Searching the Semantic Web: Approximate Query Processing based on Ontologies

Olivier Corby, Rose Dieng-Kuntz, Catherine Faron Zucker, Fabien Gandon

Searching the Semantic Web: Approximate Query Processing based on Ontologies

Olivier Corby[1], Rose Dieng-Kuntz[1], Catherine Faron-Zucker[2], and Fabien Gandon[1]

[1] INRIA (National Research Institute in Computer Science and Control), Acacia Project, Sophia Antipolis, France.
E-mail: {Olivier.Corby, Rose.Dieng, Fabien.Gandon}@sophia.inria.fr}

[2] I3S (Institute of Research in Computer Science of the University of Nice Sophia Antipolis), M@inline Team, France, E-mail: Faron@essi.fr

Abstract

The semantic web relies on ontologies representing domains through their main concepts and the relations between them. This domain knowledge is the keystone to describe the contents of web resources and services. These metadata then enable us to search for information based on the semantics of web resources rather than their syntactic forms. However, in the context of the semantic web there are many possibilities of executing queries that would not retrieve any resource. The viewpoints of the designers of ontologies, the designers of annotations and the users performing a Web search may not completely match. The user may not completely share or understand the viewpoints of the designers and this mismatch may lead to missed answers. Approximate query processing is then of prime importance for efficiently searching the Semantic Web. In this paper we present the Corese ontology-based search engine we have developed to handle RDF(S) and OWL Lite metadata. We present its theoretical foundation, its query language, and we stress its ability to process approximate queries.

Keywords

Semantic Web, Web Search, Approximate Queries, RDF(S), Conceptual Graphs

I. Introduction

The present Web comprises a huge amount of heterogeneous data dedicated to human users. The Semantic Web aims at representing the contents of Web resources in formalisms understandable by automated tools as well as by humans. It relies on rich metadata, also called semantic annotations, offering explicit semantic descriptions of Web resources. These semantic annotations are built on ontologies, representing domains through their concepts and the semantic relations between them. Ontologies are the foundations of the so called Semantic Web and the keystone of the automation of tasks on the web: searching, merging, sharing, maintaining, customizing, monitoring, etc.

Here, we focus on searching as needed in web applications such as Digital Libraries, Web Intelligence, Corporate Webs for Knowledge Management, etc. Publishing languages like HTML enable us to retrieve documents based on their presentation and their textual contents; structuring languages like XML or SGML enable us to access web resources based on their data structure. Semantic annotations improve the Web search and enable us to access web resources based on their semantic descriptions. OWL and RDFS are the two semantic web languages recommended by the W3C and both are built upon RDF.

In this paper, we address the problem of a dedicated ontology-based query language. We show how ontologies ensure an efficient retrieval of Web resources by enabling inferences based on domain knowledge and we emphasize the prime interest of semantic approximations for efficiently searching the Semantic Web. The vision of the Semantic Web implicitly relies on the (strong) hypothesis that an ontology designed to describe a domain is usable both to annotate web resources and to retrieve them. Reality is more contrasted. Usually, an ontology is built by specialists of the domain, not by specialists of the Web search task in this domain, *i.e.* the users. The user may not completely share or understand the viewpoints of the ontology designers. There may be some mismatch between the need of a clean reusable formal ontology and an effective guideline for Semantic Web search.

Users may not use the *right* concepts - from the viewpoint of the ontologist - when writing a query, and this mismatch may lead to missed answers. Some experiments of the Corese semantic search engine we have developed give us good examples of misunderstanding or misuse by the user of concepts stated by the ontologist: in the CoMMA project the *Commerce* concept has been used instead of the *Business* one, *TechnicalReport* instead of *ResearchReport*. Moreover, a user asking for a *person* working on a *subject* may appreciate, instead of a failure, the retrieval of a *research group* working on that subject, even if a research group is not exactly a person. Lastly, a user may search for some related resources without knowing how their possibly complex relation is stated in the annotations. For instance, a user may search for organizations related to human sciences while ignoring the diversity of relations used to express this relationship in the annotations. All these examples illustrate the prime interest of semantic approximations for efficiently searching the Semantic Web.

In the next section, we present a concrete scenario of semantic search with Corese to highlight its strong point: approximate ontology-based search. In section 3 we present the Corese Semantic Web search engine we have developed, its theoretical foundations, its ontology representation language and its query language dedicated to the retrieval of web resources annotated in RDF(S). In section 4 we focus on the approximate query processing provided by Corese; we show how the Corese query language enables both ontological and structural approximations. Finally, we present the software architecture and some concrete experiments which demonstrated its interest in several real world projects.

II. A Concrete Motivational Scenario

Let us consider the KMP[1] recent project for which we have built upon Corese a knowledge management platform for cartography of skills in telecommunications for Sophia Antipolis firms, members of the Telecom Valley association.

The goal of KMP is to build an innovative solution of knowledge management shared within a community, in order to foster synergies and partnerships by providing a dynamic map of the competences of the different stakeholders. The solution relies on the specification, design, building and evaluation of an on-line customizable service. This service is becoming the main component of a portal for the community of the industries, the academic institutes and the institutional organizations involved in the Telecom Valley of Sophia Antipolis. The project is a real-world experiment and the steering committee is composed of eleven pilot companies including: Amadeus, Philips Semiconductors, France Telecom R&D, Hewlett Packard, IBM, Atos Origin, Transiciel, Elan IT, Qwam System, Cross Systems.

[1] http://www.telecom.gouv.fr/rnrt/projets/res_02_88.htm

The KMP platform provides clustering views to analyze the competences present in the Telecom Valley. The screenshot in figure 1 shows one of these views called the "Clusters". It presents a distribution of grapes corresponding to resources involved in the competence and each grape contains bubbles representing actions (e.g. "*produce*", "*design*") involved in the competence. The SVG view is dynamically generated from the integrated RDF/S data collected. It provides a very powerful representation to analyze the diversity of the Telecom Valley. The grouping of competences relies on the ontology-based distance defined in Corese to evaluate the conceptual similarities between the competences.

Remaining in the domain of skill management, the following scenario also demonstrates a strong asset of Corese. Let us consider a query asking for persons both expert in Java programming and interested in XML. As an application, this query is used in suggesting profiles to build project teams or manage mobility in a company. A basic exact retrieval in the annotation base would produce no answer to this query. However, when submitted to Corese, eight answers are retrieved. One exact answer is the result of the application of a domain rule of the ontology stating that a person author of a Thesis on a given subject is an expert of this subject: Yvonne Duchard is author of a thesis on Java programming, so she is expert on it.

Moreover this answer is extended with an interesting approximation: in addition to XML, Yvonne Duchard is also interested in (aware of) Wap which is close enough to XML. This shows how Corese supports serendipity. The seven other answers approximately match the query. One answer is an engineer skilled in both XML and Java programming; another answer is a project manager skilled in both XML and EJB programming. These two annotations have the same similarity to the query: in both cases, the *IsExpertIn* property is approximated by *IsSkilledIn* which is close enough to it.

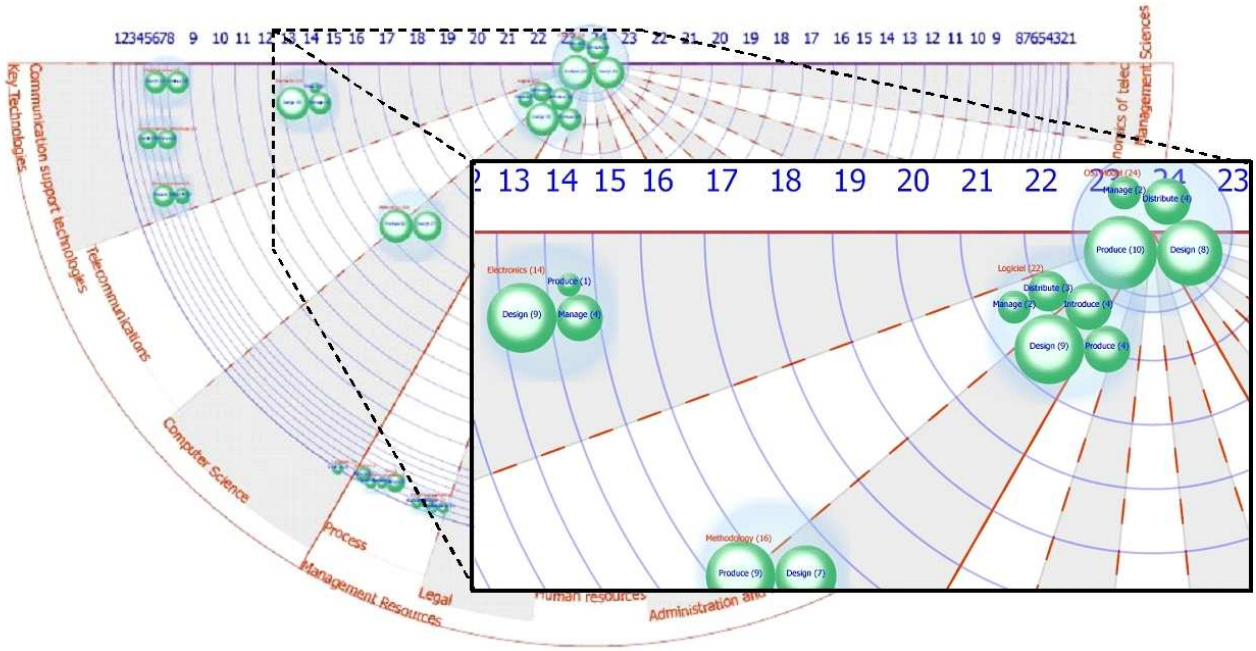

Figure 1: Conceptual clustering of the competences of the Sophia Telecom Valley

III. Ontology-based Web Search

A. A Logic-based Approach

Ontologies enable us to take into account during the query processing some background knowledge implicit in the annotations. This comprises subsumption links between concept types or relation

types, signatures of relations, axioms or rules enabling deductions, etc. This knowledge supports inferences improving the efficiency of the matching process.

The use of ontological knowledge in web search approaches is expressed in the following logical model. Given (1) a model for ontologies, (2) a model for annotations of web resources based on ontologies, (3) a model for queries based on ontologies, and (4) a matching function defining how a query is matched with any annotation, a web resource R is relevant for a query Q according to the ontology O from which they are built *iff* the annotation of R and the ontology O together logically imply Q (noted $R \wedge O \rightarrow Q$).

The query is viewed as a set of constraints on the description of the web resources to be retrieved and then corresponds to a search problem to be solved. The matching function implements the strategy chosen for solving this problem. It differs from one search system to another, depending on the formalism chosen for the descriptions, the types of query and the characteristics to be met by the result. Corese [2] implements such a matching function using the projection operator defined in the Conceptual Graphs (CG) formalism [2].

B. Theoretical Foundations of Corese

The Corese semantic search engine internally works on CG: when matching a query with an annotation according to a shared ontology, the query, the annotation and the ontology are translated into the CG model. CG and RDF(S) models share many common features and we established a mapping between RDF(S) and a large subset of the CG model. An in-depth comparison of both models was the starting point of Corese.

Both models distinguish between ontological and assertional knowledge. In both models, assertional knowledge is positive, conjunctive, and existential and it is represented by directed labeled bipartite graphs. In Corese, an RDF graph G representing an annotation or a query is thus translated into a CG. Regarding the ontological knowledge, the class (resp. relation) hierarchy in a RDF Schema corresponds to the concept (resp. relation) type hierarchy in a CG support. RDF properties are declared as first class entities like RDFS classes, in just the same way that relation types are declared independently of concept types in a CG support. This common handling of properties makes the mapping very relevant as opposed to object-oriented language, where properties are defined inside classes.

There are some differences between RDFS and CG models in their handling of classes and properties but they are easily handled. The declaration of a resource as an instance of several classes in RDF can be translated in the CG model by generating the concept type corresponding to the most general specialization of the concept types translating these classes. Similarly, the multiple domain (resp. range) constraints of an RDF property can be translated into a single domain (resp. range) constraint in CG by generating the concept type corresponding to the most general specialization of the concept types constraining the domain (resp. range).

As a result, the searching of RDF(S) through CG consists of compiling the type hierarchies of the CG support, associating a compiled type to each resource and, finally, using the projection operation of the CG model for an optimized query processing based on compiled type hierarchies.

This projection operation is the basis of reasoning in the CG model. A CG G_1 logically implies a CG G_2 *iff* it is a specialization of G_2 (noted $G_1 \leq G_2$). A CG G_1 is a specialization of G_2 *iff* there

[2] http://www.jfsowa.com/cg/cgstand.htm

exists a projection of G_2 into G_1 such that each concept or relation node of G_2 is projected on a node of G_1 whose type is the same as the type of the corresponding node of G_2 or a specialization of it, according to the concept types and relation types hierarchies.

Corese retrieves the resources for which there exists a projection of the query graph into their annotation graphs. For example the following query graph enables us to search for resources about science and their authors.

```
[Document:*]-
    -(createdBy)-[Person:*]
    -(subject)-[Science:*]
```

When processing this query, Corese retrieves a book of a professor about social science annotated with the following graph, upon which there exists a projection of the query graph.

```
[Book:#book9638]-
    -(createdBy)-[Professor:#david-dupond]
    -(topic)-[SocialScience:*]
```

The node `[Document:*]` is projected upon `[Book:#book9638]}, Book` being a subclass of `Document` in the ontology and the URI `#book9638` specializing the generic referent `*` ; likewise for the nodes `Person` and `Professor`, and the nodes `Science` and `SocialScience`. The node (`createdBy`) is projected upon its counterpart and (`subject`) is projected upon (`topic`), a subproperty of `subject` in the ontology.

C. Corese Ontology Representation Language

The first ontology representation language of Corese was RDFS. It has progressively been extended to handle some major features of OWL Lite. Our choice of RDFS is mainly historical: the first implementations of Corese with RDF(S) preceded the emergence of OWL. However the different projects in which Corese has been experimented have shown us that the expressivity of RDF(S) is sufficient in many applications - if extended with inference rules and approximation in the query language. We think that OWL Lite features are quite sufficient to handle most knowledge representation problems encountered in Semantic Web applications. Corese provides OWL value restrictions, intersection, subclass and algebraic properties such as transitivity, symmetry and inverse. It also provides the annotation, versioning and ontology OWL statements. Corese does not yet provide cardinality restrictions, property and class equivalences, `owl:sameAs` and loops in subsumption hierarchy.

These extensions to OWL features are based on domain axioms, Corese integrates an inference engine based on forward chaining production rules. The rules are applied once the annotations are loaded and before the query processing occurs: the annotation graphs are enriched before the query graph is projected. This is the key to the scalability of Corese to the web application in which we have used it. Corese implements CG rules. For instance, the following CG rule states that if a person *?m* is head of a team *?t* which has a person *?p* as a member, then *?m* manages *?p* :

```
[Person:?m]-(head)-[Team:?t]-(hasMember)-[Person:?p]
=> [Person:?m]-(manage)-[Person:?p]
```

A rule $G_1 \Rightarrow G_2$ applies to a graph G if there exists a projection π from G_1 to G_2. The resulting graph is built by joining G and G_2 while merging each $\pi(x_i)$ in G with the corresponding x_i in G_2.

Joining the graphs may lead to specializing the types of some concepts, to create relations between concepts and to create new individual concepts.

The Corese rule language is based on the triple model of RDF. For instance, the CG rule above is the translation of the following Corese rule:

```
<cos:rule>
 <cos:if>
    ?m rdf:type s:Person
    ?m s:head ?t
    ?t rdf:type s:Team
    ?t s:hasMember ?p
    ?p rdf:type s:Person
 </cos:if>
<cos:then>
    ?m s:manage ?p
</cos:then>
</cos:rule>
```

D. Corese RDF Query Language

A query is either a triple or a Boolean combination of triples. For instance the following query retrieves all the persons (line 1) with their names (line 2) who are authors (line 3) of a thesis (line 4) and returns its title (line 5):

```
 (1) ?p rdf:type kmp:Person
 (2) ?p kmp:name ?n
 (3) ?p kmp:author ?doc
 (4) ?doc rdf:type kmp:Thesis
 (5) ?doc kmp:Title ?t
```

The first element of a Corese triple is a variable or a resource qualified name (XML qname); the second is either a property qname, a variable or a comparison operator; the third is a variable, a value or a resource qname. Class and property names are qnames whose namespaces are either standard and denoted by predefined prefixes (`rdf, rdfs, xsd, owl` and `cos` for the Corese namespace) or user-defined prefixes e.g.:

```
dc as http://purl.org/dc/elements/1.1/
```

Variable names begin with a question mark. Values are typed with the XML Schema Datatypes (XSD): numerical, `xsd:string`, `xsd:boolean` and `xsd:date`. The language of the value of a literal can be specified using `@` operator and the constants defined for `xml:lang`. For instance, we can constrain the title of the thesis to be in English:

```
?doc kmp:Title ?t@en
```

The comparison operators for equality and difference `(=, !=)`, ordering (`<, <=, >, >=`) and string inclusion and exclusion (~, !~) enable us to compare a variable with a constant or another variable. For instance we can state the title must include the term 'web':

```
?t   ~ "web"
```

Type comparators (<:, <=:, =:, >=:, >:) and combinations with the ! (negation operator) enable us to specify constraints on some types in a query. For instance, we can constrain the document to be a strict specialization of a thesis (e.g. a PhD thesis, a MSc thesis):

```
?doc <: kmp:Thesis
```

By default, a list of triples is a conjunction. The `or` and `and` operators with parenthesis enable us to combine conjunctions and disjunctions in a query. Corese handles such queries by transforming them into disjunctive normal form, processing each conjunctive subquery and juxtaposing all the results.

Negation as failure is provided by a `not` operator to prefix properties that should not be found in an annotation. For instance, we can limit our search to not graded documents:

```
?doc not::kmp:grade ?g
```

The Corese query language supports queries on ontologies just like on annotations since RDF Schemas are RDF graphs. For instance, the following query retrieves the properties whose `domain` is a subclass of `kmp:Document`.

```
?p rdf:type rdf:Property
?p rdfs:domain ?c
?c rdfs:subClassOf kmp:Document
```

Some SQL-like operators customize the presentation of the retrieved answers:

- By default, all the values of the variables are returned. A `select` operator allows us to list the values desired in the answers. For instance, we can choose to only return the title of the documents and the name of their author:
  ```
  select ?t ?n
  ```

- A `group` operator allows us to group the retrieved answers according to one or more concepts instead of listing separately answers about the same concept(s). For instance, when looking for documents on a specific subject and written by an author, a `group` on the `?doc` variable will avoid that a document written by several authors appears for each of its authors.

- A `count` operator, combined with `group` allows the counting of the (different) answers retrieved. For instance, to count the number of documents written by a person, `count` is applied to the variable `?doc` and `group` to the variable `?p`. Finally, the SPARQL syntax for the query language is under development.

IV Approximate Semantic Web search

We have extended the core query language of Corese to address the problem of possible mismatch between end-user and ontologist concepts. Corese is able to cope with queries for which there is no exact answer by approximating the semantics of the query, its structure, or both.

A. Ontological Approximation

The first principle of the Corese semantic approximation is to evaluate semantic distances between ontological types. Based on this ontological distance, Corese not only retrieves web resources

whose annotations are specializations *of* the query, it can also retrieve those whose annotations are *semantically close*.

1) Ontological Distance

To evaluate conceptual relatedness, Corese relies on the structure of the ontology. In CG, structured-based distances are the key to define a non binary projection, *i.e.* a similarity $S: C^2 \rightarrow [0,1]$ where 1 is the perfect match and 0 the absolute mismatch. Corese uses such a similarity to carry out approximate search.

Starting from the fact that in an ontology, low level classes are semantically closer than top level classes (for instance *TechnicalReport* and *ResearchReport* which are brothers at depth 10 are closer than *Event* and *Entity* which are brothers at depth 1), we want the ontological distance between types to decrease with depth.
To capture this, let the length of a subsumption link (t, t') between a type t and one of its direct super types t' in an inheritance hierarchy H be $1/2^{d_H(t')}$, where $d_H(t')$ is the depth of t' in H. Because of multiple inheritance, d_H refers to the maximal depth (with $d_H(T)=0$, $\forall x \in H$, $d_H(x) \leq d_H(\perp)$, and $\forall (x,y) \in H^2$, $y < x \rightarrow d_H(y) < d_H(x)$; T and $\perp$ being the root and the bottom of the hierarchy).

Then we define the length of a subsumption path between a type t_1 and one of its super types t_2 in an inheritance hierarchy H as the sum of the lengths of the subsumption links making up this subsumption path:

$\forall(t_1, t_2) \in H^2$, $l_H(< t_1, t_2 >) = \sum_{\{t \in < t1, t2 >, t \neq t1\}} 1/2^{d_H(t)}$.

Finally, we define the ontological distance between two types as the minimum of the sum of the lengths of the subsumption paths between each of them and a common super type:
$D_H(t_1, t_2) = \min_{\{t \geq t1, t \geq t2\}} (l_H(< t_1, t >) + l_H(< t_2, t >))$.
We proved that D_H is a semi-distance [3].

2) Contextual Closeness

The ontological distance between two classes is not always sufficient to render the closeness of some concepts. We encountered cases where concepts are distant in the ontology but share some features that make them close from the search point of view. For instance, in the O'CoMMA ontology, *KnowledgeDissemination* which is in the Activity viewpoint and *KnowledgeEngineering* which is in the Topic viewpoint share some semantics that is not expressed by the `rdfs:subClassOf` link. When querying for *KnowledgeDissemination*, one may want to retrieve *KnowledgeEngineering* resources in case of failure. Similarly, some properties may share a semantic proximity such that it makes it desirable to authorize the occurrence of one of them instead of the other when matching a query with an annotation.

Hence, Corese provides the ability to express relatedness by means of the standard `rdfs:seeAlso` property. This property can be added to any existing RDF Schema, so that a given ontology can be parameterized to better fit a specific Web search task or a particular user class. This addition does not only improve browsing capabilities, it also shortens the semantic distance and tunes approximate matching. It is worth considering the `rdfs:seeAlso` property be inherited by subclasses and subproperties. Hence any Corese ontology has the following rule for classes (and the equivalent one for properties):

```
?x rdfs:seeAlso ?y
?z rdfs:subClassOf ?x
=> ?z rdfs:seeAlso ?y
```

3) Approximate Projection

Based on the ontological distance defined above, Corese distinguishes between *exact answers* for which there exists a projection of the query upon their annotations and *approximate answers* for which there exists an *approximate projection* of the query upon their annotations. These annotations have a structure upon which the query can be projected but their concept and relation types are not necessarily subsumed by those of the query: they are just close enough to them in the ontology.

Formally, we define an approximate projection from a CG $G = (C_G, R_G, E_G, l_G)$ to a CG $H=(C_H, R_H, E_H, l_H)$ as a mapping Π from C_G to C_H and from R_G to R_H which (1) preserves adjacency and order on edges, (2) may change the labels of concept nodes to ontologically close ones (the ontological distance between a concept type in *G* and its projection in *H* must be lower than a given threshold), (3) may decrease the labels of relation nodes or change them to contextually close ones (for which a *seeAlso* property stands) [3].

Corese authorizes the approximation of a class by potentially any other class of the ontology whereas for combinatorial constraints, the approximation of a property is limited to contextual closeness. Ontological distances are thus computed between concept types (and not between relation types) and the similarity between a resource annotation and a query depends on the ontological distances between the types of their concept nodes, contextual closeness being translated in terms of ontological distances.

Setting a `rdfs:seeAlso` property between two concept types c_1 and c_2 has for effect to shorten the ontological distance between them to a brotherhood distance and consequently increase the similarity between two graphs for which there exists an approximate projection mapping a node of type c_1 in one graph to a node of type c_2 in the other graph.

Setting a `rdfs:seeAlso` property between two relation types r_1 and r_2 is also taken into account in the computation of the similarity between two graphs for which there exists an approximate projection mapping a node of type r_1 in the query graph with a node of type r_2 in the target graph. The cost of this approximation is proportional to $1/2^d$ where $d = \max(d_H(c_1), d_H(c'_1))$, i.e. the maximum depth of the types c_1 and c'_1 of the neighbour concept nodes of r_1.

The relative relevance of the retrieved annotations is measured by their similarity to the query. Those whose similarity does not overpass a given threshold are presented to the user, sorted by decreasing similarity. This threshold is relative to the best found approximation of the query.

Syntactically, the `more` keyword in the `select` clause of a Corese query asks for approximate answers. In this case, Corese basically approximates every concept of the query. However, its query language allows to require the specialization of some concepts while approximating the others by using type comparators. For instance, by using the `<=:` operator, Corese is able to retrieve the persons interested in *Knowledge Engineering* (or something close) and member of a project (or something close) :

```
select more where
 ?person c:interestedIn ?k
 ?person <=: c:Person
 ?k rdf:type c:KnowledgeEngineering
 ?person c:member ?project
```

```
?project rdf:type c:Project
```

In this query, the class *Person* or one of its subclasses is required (by `<=:`), while *KnowledgeEngineering* and *Project* may be approximated.

B. Structural Approximation

The ontology-based approximation described above makes up for the possible divergences between the *vocabularies*. Another kind of approximation supported by the Corese query language makes up for the possible divergences between the annotation structures and the query structure. In some cases, the user will search for conceptually related resources while ignoring how to express their relationship, *i.e.* how the annotator has described it. For instance, the user may search for organizations related to *Human Science*, whatever the relationship is. This kind of approximation concerns the *structure* of the annotations but still remains semantic. It can be viewed as the approximation of a complex relationship that cannot be represented by a single property and requires a graph to define it.

The Corese query language supports such approximations through the *path graph* feature. It allows to search for resources related by a relation path graph (made of successive binary relations between a series of intermediate concepts).

We have extended our definition of an approximate projection of a CG *G* to a CG *H* in order to allow the mapping of a relation node with a path graph : adjacency and order on edges are preserved, considering the graph *H'* where the path graphs of *H* upon which relation nodes of *G* are projected are contracted to relation nodes whose types are defined by these path graphs [3].

Syntactically, in the Corese query language, the relation for which a complex relationship is searched for must be suffixed by the maximal length of the path graphs to search for. By default, Corese stops after retrieving one path (with the shortest length). It computes all the possible paths when the relation is prefixed by the `all` qualifier.

For instance, let us consider the following query asking for the organizations related to *Human Science* by a (non directed) relation path of length smaller or equal to two:

```
?org all::c:relation{2} ?topic
?org rdf:type c:Organization
?topic rdf:type c:HumanScience
```

The two following annotations answer this query: the CNRS institute is interested in *Human Science* and a member of the INRIA institute is graduated in *Human Science*.

```
[Institute:#CNRS]
   - (interestedIn)-[HumanScience:*]

[Person:#Alain]
   -(memberOf)-[Institute:#INRIA]
   -(graduatedIn)-[HumanScience:*]
```

V. Software, Applications and Evaluation

A. Architecture

Corese is developed in Java and publicly available under the INRIA licence at `http://www.inria.fr/acacia/corese` including Java packages, documentation and GUI. A Corese semantic web server has also been developed according to a 3-tier architecture (figure 2) as described in the following subsections.

1) Presentation Layer

It generates the content to be presented in the user's browser (ontology views and browsing controls, query edition interfaces, annotation forms, answers, etc.), this part relies on a model-view-controller architecture to handle HTTP requests and generate responses fed by the appropriate Corese services of the Business Logic Layer and formatted using XSLT or JSP templates. It is implemented by servlets and provides the front-end of what we call a Semantic Web Server, i.e. an HTTP server able to: solve semantic web queries submitted through HTTP requests; provide JSP tags to include semantic web processing and render results in web pages; provide XSLT extensions to perform semantic web functions related to XPath expressions, thus improving RDF/XML transformation capabilities; provide a form description language to dynamically build forms using queries for instance to populate the different choices of a drop-down box.

2) Business Layer

It consists of a platform that implements three main services accessible through an API: a Conceptual Graph server (using the Notio API [3]), a query engine and a rule engine. Parsers transform RDF to CG, Rules to CG Rules and Queries to CG graphs to be projected. The core CG server implements the management of the CG base, the projection and join operators and type inferences on the type hierarchies. A CG-to-RDF pretty-printer produces results in RDF/XML syntax. This layer is an independent package and provides an API that can be used by developers to add semantic web capabilities to their applications.

3) Persistent Layer

RDF(S) data are accessed by means of the ARP[4] parser and translated by the RDF-to-CG Parser. Rules are saved in separate files and parsed by the Rule Parser.

[3] http://www.cs.ualberta.ca/~finnegan/notio/
[4] http://www.hpl.hp.com/semweb/arp.htm

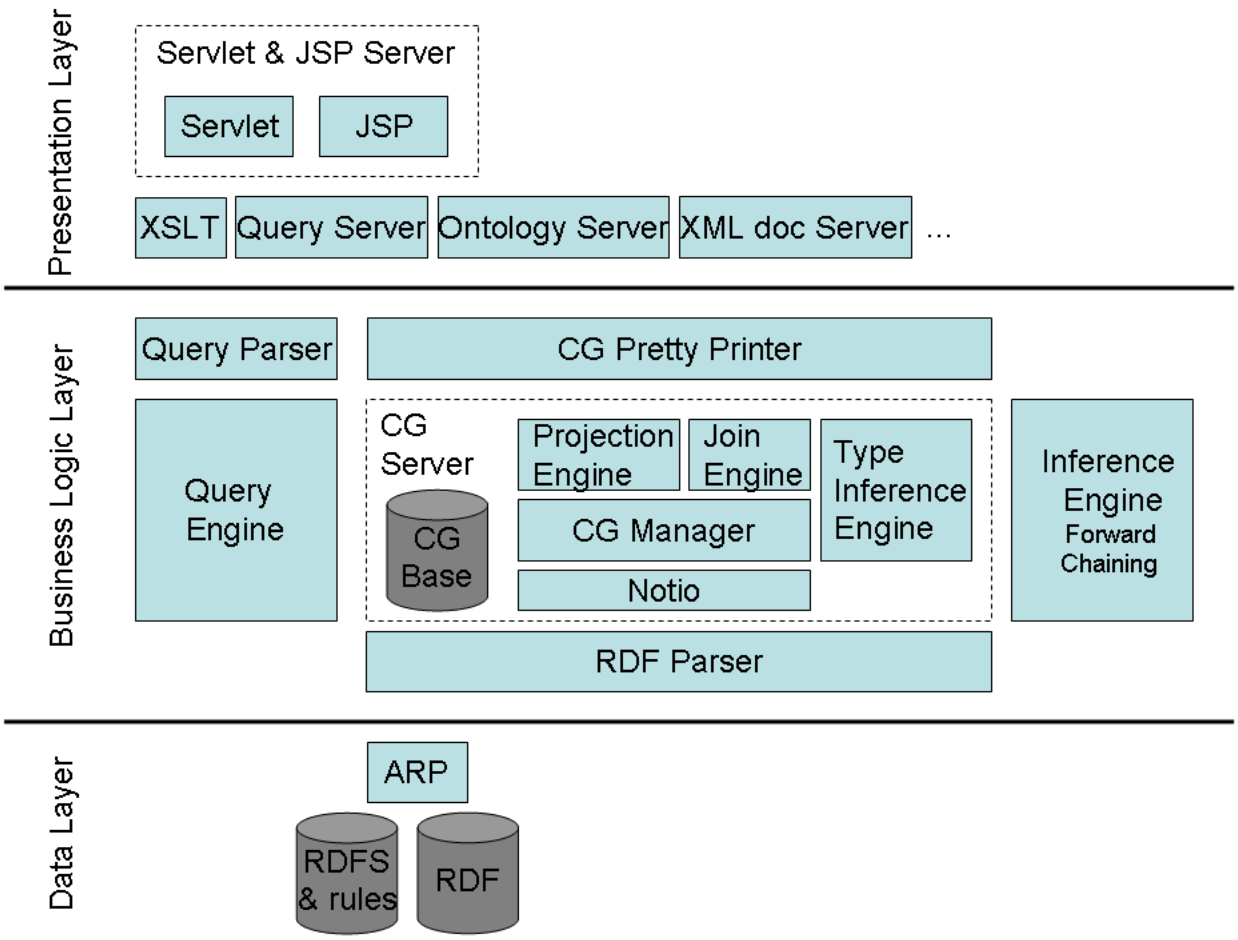

Figure 2 : Corese 3-tier architecture

B. Real World Applications

Corese has been tested on several real-world, large-scaled applications with ontologies: these applications are detailed in [5].

1) SAMOVAR: a system supporting a vehicle project memory for Renault car manufacturer. The ontology has 792 concept types and 4 relation types, and annotates 4483 problem descriptions. Corese answers queries such as: *"Find all fixing problems that occurred on the dashboard in a past project"*.

2) CoMMA IST project: a multi-agent system for corporate memory management (integration of a new employee and technological watch). The O'CoMMA ontology comprises 472 concepts types and 80 relation types used for annotating documents or people in an organization. Corese answers distributed queries over several annotation bases such as *"Find the users who may be interested in the technological news that was just submitted about GSM v3"*.

3) ESCRIRE: the annotation and search of abstracts of Medline database on genetics. Corese answers queries such as: *"Find the articles describing interactions where the Ubx gene acts as target and where the instigator is either* en *or* dpp *gene"*.

4) KMP: a knowledge management platform for cartography of skills in telecommunications for Sophia Antipolis firms. The KMP ontology comprises 1136 concept types with a maximal depth of 15 and 83 relation types. Corese answers queries such as: *"Who are the possible industrial partners knowing how to design integrated circuits within the GSM field for cellular/mobile phone manufacturers?"*.

5) Ligne de Vie: a virtual staff for a health network relying on an ontology comprising 26432 concept types and 13 relation types. It guides physicians discussing the possible diagnoses and the alternative therapies for a given pathology, according to the patient's features. It enables to answer queries such as: *"Find the past sessions of virtual staff where a given therapy was chosen for the patient and indicate what were the arguments in favour of this therapy"*.

6) MEAT: a memory of experiments of biologists on DNA microarray relying on annotations on scientific articles, and using UMLS as an ontology. Corese answers queries such as *"Find all the articles asserting that HGF gene plays a role in lung disease".*

Corese has also been tested with other ontologies such as the Gene ontology (represented by an RDF graph with 13700 concept types and 950000 relations), IEEE LOM, W3C CC/PP, Dublin Core, etc.

C. Evaluation

We will illustrate the evaluation from system viewpoint (performance) and from end-user viewpoint (scenario-based evaluation).

1) Corese Performance

Corese engine performance has been measured on an RDF(S) base which comprises 19000 properties, 8000 resources and 10 rules. The Corese standard test base of 262 queries covering all the features of the query language runs in 9.7 seconds on a laptop. The average answer time is 0.037 second per query. The efficient projection operator enables Corese to achieve good performances and then to be usable in real world applications.

2) Scenario-based evaluation

Once provided with domain axioms, approximate queries and presentation capabilities (features that were really required by the users), Corese received a very positive evaluation by its users and [6] details the scenario-based evaluation used for several applications among which CoMMA and KMP.

The evaluation conducted on the KMP application of Corese involved 10 mediators and about 30 users from 17 organizations. While users appreciate the technical features of Corese, they criticize some useability aspects of the application.

The following positive points were emphasized:

- the users found the Corese query language power effective,
- the users appreciated the ontology-driven user interface forms,

- the users appreciated the approximate search feature of Corese and considered it as unique and very useful since it enabled them to find the best match for any query with the ontology.

Several useful improvements on the KMP system were suggested by the users:

- Some users would like more dynamic interactions in the query answer cycle. They would like to be able to easily refine a query from the answer. When a query is refined, they would like the differences in the answer to be enhanced. They would also appreciate the system to manage a history of queries.
- Some users estimated that the ordering of approximate answers could be improved. They would also like the system to justify the proposed approximations. Some users would also like the ability to tune the approximation: e.g. which concept can be approximated and how. In the result, it should be possible to document the distance of each approximate concept to its query concept. Specific style sheets are necessary for approximate results, which is already the case in the KMP application.
- Some experiments showed that the generic distance was not always completely accurate: sometimes a class is closer to its brother class than to its direct ancestor. This incites us to some more work on distance modelling in ontologies.

As a conclusion, ontology-driven tools are powerful and useful. However, the *interaction* with users should not be directly driven by the ontology but by user, task and domain models.

VI. Related Work

A. Query Languages for RDF

The Corese RDF Query Language is close to RDQL, SeRQL[5] (Sesame language [1]) and SPARQL[6] which may become a W3C recommendation to query RDF.
Like SPARQL, Corese Query Language is based on a Boolean combination of triples that can be constrained by computable expressions. Corese also processes datatyped RDF literals, optional patterns and alternatives. Corese returns an RDF/XML graph or an XML binding format. The bindings are available through an API. Corese also provides the `select, distinct, order` and an equivalent of limit statements but not the `construct, describe` and `ask` SPARQL statements. The two last ones can be simulated by Corese.

In addition to SPARQL statements, Corese provides approximate search and structural path graph. It enables to group and count results. It enables to merge all results into one graph or provide the results as a list of graphs. It can also, at user option, generate the result using the vocabulary (the classes) used in the query instead of the possibly specialized vocabulary of the target RDF graph.

B. Ontology-Based Web Search Applications

Among previous ontology-based Web search applications, let us cite OntoBroker [4] in which ontologies and queries are expressed in Frame Logic and translated into Horn Logic. Sesame [1] and RDQL [10] rely on Database Management Systems (DBMS) to store and query RDF(S).

[5] http://www.openrdf.org/doc/users/ch06.html
[6] http://www.w3.org/TR/rdf-sparql-query/

Beside these general-purpose reasoners, WebKB [9] and OntoSeek [7] are search-oriented applications, based on CG. WebKB interprets statements expressed in a CG linear notation and embedded in HTML documents; it allows to query lexical or structural properties of HTML documents. OntoSeek focuses on lexical and semantic constraints in the encoding of resources into CG and the building of queries.
WebKB, OntoSeek and Corese are all built upon CG and consequently use the same core principle of matching a query graph against annotation graphs with respect to subsumption relations between concepts or relations. However neither WebKB nor OntoSeek handle RDF(S) data like Corese does, and they do not handle rules in their ontology representation language. Moreover they both focus on the annotation activity and ontological problematics and they have no expressive query language like Corese.

Above all, when compared to these applications, Corese is the only ontology-based system to provide approximate search features. To the best of our knowledge, it is the only web search application addressing the problem of structural approximation of queries. There are some few recent works addressing the problem of ontological approximation for searching the web. Among them, let us cite [8] that approximates overlap between RDFS concepts based on Bayesian networks and proposes to apply their approximation to define a semantic distance between concepts and sort the answers to an ontology-based search. But this method has not actually been applied to web search and it focuses on overlap rather than subsumption. The PASS system [12] searches abstracts of research papers; the search uses a fuzzy ontology of term associations for query refinement. When compared to Corese, PASS searches for documents tagged with domain-specific keywords while Corese searches for documents annotated by more expressive descriptions (RDF graphs), based on ontologies; the PASS fuzzy ontology of term associations is similar to the Corese *see-Also* network of concepts and the measure of the so-called *narrower* and *broader relations* between terms would correspond to our semantic distances between the only concepts related by *see-Also* relations - and not between any two concepts in the ontology.

C. Ontology Alignment or Versioning

Last, an analogy could be made with the mapping between classes of two ontologies to be aligned or with the comparison of two versions of the same ontology. The various approaches for ontology alignment (see the state of the art on current alignment techniques provided by the Knowledge Web network[7] or the PromptDiff algorithm heuristic matchers [11] for finding the differences of two versions of the same ontology could be useful if Corese aimed at finding an alignment between the ontology and the user's (implicit) personal ontology. But Corese rather focuses on finding the RDF annotations the closest "semantically" (i.e. w.r.t. the ontology and our ontological distance) to the user's query.

VI Conclusion

We have presented the Corese ontology-based Web search system whose query language handles RDF annotations, RDFS and some major features of OWL Lite. We have stressed the need for approximation in querying the semantic web and detailed the mechanism Corese integrates to provide a generic scheme for approximate search: a semantic approximation based on the definition of an ontological distance which enables us to sort approximate answers by decreasing similarity from the query, and on the definition of relation paths which enables us to approximate the structure of the searched annotations.

[7] http://knowledgeweb.semanticweb.org/semanticportal/home.jsp

Beside ontology-based Web search, Corese definition of semantic distances between concepts could be integrated in existing alignment techniques. On the other hand, we could benefit from such alignment techniques for integrating other aspects than simple structural distance and ontology depth in the Corese semantic distance.

We are also currently exploring how to specify semantic distances or semantic heaps between classes in the ontology depending on viewpoints to take into account different user profiles in the query processing. With this very same goal, we aim at contextualizing the distance of the *seeAlso* property and make it depend on user profiles or user tasks. This will enable us to integrate user profile features into the Corese query language.

Acknowledgements

We deeply thank Olivier Savoie in participation to Corese implementation, INRIA that funded him, and Alain Giboin, Nicolas Gronnier, Cécile Guigard and Karine Delêtre for KMP evaluation.

References

[1] J. Broekstra, A. Kampman, F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In ISWC'2002, p. 54-68, 2002.

[2] O. Corby, R. Dieng-Kuntz, C. Faron-Zucker. Querying the Semantic Web with the Corese Search Engine. In ECAI'2004, Valencia, 2004, IOS Press, p. 705-709.

[3] O. Corby, R. Dieng-Kuntz, C. Faron-Zucker, F. Gandon. Ontology-based Approximate Query Processing for Searching the Semantic Web with Corese. INRIA Research Report RR-5621, July 2005.

[4] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In Semantic Issues in Multimedia Systems. Kluwer Academic Publisher, 1999, p. 351-369.

[5] R. Dieng-Kuntz, O. Corby. Conceptual Graphs for Semantic Web Applications. In ICCS'2005, LNAI 3596, Kassel, July 2005, p. 19-50.

[6] A. Giboin, F. Gandon, O. Corby, R. Dieng. Assessment of Ontology-based Tools: Systemizing the Scenario Approach. In EKAW'2002 EON Workshop, p. 63-73, 2002.

[7] N. Guarino, C. Masolo, G. Vetere. Ontoseek: Content-based access to the Web. In IEEE Intelligent Systems, 14(3):70-80, 1999.

[8] M. Holi, E. Hyvönen. A Method for Modeling Uncertainty in Semantic Web Ontologies. In WWW'2004 Alternate track papers & posters, 2004, p. 296-297.

[9] P. Martin, P. Eklund. Knowledge Retrieval and the World Wide Web. In IEEE Intelligent Systems, 15(3):18-25, 2000

[10] L. Miller, A.Seaborne, A. Reggiori. Three Implementations of SquishQL, a Simple RDF Query Language. In ISWC'2002, LNCS 2342, p. 423-435, 2002.

[11] N. F. Noy, M. A. Musen. Ontology Versioning in an Ontology Management Framework. In IEEE Intelligent Systems, 19(4):6-13, 2004.

[12] D.H. Widyantoro, J. Yen. A Fuzzy Ontology-based Abstract Search Engine and its User Studies. In 10th IEEE Int. Conf. on Fuzzy Systems, p. 1291-1294, 2001.