return results in *ScubaDiving, Windsurf,* and other subclasses, in *Cádiz, Málaga, Almería,* and other Spanish locations (by transitivity of *locatedIn*).
- Better precision by reducing polysemic ambiguities using instance labels and classifications of concepts and documents.

As explained and shown along this paper, the degree of improvement of our semantic retrieval model depends on the completeness and quality of the ontology, the KB, and the concept labels. For the sake of robustness, the system resorts to keyword-based search when the KB returns poor results.

The combination of keyword ranking and semantic ranking is tricky. We have observed that occasionally a good semantic ranking score is spoiled by a low keyword-based value. A simple solution would be to set a minimum threshold for the keyword-based score to be counted. Anyhow, these cases, albeit infrequent, suggest that more sophisticated methods than the linear combination of both values should be researched to improve our initial results.

7 Conclusion

The research presented here started as a continuation of our previous work on the construction, exploitation, and maintenance of domain-specific KBs using Semantic Web technologies [1,2]. While some basic semantic search facilities were included in these prior proposals, room for improvement was acknowledged, because the level of semantic detail was insufficient, since it was essentially based on types of documents and taxonomic classifications. The aim of our current work is to provide better search capabilities which yield a qualitative improvement over keyword-based full-text search, by introducing and exploiting finer-grained domain ontologies.

Our approach can be seen as an evolution of the classic vector-space model, where keyword-based indices are replaced by an ontology-based KB, and a semi-automatic document annotation and weighting procedure is the equivalent of the keyword extraction and indexing process. We show that it is possible to develop a consistent ranking algorithm on this basis, yielding measurable improvements with respect to keyword-based search, subject to the quality and critical mass of metadata. Our proposal is an adaptation of the vector-based ranking model that takes advantage of an ontology-based knowledge representation.

Our proposal inherits all the well-known problems of building and sharing well-defined ontologies, populating knowledge bases, and mapping keywords to concepts. Recent research on these areas is yielding promising results [5,11]. It is our aim to provide a consistent model by which any advancement on these problems is played to the benefit of semantic search improvements.

Further experimentation, larger KBs, and larger document sets are needed to test and improve our model. For instance, our annotation weighting scheme is not taking advantage yet of the different relevance of structured document fields (e.g. title is more important than body). Annotating documents with statements, besides instances, is another interesting possibility to experiment with. Also, we are currently extending our model with a profile of user interests for personalised search [6].

8 Acknowledgements

This research was supported by the European Commission under contract FP6-001765 aceMedia. The expressed content is the view of the authors but not necessarily the view of the aceMedia project as a whole. The authors would like to thank the reviewers for their detailed, accurate and helpful coments.

9 References

1. Castells, P., Foncillas, B., Lara, R., Rico, M., Alonso, J. L.: Semantic Web Technologies for Economic and Financial Information Management. In: Davies, J., Fensel, D., Bussler, C., Studer, R. (eds.): The Semantic Web: Research and Applications – 1st European Semantic Web Symposium (ESWS 2004). Lecture Notes in Computer Science, Vol. 3053. Springer Verlag, Berlin Heidelberg New York (2004) 473-487

2. Castells, P., Perdrix, F., Pulido, E., Rico, M., Benjamins, V. R., Contreras, J., Lorés, J.: Neptuno: Semantic Web Technologies for a Digital Newspaper Archive. In: Davies, J., Fensel, D., Bussler, C., Studer, R. (eds.): The Semantic Web: Research and Applications – 1st European Semantic Web Symposium (ESWS 2004). Lecture Notes in Computer Science, Vol. 3053. Springer Verlag, Berlin Heidelberg New York (2004) 445-458
3. Christophides, V., Karvounarakis, G., Plexousakis, D., Scholl, M. and Tourtounis, S.: Optimizing taxonomic semantic web queries using labeling schemes. Journal of Web Sematics 1, Issue 2, Elsevier (2003) 207-228
4. Contreras, J., Benjamins, V. R., Blázquez, M., Losada, S., Salla, R. et al: A Semantic Portal for the International Affairs Sector. In: Motta, E., Shadbolt, N., Stutt, A., Gibbins, N. (eds.): Engineering Knowledge in the Age of the Semantic Web – 14th Intl. Conference on Knowledge Engineering and Knowledge Management (EKAW 2004). Lecture Notes in Computer Science, Vol. 3257. Springer Verlag, Berlin Heidelberg New York (2004) 203-215
5. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R. et al: A Case for Automated Large Scale Semantic Annotation. Journal of Web Sematics 1, Issue 1, Elsevier (2003) 115-132
6. Gauch, S., Chaffee, J., and Pretschner, A.: Ontology-based personalized search and browsing. Web Intelligence and Agent System 1, Issue 3-4 (2003) 219-234
7. Guarino, N., Masolo, C., Vetere, G.: OntoSeek: Content-Based Access to the Web. IEEE Intelligent Systems 14, Issue 3 (1999) 70-80
8. Guha, R. V., McCool, R., Miller, E.: Semantic search. In Proc. of the 12th Intl. World Wide Web Conference (WWW 2003), Budapest, Hungary, (2003) 700-709
9. Handschuh, S., Staab, S., and Ciravegna, F.: S-cream – Semi-automatic Creation of Metadata. In: A. Gómez-Pérez , V. Richard Benjamins (eds.): 13th Intl. Conf. on Knowledge Engineering and Knowledge Management – Ontologies and the Semantic Web (EKAW'02). LNCS, Vol. 2473. Springer Verlag, Berlin Heidelberg New York (2002) 358-372
10. Järvelin, K., Kekäläinen, J., and Niemi, T.: ExpansionTool: Concept-based query expansion and construction. Information Retrieval 4, Issue 3-4 (2001) 231-255
11. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic Annotation, Indexing, and Retrieval. Journal of Web Sematics 2, Issue 1, Elsevier (2004) 47-49
12. Maedche, A., Staab, S., Stojanovic, N., Studer, R., Sure, Y.: SEmantic portAL: The SEAL Approach. In: Fensel, D., Hendler, J. A., Lieberman, H., Wahlster, W. (eds.): Spinning the Semantic Web. MIT Press, Cambridge London (2003) 317-359
13. Mayfield, J., and Finin, T.: Information retrieval on the Semantic Web: Integrating inference and retrieval. In: Workshop on the Semantic Web at the 26th Intl. ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada (2003)
14. Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A.: KIM – A Semantic Platform for Information Extaction and Retrieval. Journal of Natural Language Engineering 10, Issue 3-4, Cambridge University Press (2004) 375-392
15. Rocha, C., Schwabe, D., de Aragão, M. P.: A Hybrid Approach for Searching in the Semantic Web. In Proc. of 13h Intl. World Wide Web Conf. (WWW 2004), NY (2004) 374-383
16. Salton, G. and McGill, M. Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
17. Sheth, A., Bertram, C., Avant, D., Hammond, B., Kochut, K., and Warke, Y.: Managing Semantic Content for the Web. IEEE Internet Computing 6, Issue 4 (2002) 80-87
18. Stojanovic, N.: On Analysing Query Ambiguity for Query Refinement: The Librarian Agent Approach. In: Song, I.-Y.; Liddle, S.W.; Ling, T.W.; Scheuermann, P. (eds.): Conceptual Modeling – ER 2003, 22nd Intl. Conf. on Conceptual Modeling. Lecture Notes in Computer Science, Vol. 2813. Springer Verlag, Berlin Heidelberg New York (2003) 490-505
19. Stojanovic, N., Studer, R., and Stojanovic, L.: An Approach for the Ranking of Query Results in the Semantic Web. In: Fensel, D., Sycara, K. P., Mylopoulos, J. (eds.): The Semantic Web – ISWC 2003, 2nd Intl. Semantic Web Conference. Lecture Notes in Computer Science, Vol. 2870. Springer Verlag, Berlin Heidelberg New York (2003) 500-516