

# An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval

Pablo Castells, Miriam Fernández, and David Vallet

**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 knowledge bases to improve search over large document repositories. In our view of Information Retrieval on the Semantic Web, a search engine returns documents rather than, or in addition to, exact values in response to user queries. For this purpose, our approach includes an ontology-based scheme for the semiautomatic 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 conventional keyword-based retrieval to achieve tolerance to knowledge base incompleteness. Experiments are shown where our approach is tested on corpora of significant scale, showing clear improvements with respect to keyword-based search.

**Index Terms**—Information retrieval models, ontology languages, semantic search, semantic Web.

## 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 1990s. One way to view a semantic search engine is as a tool that gets formal ontology-based queries (e.g., in RDQL [32], RQL [19], SPARQL [29], etc.) from a client, executes them against a knowledge base (KB), and returns tuples of ontology values that satisfy the query [3], [4], [6], [24]. These techniques typically use Boolean search models based on an ideal view of the information space as consisting of nonambiguous, nonredundant, formal pieces of ontological knowledge. In this view, the information retrieval (IR) 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 percent precise, and there is no notion of an approximate answer to an information need. While this conception of semantic search brings key advantages already, our work aims at taking a step beyond.

A purely Boolean ontology-based retrieval model makes sense when the whole information corpus can be fully represented as an ontology-driven knowledge base. But, there are well-known limits to the extent to which knowledge can be formalized this way. First, because of the huge amount of information currently available 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 knowledge atoms inevitably implies a loss of information value, and although it may be useful to break documents down into smaller information units that can be reused and reassembled to serve different purposes, it is 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 of results. Boolean search does not provide clear ranking criteria, without which the search system may become useless if the retrieval 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 [31] 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 [10],

• The authors are with the Escuela Politécnica Superior, c/ Tomas y Valiente 11, Universidad Autónoma de Madrid, Ciudad universitaria de Cantoblanco, 28049 Madrid, Spain.  
E-mail: {pablo.castells, miriam.fernandez, david.vallet}@uam.es.

Manuscript received 2 Oct. 2005; revised 16 Apr. 2006; accepted 2 June 2006; published online 19 Dec. 2006.

For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0456-1005.

[17], [20], [28]. Until, 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 midterm. Consequently, 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.

The rest of the paper is organized 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 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 conclusions are given.

## 2 STATE-OF-THE-ART

Conceptual search, i.e., search based on meaning rather than just character strings, has been the motivation of a large body of research in the IR field long before the Semantic Web vision emerged [1], [18]. This drive can be found in popular and widely explored areas such as Latent Semantic Indexing [9], [23], linguistic conceptualization approaches [14], [25], or the use of thesaurus and taxonomies to improve retrieval [5], [12], to name a few. Such proposals are commonly based on shallow and sparse conceptualizations, usually considering very few different types of relations between concepts and low information specificity levels. Compared to these, our proposal considers a much more detailed and densely populated conceptual space in the form of an ontology-based KB. An obvious, immediate trade-off of our approach is that such a rich conceptual space is more difficult and expensive to obtain, but this is one of the major targets addressed by the Semantic Web research community, which is already providing significant results and dependable grounds to build upon [10], [28], as we propose here.

Our approach can be seen as combining the flexibility and generality of an IR model for unstructured search spaces, with the expressiveness and detail of a structured relational model describing some of the knowledge involved in the unstructured information space, in a structured and formal way, with powerful and precise data querying facilities. But, rather than the conventional relational technologies, we rely on an ontology-based approach, which compared to the latter enables further inferencing capabilities [13] that can be exploited to enhance the retrieval process. Furthermore, the ontological view is particularly well suited for the development of scaleable interoperability bridges between heterogeneous systems [27], which is of key importance in large-scale networked environments (e.g., the WWW). By building upon an ontology-based layer, our model can benefit from semantic data integration facilities such as the ones proposed in [22], [27] for the combination of heterogeneous schemas.

Our view of the semantic retrieval problem is very close to the latest proposals in KIM [20], [28]. While KIM focuses on automatic population and annotation of documents, our

work focuses on the ranking algorithms for semantic search. Along with TAP [16], 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 [20], 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. In fact, we have used the public results available from the KIM project as a testbed for our retrieval model (see Section 5).

TAP [16] 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 1) the development of a distributed query infrastructure for ontology data in the Semantic Web and 2) 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 SPARQL [29], RDQL [32], etc. The supported search capability is limited to keyword search within the "title properties" of instances, and no ranking is provided.

Mayfield and Finin [26] 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 missing information (e.g., by way of the transitivity of the *locatedIn* relationship over geographical locations, we may infer that Verona is located in Europe because it is located in Italy, which is located in Europe).

Semantic Portals [3], [4], [6], [24] typically provide simple search functionalities that may be better characterized 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 [6], but neither the instances, nor the documents are ranked. Maedche et al. do provide a criterion for query result ranking in the SEAL Portal [24], 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.

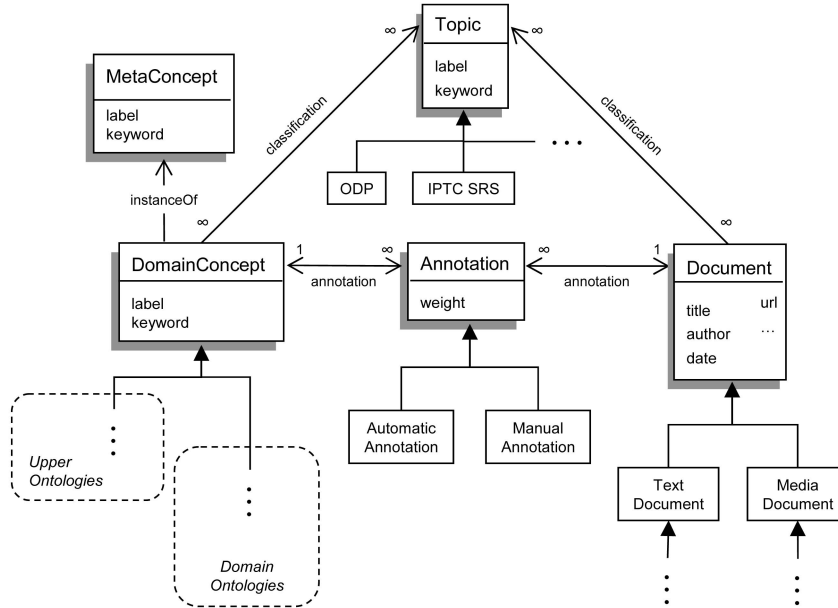


Fig. 1. Root ontology classes.

The ranking problem has been taken up again in [36], and, more recently, [30]. 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 [30]. 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 [36]. 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 with 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. Our system can work with any arbitrary domain ontology with essentially no restrictions, except for some minimal requirements, which basically consist of conforming to a set of root ontology classes, which are described in Section 3.1.

The concepts and instances in the KB are linked to the documents by means of explicit, nonembedded annotations to the documents. While we do not address here the problem of knowledge extraction from text [6], [10], [17], [20], [28], we provide a vocabulary and some simple mechanisms to aid in the semiautomatic annotation of documents, once ontology instances have been created (manually or automatically). These are described in Section 3.2. Domain ontology and KB engineering and management are complex tasks by themselves that have been addressed at length in a large body of research in the Semantic Web and Knowledge Engineering areas (see, e.g., [7], [13], [34], and our previous work [3], [4]), but they are not the focus of this paper. In our current experiments, we have reused the public KIM ontology and KB, available from the KIM project [20], to test our retrieval model (see Section 5). The KIM corpus has been built automatically to a great extent (see [28]), with a fair amount of human supervision. We have extended it manually with a few further classes and instances for testing purposes. We have reused other lower-scale ontologies and data sets as well, which were built in our previous research [37]. One of them will be used for a detailed example in Section 4.3.

#### 3.1 Root Ontology Classes

The conceptualization of the information/knowledge space in our approach is embodied as a set of root classes that are described next. Our system requires that the knowledge base be constructed from three main base classes: *DomainConcept*, *Topic*, and *Document* (see Fig. 1). *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 supplied, 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*, and *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 (in our current implementation, a URL) from which the actual document contents can be retrieved.

*Topic* is the root for class hierarchies that are merely used as classification schemes and are never instantiated. These taxonomies can be any of the ones commonly associated to collections of documents, such as Open Directory Project<sup>1</sup> (ODP) on the WWW, or the ones used in digital or physical libraries and online catalogs (e.g., the Dewey Decimal System<sup>2</sup>). Our system can import any such standard or application-specific classification hierarchy by just making it available in a compatible format for our implementation (e.g., as an RDF class hierarchy). Taxonomies are used in our system as a terminology to annotate documents and concept classes, by assigning them as values for dedicated properties. For instance, in a KB for news, classes like *Culture*, *Politics*, *Economy*, *Sports*, etc. (after the IPTC Subject Reference System<sup>3</sup> standard), 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*, *Topic*, and *Document* arises from our own experience in previous Semantic Web projects [3], [4] and many other observed information systems where this or a similar distinction seems to be natural, useful, and recurrent (see, e.g., [33]). 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 nonembedded annotations. In many respects, our scheme for semiautomatic annotation is similar to the one recently reported in [20]. For convenience, annotations are represented as an extension of the ontology, but they could be implemented by any other means, as they are not a proper part of the domain knowledge. In fact, as an optimization in our current implementation, we store them in a separate

relational database. However, the availability of the annotations in ontological form has advantages such as being able to use the same tools and environments for browsing and correcting annotations as are used for editing the KB itself, and other simplifications at the implementation level. To any extent, we shall use the ontological notation here as a means to describe the structures and entities involved in the annotation of documents and how this information is organized.

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 semiautomatically. The subclasses *ManualAnnotation* and *AutomaticAnnotation* are used, respectively, to differentiate each case. We have found this distinction useful for the system at least because 1) manual annotations are more reliable than automatic ones, and when available should prevail, and 2) while automatic annotations can be deleted for recalculation, manual annotations should be preserved.

Our system provides a simple facility for semiautomatic 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 [20], [28] and TAP [16]. The value of this property can be set by hand by an ontology designer, or by semiautomatic means, if an external instance generation system is plugged to our system. In our experiments, reported in Section 5, we have reused the automatic concept to label mapping available from the KIM KB, which we have complemented manually to add our own extensions. The semiautomatic techniques and heuristics, based on natural language processing, by which concepts are bound to strings in KIM are described at length in [20] and [28]. Similarly to KIM, once this mapping is available, instance labels are used by the automatic annotator of our system 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 polysemy, 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." The principle behind this is that a longer string is assumed to carry more specific information, which takes precedence over a more general and common meaning (indeed, if a string *a* contains string *b*, then *a* occurs necessarily less or as frequently as *b*, and selecting *a* brings better accuracy in most, if not all cases). Second, classification taxonomies are used as a source of semantic context 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

1. <http://dmoz.org>.

2. <http://www.oclc.org/dewey>.

3. <http://www.iptc.org/NewsCodes>.

has the closest classification to the document is chosen. For example, the word “Irises” in a document 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. Since human supervision highly improves the accuracy of annotations, yet manually revising millions of annotations is unrealistic, after running the automatic annotation process, the system presents a reasonably short list of most uncertain annotations, to be confirmed or rejected by a domain expert. These include, for instance, unsolved polysemies and annotations where the concepts and the documents do not have any common classification category, an indication that the right concept corresponding to the proper sense of a word might be missing from the KB.

Our semiautomatic annotation mechanisms can be further improved, but this is out of the extent of our ongoing research. More sophisticated annotation techniques, as have been reported in the literature [10], [17], [20], [28], could be reused and integrated here in a complementary and modular way, to the benefit of 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 [31]. 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 [31], based on the frequency of occurrence of the instances in each document. More specifically, the weight  $d_x$  of an instance  $x$  for a document  $d$  is computed as:

$$d_x = \frac{freq_{x,d}}{\max_y freq_{y,d}} \cdot \log \frac{|\mathcal{D}|}{n_x},$$

where  $freq_{x,d}$  is the number of occurrences in  $d$  of the keywords attached to  $x$ ,  $\max_y freq_{y,d}$  is the frequency of the most repeated instance in  $d$ ,  $n_x$  is the number of documents annotated with  $x$ , and  $\mathcal{D}$  is the set of all 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 realized 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 (i.e., an instance) in free text is a very

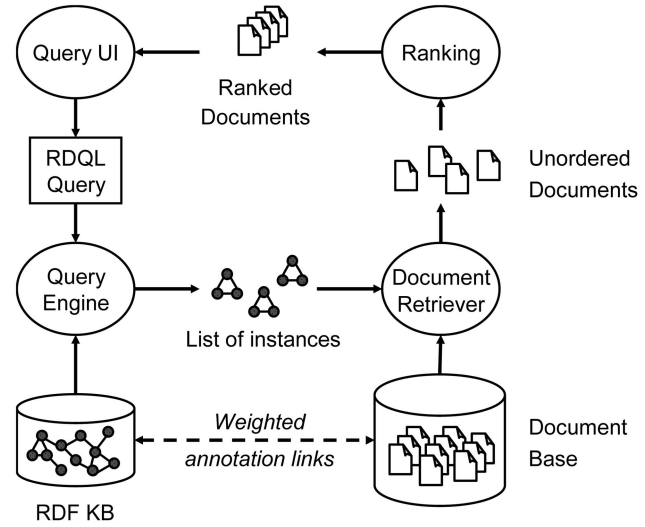


Fig. 2. Our view of ontology-based information retrieval.

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 *keywords*. 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. For example, if “van Gogh” and “Gauguin” are cited in the same text, a reference such as “the painter” will be inaccurately counted in our current implementation as an occurrence of both painters. However, in our experiments, the improvements obtained with this technique outweigh 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. 2 and consists of the following steps: Our

system takes as input a formal RDQL query. This query could be generated from a keyword query, as in e.g., [16], [30], [35], a natural language query [6], a form-based interface where the user can explicitly select ontology classes and enter property values [3], [20], [24], or more sophisticated search interfaces [15]. 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. This step of the process is purely Boolean (i.e., based on an exact match), so that the returned instances must strictly hold all the conditions in the formal query. Rather than proposing a new approach for this operation, we reuse state-of-the-art techniques for the execution of the formal query by a standard ontology-based query engine, such as the ones packaged with popular ontology processing libraries like Jena,<sup>4</sup> Sesame,<sup>5</sup> etc. In our current implementation, we are using the Jena toolkit. Note, however, that it is possible to further elaborate on the approach at this point toward a nonstrictly Boolean model. For instance, as a supplementary enhancement, the conditions of the query could be relaxed to improve recall when not enough results are returned. Disjunctive variants of a conjunctive query could be formed in such cases, in a way that the number of conditions held determines the ranking of the result set tuples. Query variable weights (described in the next section) could be used here as auxiliary hints to decide how strictly or flexibly each query condition should be imposed. Though we plan to explore these extensions as future work, they are not developed in our current system, where formal queries are processed by a standard Boolean engine.

Finally, the documents that are annotated with the instances returned in the previous step are retrieved, ranked, and presented to the user. In contrast to the formal query execution, the document retrieval phase is based on an approximate match since the relation between a document and the concepts that annotate it has an inherent degree of fuzziness and incompleteness: On the one hand, each annotation has a different importance for the document (as represented by the weight of the annotation), and on the other, we assume that the annotations do not describe all the meaning conveyed by the document in a complete way, but rather a subset of it, as stated in the Introduction. Indeed, the approach to automatic annotation described in Section 3.2 results in a description of the concepts mentioned or appearing in a document, but the annotations do not describe what is said about these concepts in the documents. In this setting, the definition of an effective retrieval model upon which the documents can be filtered and ranked by relevance to the original formal ontology-based query is an open research problem, which is addressed by the techniques described in the next

sections, building upon the annotation weighting scheme proposed in Section 3.3.

#### 4.1 Query Encoding and Document Retrieval

The RDQL queries supported by our model can express conditions involving domain ontology instances, document properties (such as *author*, *date*, *publisher*, etc.), or classification values, i.e., “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, or its discriminating power for discerning relevant from irrelevant documents. 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 concept frequency analysis, user preferences, or other strategies [31]. In our experiments, the weights are assigned to 1 by default. For testing purposes, the user interface for the experiments provides one slider per variable with which the weights can be manually set from 0 to 1.

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 the 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  $\mathcal{O}$  be the set of all classes and instances in the ontology, and  $\mathcal{D}$  be the set of all documents in the search space. Let  $q \in \mathcal{Q}$  be an RDQL query, let  $V_q$  be the set of variables in the SELECT clause of  $q$ , let  $w$  be the weight vector for these variables, where for each  $v \in V_q$ ,  $w_v \in [0, 1]$ . Let  $T_q \subset \mathcal{O}^{|V_q|}$  be the list of tuples in the query result set, where for each tuple  $t \in T_q$  and each  $v \in V_q$ ,  $t_v \in \mathcal{O}$ .

4. <http://jena.sourceforge.net>.

5. <http://www.openrdf.org>.

We represent each document in the search space as a *document vector*  $d \in \mathcal{D}$ , where  $d_x$  is the weight of the annotation of the document with concept  $x$  for each  $x \in \mathcal{O}$ , if such annotation exists, and zero otherwise. We define the *extended query vector*<sup>6</sup> as given by  $q_x = \sum_{\exists t \in T_q, t_v=x} w_v$ , i.e., the query vector element corresponding to  $x$  is added the variable weight  $w_v$  if there is a tuple  $t$ , where  $t_v = x$  (even if there is more than one such tuple,  $w_v$  is not added more than once for the same  $v$  and  $x$ ). Note that the sum rarely has more than one term since this would mean that the same instance appears as a satisfying value for different variables in different (or the same) result set tuples. If  $x$  does not appear in any tuple, we set  $q_x = 0$ .

Now, the similarity measure between a document  $d$  and the query  $q$  is computed as:

$$\text{sim}(d, q) = \frac{d \bullet q}{|d| \bullet |q|}.$$

Because of the way  $q$  is constructed,  $|q|$  is usually quite large, and the values of  $\text{sim}(d, q)$  are quite low. For example, if the user queries for special offers for summer holidays in the Aegean Islands, it can be seen that 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 would display nearly all offers could get close to similarity 1. This potential problem is solved by a normalization of the similarity scores that is part of the final retrieval step, explained next.

### 4.3 Combination with Keyword-Based Retrieval

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 that might be, keyword-based search will likely 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.

Combining the output of several search engines has been a widely addressed research topic in the IR field [8], [21]. After testing several approaches, we have selected the so-called CombSUM strategy, which has also been found to be among the most simple and effective in prior work, and consists of computing the combined ranking score by a linear combination of the input scores. That is, in our case, the final score is  $\lambda \cdot \text{sim}(d, q) + (1 - \lambda) \text{ksim}(d, q)$ , where  $\text{ksim}$  is computed by a keyword-based algorithm, and  $\lambda \in [0, 1]$ . We have taken  $\lambda = 0.5$ , which seems to perform well in our experiments. As a further adjustment, if  $\text{ksim}$  returns 0, we take  $\lambda = 1$ , and if  $\text{sim}$  returns 0, we take  $\lambda = 0.2$ . 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  $\lambda$  is moved. Obviously, for the combination of scores to make sense, the scores have to be first made comparable, which involves a normalization step. For this purpose, we use our own optimized normalization method, which not only scales the scores to the same range (the  $[0, 1]$  interval) as other standard approaches proposed in the literature do [21], but, moreover, undoes potential biases in the distribution of the scores (see [11] for further details).

The automatic creation of a keyword-based query, to be combined with the results of the semantic query, remains to be explained. The keywords for the  $\text{ksim}$  algorithm could be extracted directly from the user query, if a keyword-based or even natural language interface is used. In our current implementation, we extract the keywords from the RDQL query, which is suitable enough for our present tests, and would be appropriate for a form-based query interface as well. More specifically, the value of the *label* property of 1) the class of all query variables for which a *rdf:type* clause is included in the query and 2) any instances explicitly appearing within the RDQL query are taken as query keywords. For example, the following query:

```
SELECT    ?company
WHERE     (?company <rdf:type> <kb:Company>)
          (?company <kb:activeInSector> <kb:FoodSector>)
          (?company <kb:locatedIn> <kb:USA>)
          (?company <kb:income> ?income)
AND       ?income > 3000000
```

would yield the query keywords “company,” “Food, Beverage & Tobacco,” “located in,” “USA,” “net income,” “greater than,” “3,000,000.” 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.

### 4.4 Example

In order to illustrate our model, consider the query “Players from USA playing in basketball teams of Catalonia.” This would be formalized as:

```
SELECT    ?player ?team
WHERE     (?player <rdf:type> <kb:SportsPlayer>)
          (?player <kb:plays> <kb:Basketball>)
          (?player <kb:nationality> <kb:USA>)
          (?player <kb:playsIn> ?team)
          (?team <kb:locatedIn> <kb:Catalonia>)
```

Assume that in order to give higher priority to the players themselves, a weight of 1.0 is assigned to the variable  $?player$ , and a weight of 0.5 to the  $?team$  variable. We have run this query against a reduced sample document set taken from a regional Spanish newspaper archive, using a small KB containing knowledge about sports in Spain (see [37] for the details of how this KB was set up), which returns the following tuples:

<u>Player</u>	<u>Team</u>
Aaron Jordan Bramlett	Caprabo Lleida
Derrick Alston	Caprabo Lleida
Venson Hamilton	DKV Joventut
Jamie Arnold	DKV Joventut

6. Without loss of generality, we shall use the same symbol for the query and the corresponding query vector. Likewise, we identify a document with its document vector.

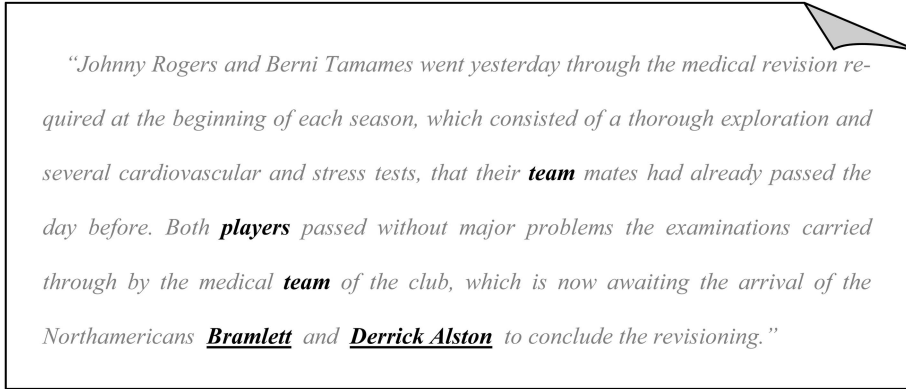


Fig. 3. First result for “Players from USA playing in basketball teams of Catalonia.”

Therefore, the query vector  $q$  has a value of 1 for the vector-space “axis” defined by the four player instances, and a value of 0.5 for the two teams.

The second step of the retrieval algorithm retrieves 66 news articles ranked from 0.1 to 0.52. As an example, the document in the top of the result list is shown in Fig. 3. The document is annotated by the instances that represent the players *Aaron Bramlett* and *Derrick Alston* shown in underlined italic in the text. The weights computed for the annotations are 1.73 and 1.65, respectively, and thereby, the document vector  $d$  has these values in the coordinates that correspond to the players, and 0 anywhere else. The resulting rank value for the document is  $\text{sim}(d, q) = 0.12$ , and the score computed by the keyword-based algorithm is  $\text{ksim}(d, q) = 0.06$ , resulting from the occurrence of the keywords shown in bold in the text. These scores are normalized to 0.63 and 0.41, respectively, so that the combined rank value is 0.52.

## 5 EXPERIMENTS

We have tested our system on a corpus of 145,316 documents (445 MB) from the CNN Web site.<sup>7</sup> We have used the KIM domain ontology and KB [20], publicly available as part of the KIM Platform, developed by Ontotext Lab,<sup>8</sup> with minor extensions and adjustments to conform to our top-level metamodel described in Section 3. We have also manually added classes and instances in areas where the KIM KB fell short (such as the Sports domain), in order to support a larger testbed for experimentation. Only one classification taxonomy is used, based on the categories of the CNN archive (such as *Business*, *Politics*, *Sports*, etc., and subcategories thereof), 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. The complete KB includes 281 classes, 138 properties, 35,689 instances, and 465,848 sentences, taking a total of 71 MB in RDF text format. For efficiency, the KB has been stored on a MySQL backend using Jena 2.2. Based on the concept-keyword mapping available in the KIM KB, over  $3 \cdot 10^6$  annotations (i.e., over 25 per document on average)

are automatically generated by the techniques described in Section 3.2.

Once the experimental setting has been set up, we have tested the retrieval algorithm with some examples, and compared it to a conventional keyword-only search, using the Jakarta Lucene library.<sup>9</sup> Although a systematic efficiency testing has not yet been conducted, the average informally observed response time on a standard professional desktop computer is below 30 seconds. A main bottleneck in our first implementation was the traversal of annotations to retrieve the document vectors, the cost of which grows linearly with the size of the result sets ( $|T_q|$  and  $|R_q|$ , where  $R_q = \{d \in \mathcal{D} \mid \text{sim}(d, q) > 0\}$ ). This was drastically reduced by storing the annotations in a separate database.

Since the semantic queries to our system can be arbitrarily complex and should be based on the available ontologies, it is not clear how a meaningful random query generation procedure could be put in place. Therefore, a set of 20 queries was prepared manually for the comparative performance measurement. We report and discuss next the observed results on three examples selected among those 20, showing different levels of performance for different characteristic cases, where we have intentionally chosen examples where the ontology-based method does not always return the best results. The overall performance over the 20 queries is also shown on average over the whole test set. 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 measurements 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 results are shown in Fig. 4.

**Query a.** “News about banks that trade on NASDAQ, with fiscal net income greater than two billion dollars.” In this example, the semantic retrieval algorithm outperforms keyword-based search because the limited expressive power of the latter fails to express all the conditions in the query. Furthermore, the KB contains many instances of banks, some of which match the query, and news about these banks are recognized as relevant by the semantic

7. [http://dmoz.org/News/Online\\_Archives/CNN.com](http://dmoz.org/News/Online_Archives/CNN.com).

8. <http://www.ontotext.com/kim>.

9. <http://lucene.apache.org>.

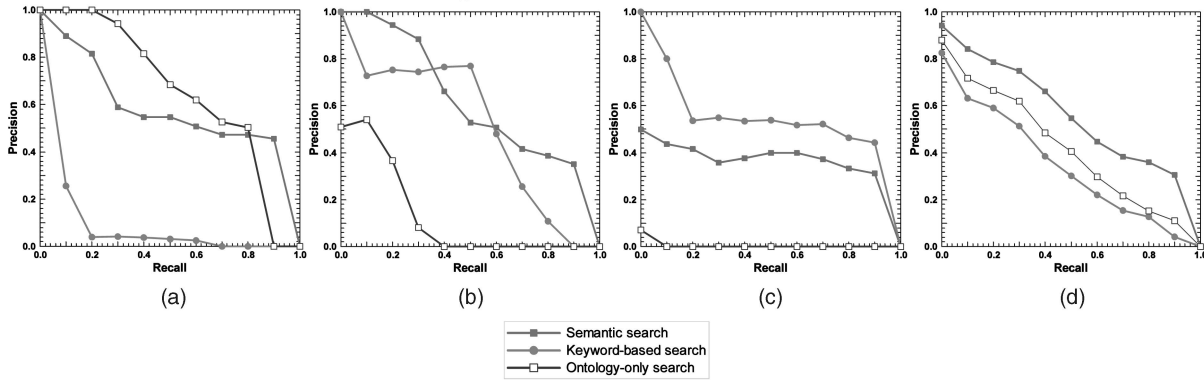


Fig. 4. Evaluation of ontology-based search (alone and combined with keyword-based) against keyword-based only. The performance of the algorithms, in terms of precision versus recall figures (as defined in, e.g., [31]), is shown for three different queries (a), (b), and (c), and averaged over 20 queries (d).

search algorithm as soon as their name is mentioned in the document, even if the text does not mention trade markets or fiscal incomes. With keyword-based search, only the documents that explicitly contain words like “bank” and “NASDAQ” are ranked highly.

These are typical results when a search query involves a region of the ontology with a high degree of completeness in terms of instances and annotations. These cases yield a high precision up to almost maximum recall. However, the KB does not contain all banks, which explains the decrease of precision at 100 percent recall. If more instances were added, precision would stand at high levels for all the recall values.

**Query b.** “News about telecom companies.” In this example, the ontology KB has only a few instances of telecom companies, so not all documents relevant to the query are annotated. This causes low precision values for the ontology-based approach, which drop to 0 for higher recall. The example shows how the combination of semantic and keyword-based results retains the efficiency of the latter when the former fails. Furthermore, in the areas where semantic search does work (here, at low recall), the combined approach takes advantage of these few good results to perform better than the keyword-based techniques.

**Query c.** “News about insurance companies in USA.” This example shows a case where our method fails. The performance of the semantic retrieval is spoiled by incorrect annotations, namely, the “Kaye” insurance company is confused with “Kaye” as a person’s name. Similarly, the “Farmers” insurance group is incorrectly assigned as an annotation to documents where the word “Farmers” refers to farm people but has no relation with insurance. It is clear that these false positives could be considerably reduced by better information extraction techniques, beyond the scope of this paper, which would discard all different meanings of these words in a text but one, once the word “insurance” is found. In exchange, this would cause misses when, e.g., the word “Kaye” appears several times in the same document with legitimately different meanings (the company versus a person), but this case is likely to be quite rare. The problem with the word “Farmers” is also related to the fact that the concept of “Farmer” as a farm person is missing in the

ontology, so it cannot even be considered as an alternative for annotation. If it was included, at least an ambiguity would be detected, and there would be a chance to solve it—even as a last resort the user might be warned. Despite these problems and aside all the possible improvements to overcome them, it can be seen that the combination with keyword-based relevance reduces the loss of precision considerably already.

The examples described in this section are representative of the typical behavior of our techniques in characteristic cases. Situations like the one illustrated by query c, where conventional search would work better, and others where the lack of knowledge in the KB results in a loss of precision, are compensated on average by the cases where the KB has a good coverage and the annotations are accurate. Fig. 4d shows an average comparison of the performance of our system over the set of 20 queries (which is comprised of queries a, b, and c analyzed earlier), and Fig. 5 shows the difference in performance (measured by R-precision) between our approach and conventional search for each of the 20 test queries. Queries (a), (b), and (c) correspond to 2, 6, and 15 in the figure, respectively. The worst performing results in queries 16 and 18 are due, again, to incorrect annotations. This suggests that further work on the automatic annotation techniques, combined with better aids for human supervision, are worthy areas for enhancing the behavior of our model. Overall, a significant improvement achieved by our approach can be observed in the global comparison provided by the histogram and the average precision curve.

## 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

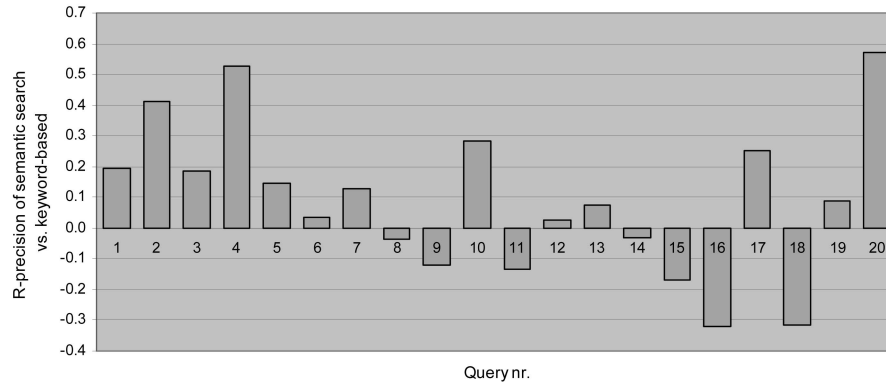


Fig. 5. Comparative precision histogram for semantic search versus keyword-based search.

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 “British companies quoted on NYSE” would return documents that mention, e.g., *Barclays PLC*, *Vodafone*, and other such companies, even if the words “British” and “NYSE” 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 clearly possible to distinguish a query for USA players in European basket teams versus European 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 European teams, regardless of whether the news mention the team at all.
- 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 the transitivity of *locatedIn*).
- Better precision by reducing polysemic ambiguities using instance labels and classifications of concepts and documents.
- Despite the separation of the content space (documents) and the concept space, it is possible to combine conditions on concepts and conditions on contents. For example, in a query like “film reviews published within the current year about Japanese sci-fi movies,” the type (film review) and date (current year) requirements are set on the document,

whereas the rest of the query defines conditions on some concept (a movie), not in the document space, that annotates the document.

- The improvements of our method with respect to keyword-based search increase with the number of clauses in (i.e., the specificity of) the formal query. This is not surprising since the higher the complexity of the information need, the more query information is lost in a keyword-based query.
- 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-based and semantic-based rankings is tricky. While the inclusion of keyword-based results ensures the robustness of our method when ontology-based results are bad, this is at the expense of a precision loss in the opposite case. The employed score combination strategy [11] achieves an effectiveness above the average of both techniques, in fact closer to that of the best performing model for each query, as was shown in the examples, but further investigation is worth it in this area.

## 7 CONCLUSION

The research presented here started as a continuation of previous work on the construction, exploitation, and maintenance of domain-specific KBs using Semantic Web technologies [3], [4]. While some basic semantic search facilities were included in these prior proposals, there was significant room for improvement because of the rather low level of semantic detail, since the search 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 latest experiments, reported here, confirm earlier observations on lower scale corpora [37].

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 semiautomatic

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 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 [10], [20], [28]. 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. Our largest-scale experiments were based on the KIM KB, the largest-sized, publicly available ontology, providing a reasonably good coverage of knowledge areas of general importance (geographical locations and organizations), suitable for inter-domain Web content [20]. One particularly realistic direction to achieve better levels of knowledge coverage by formal KBs is to consider the (semi)automatic integration of independently developed and maintained KBs [22], [27]. This should be addressed by extending our experiments to deal with multiple heterogeneous data sets, such as complex scientific data.

Besides such important future work extensions, there is ample room for further improvement and research beyond our current results. 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). This could be addressed in straightforward ways by boosting the weight of query variables involved in conditions on document attributes, according to the importance of the fields. Further complexities arise when interoperation relationships among heterogeneous structures from different sources are involved, which could be addressed by articulating semantic graph transformation bridges, e.g., as proposed in recent work in this area [27]. 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 personalized search [2], [12].

## ACKNOWLEDGMENTS

This research was supported by the European Commission (FP6-027685—MESH), and the Spanish Ministry of Science and Education (TIN2005-06885). The expressed content is the view of the authors but not necessarily the view of the MESH project as a whole.

## REFERENCES

- [1] M. Agosti, F. Crestani, G. Gradenigo, and P. Mattiello, "An Approach to Conceptual Modelling of IR Auxiliary Data," *Proc. IEEE Int'l Conf. Computer and Comm.*, 1990.
- [2] P. Castells, M. Fernández, D. Vallet, P. Mylonas, and Y. Avrithis, "Self-Tuning Personalized Information Retrieval in an Ontology-Based Framework," *Proc. First Int'l Workshop Web Semantics (SWWS '05)*, 2005.
- [3] P. Castells, B. Foncillas, R. Lara, M. Rico, and J.L. Alonso, "Semantic Web Technologies for Economic and Financial Information Management," *Proc. First European Semantic Web Symp. (ESWS '04)*, 2004.
- [4] P. Castells, F. Perdrix, E. Pulido, M. Rico, V.R. Benjamins, J. Contreras, and J. Lorés, "Neptuno: Semantic Web Technologies for a Digital Newspaper Archive," *Proc. First European Semantic Web Symp. (ESWS '04)*, 2004.
- [5] V. Christophides, G. Karvounarakis, D. Plexousakis, and S. Tourtounis, "Optimizing Taxonomic Semantic Web Queries Using Labeling Schemes," *J. Web Semantics*, vol. 1, no. 2, pp. 207-228, 2003.
- [6] J. Contreras, V.R. Benjamins, M. Blázquez, S. Losada, R. Salla, J. Sevilla, D. Navarro, J. Casillas, A. Mompó, D. Patón, O. Corcho, P. Tena, and I. Martos, "A Semantic Portal for the International Affairs Sector," *Proc. 14th Int'l Conf. Knowledge Eng. and Knowledge Management (EKAW '04)*, 2004.
- [7] M. Cristani and R. Cuel, "A Survey on Ontology Creation Methodologies," *Int'l J. Semantic Web and Information Systems*, vol. 1, no. 2, pp. 49-69, 2005.
- [8] W.B. Croft, "Combining Approaches to Information Retrieval," *Advances in Information Retrieval*, pp. 1-36, Kluwer Academic, 2000.
- [9] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by Latent Semantic Analysis," *J. Am. Soc. Information Science*, vol. 41, no. 6, pp. 391-407, 1990.
- [10] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K.S. McCurley, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zien, "A Case for Automated Large Scale Semantic Annotation," *J. Web Semantics*, vol. 1, no. 1, pp. 115-132, 2003.
- [11] M. Fernández, D. Vallet, and P. Castells, "Probabilistic Score Normalization for Rank Aggregation," *Proc. 28th European Conf. Information Retrieval (ECIR '06)*, 2006.
- [12] S. Gauch, J. Chaffee, and A. Pretschner, "Ontology-Based Personalized Search and Browsing," *Web Intelligence and Agent Systems*, vol. 1, nos. 3-4, pp. 219-234, 2003.
- [13] A. Gómez-Pérez, M. Fernández-López, and O. Corcho, *Ontological Engineering*. Springer-Verlag, 2003.
- [14] J. Gonzalo, F. Verdejo, I. Chugur, and J. Cigarrán, "Indexing with WordNet Synsets Can Improve Text Retrieval," *Proc. COLING/ACL Workshop Usage of WordNet for Natural Language Processing*, 1998.
- [15] N. Guarino, C. Masolo, and G. Vetere, "OntoSeek: Content-Based Access to the Web," *IEEE Intelligent Systems*, vol. 14, no. 3, pp. 70-80, 1990.
- [16] R.V. Guha, R. McCool, and E. Miller, "Semantic Search," *Proc. 12th Int'l World Wide Web Conf. (WWW '03)*, pp. 700-709, 2003.
- [17] S. Handschuh, S. Staab, and F. Ciravegna, "S-Cream—Semi-Automatic Creation of Metadata," *Proc. 13th Int'l Conf. Knowledge Eng. and Knowledge Management—Ontologies and the Semantic Web (EKAW '02)*, 2002.
- [18] K. Jörvelin, J. Kekäläinen, and T. Niemi, "ExpansionTool: Concept-Based Query Expansion and Construction," *Information Retrieval*, vol. 4, nos. 3-4, pp. 231-255, 2001.
- [19] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl, "RQL: A Declarative Query Language for RDF," *Proc. 11th Int'l World Wide Web Conf. (WWW '02)*, 2002.
- [20] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff, "Semantic Annotation, Indexing, and Retrieval," *J. Web Semantics*, vol. 2, no. 1, pp. 49-79, 2004.
- [21] J.H. Lee, "Analysis of Multiple Evidence Combination," *Proc. 20th ACM Int'l Conf. Research and Development in Information Retrieval (SIGIR '97)*, pp. 267-276, 1997.
- [22] P. Lehti and P. Fankhauser, "SWQL—A Query Language for Data Integration Based on OWL," *Proc. OTM Workshops*, 2005.
- [23] T.A. Letsche and M.W. Berry, "Large-Scale Information Retrieval with Latent Semantic Indexing," *Information Sciences—Applications*, vol. 100, nos. 1-4, pp. 105-137, 1997.
- [24] A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure, "Semantic portAL: The SEAL Approach," *Spinning the Semantic Web*, pp. 317-359, 2003.
- [25] R. Madala, T. Takenobu, and T. Hozumi, "The Use of WordNet in Information Retrieval. Montreal," *Proc. Conf. Use of WordNet in Natural Language Processing Systems*, pp. 31-37, 1998.
- [26] J. Mayfield and T. Finin, "Information Retrieval on the Semantic Web: Integrating Inference and Retrieval," *Proc. Workshop Semantic Web at the 26th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, 2003.
- [27] P. Mitra, M. Kersten, and G. Wiederhold, "A Graph-Oriented Model for Articulation of Ontology Interdependencies," *Proc. Conf. Extending Database Technology (EDBT '00)*, 2000.

- [28] B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, and A. Kirilov, "KIM—A Semantic Platform for Information Extraction and Retrieval," *J. Natural Language Eng.*, vol. 10, nos. 3-4, pp. 375-392, 2004.
- [29] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," W3C working draft, <http://www.w3.org/TR/rdf-sparql-query>, 2006.
- [30] C. Rocha, D. Schwabe, and M.P. de Aragão, "A Hybrid Approach for Searching in the Semantic Web," *Proc. 13th Int'l World Wide Web Conf. (WWW '04)*, pp. 374-383, 2004.
- [31] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
- [32] A. Seaborne, "RDQL—A Query Language for RDF," W3C member submission, <http://www.w3.org/Submission/RDQL>, 2004.
- [33] A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, and Y. Warke, "Managing Semantic Content for the Web," *IEEE Internet Computing*, vol. 6, no. 4, pp. 80-87, 2002.
- [34] *Handbook on Ontologies*. S. Staab, and R. Studer, eds. Springer Verlag, 2004.
- [35] N. Stojanovic, "On Analysing Query Ambiguity for Query Refinement: The Librarian Agent Approach," *Proc. 22nd Int'l Conf. Conceptual Modeling*, 2003.
- [36] N. Stojanovic, R. Studer, and L. Stojanovic, "An Approach for the Ranking of Query Results in the Semantic Web," *Proc. Second Int'l Semantic Web Conf.*, 2003.
- [37] D. Vallet, M. Fernández, and P. Castells, "An Ontology-Based Information Retrieval Model," *Proc. Second European Semantic Web Conf. (ESWC '05)*, 2005.



**Pablo Castells** received the PhD degree in computer science in 1994 at Universidad Autónoma de Madrid (UAM), with a thesis on Automated Theorem Proving. He has been an associate professor at UAM since 1999. In 1994/1995, he was a postdoctoral research fellow at the University of Southern California. More recently, he has led or participated in several national and international projects in the areas of the semantic Web and knowledge-based systems, in application domains such as News, Finance, and Healthcare. His current research focuses on information retrieval, semantic-based technologies, and personalization. He leads the Networked Semantics Team at UAM.



**Miriam Fernández** received the MS degree in computer science from the Universidad Autónoma de Madrid (UAM), where she works as a research assistant. Her research interests include ontology engineering, semantic search, and semantic annotation.



**David Vallet** received the MS degree in computer science from the Universidad Autónoma de Madrid (UAM), where he is currently a research assistant. His research interests are focused on the confluence of information retrieval, user modeling, and context modeling.

▷ For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).