Searching the Semantic Web: Approximate Query Processing Based on Ontologies

Olivier Corby, Rose Dieng-Kuntz, and Fabien Gandon, *INRIA*

Catherine Faron-Zucker, *I3S, UMR CNRS*

The Corese ontology-based search engine handles RDF Schema, part of OWL Lite and RDF metadata, and can process approximate queries, resulting in more efficient Web searches.

Th Semantic Web aims to represent the contents of Web resources in formalisms that both programs and humans can understand. It relies on rich metadata, called *semantic annotations*, offering explicit semantic descriptions of Web resources. These annotations are built on ontologies, representing domains through their concepts and the semantic relations between them. Ontologies are the foundations of the Semantic Web and the keystone of the Web's automated tasks—searching, merging, sharing, maintaining, customizing, and monitoring.

Our work focuses on searching as needed in Web applications such as digital libraries, Web intelligence, and corporate intranets for knowledge management. Publishing languages such as HTML let us retrieve documents on the basis of their presentation and textual contents. Structuring languages such as XML or SGML let us access Web resources on the basis of their data structure. Semantic annotations improve Web searches by letting us access Web resources on the basis of their semantic descriptions.

Here, we address the problem of a dedicated ontology-based query language. Ontologies ensure an efficient retrieval of Web resources by enabling inferences based on domain knowledge. However, the vision of the Semantic Web implicitly relies on the assumption that an ontology designed to describe a domain can both annotate and retrieve Web resources. In reality, this isn't always the case, because domain specialists usually build the ontologies, and users don't always share or understand their viewpoints. Users might not use the right concepts—from an ontologist's viewpoint—when writing a query, leading to missed answers. For example, a user might use *commerce* instead of *business*. Or, perhaps a user asking for a *person* working on a *subject* might also appreciate the retrieval of a *research group* working on that subject.

Consequently, approximate-query processing is of prime importance for efficiently searching the Semantic Web. Our Corese ontology-based search engine handles RDF Schema, OWL Lite, and RDF metadata, and its query language enables both ontological and structural approximations. Several real-world projects using Corese illustrate its potential.

Ontology-based Web search

Ontologies let us take into account, during 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, and so forth. This knowledge supports inferences that improve the matching process's efficiency.

The following logical model expresses the use of ontological knowledge in Web search approaches. Assume we have a model for ontologies, a model for annotations of Web resources based on ontologies, a model for queries based on ontologies, and a matching function defining how a query is matched with any annotation. In this case, a Web resource R is relevant for a query Q according to the ontology O from which they're built if and only if the annotation of R and the ontology O together logically imply Q (noted $R \wedge O \rightarrow Q$).

We can view the query as a set of constraints on the description of the Web resources to be retrieved, which corresponds to a search problem. The matching function implements the strategy chosen to solve this problem. It differs from one search system to another, depending on the formalism chosen for the descriptions, the types of query, and the requirements the results must satisfy. To implement such a matching function, Corese uses the projection operator defined in the *conceptual graphs* formalism (www.jfsowa.com/cg/cgstand.htm).[1]

Corese's theoretical foundations

The Corese semantic search engine internally works on CG. When matching a query with an annotation according to a shared ontology, the search engine translates the query, annotation, and ontology into the CG model. CG and RDF Schema models share many common features, and we established a mapping between RDF Schema and a large subset of the CG model. An in-depth comparison of both models was our starting point for Corese.

Both models distinguish between ontological and assertional knowledge. Assertional knowledge is positive, conjunctive, and existential and is represented by oriented labeled bipartite graphs. Corese translates an RDF graph G representing an annotation or a query into a CG. Regarding the ontological knowledge, the class (respectively, property) hierarchy in a RDF Schema corresponds to the concept (respectively, relation) type hierarchy in a CG support (an ontology). RDF properties are declared as first-class entities such as RDF Schema classes, similar to how a CG support declares relation types independently of concept types. This common handling of properties makes the mapping very relevant, as opposed to in an object-oriented language, where properties are defined inside classes.

Some differences exist between RDF Schema and CG models in their handling of classes and properties, but we can easily manage these differences. 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, Corese can translate an RDF property's multiple domain (respectively, range) constraints into a single domain (respectively, range) constraint in CG by generating the concept type corresponding to the most general specialization of the concept types constraining the domain (respectively, range).

As a result, searching RDF Schema through CG consists of compiling the CG support's type hierarchies, associating a compiled type to each resource, and using the CG model's projection operation 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 if and only if it's a specialization of G_2 (noted $G_1 \leq G_2$). A CG G_1 is a specialization of G_2 if and only if a projection of G_2 into G_1 exists such that each concept or relation node of G_2 is projected on a node of G_1. This node's type must be the same as the type of the corresponding node of G_2 or a specialization of it, according to the concept- and relation-type hierarchies.

When matching a query with an annotation according to a shared ontology, the Corese semantic search engine translates the query, annotation, and ontology into the Conceptual Graph model.

Corese retrieves the resources that have a projection of the query graph onto their annotation graphs. For example, the following query graph lets us search for science resources and their authors:

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

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

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

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

The Corese ontology representation language

Corese's first ontology representation language was RDF Schema. We've progressively extended the language to handle some major features of OWL Lite. We chose RDF Schema mainly because the first implementations of Corese with RDF Schema came before OWL. However, the different projects experimenting with Corese have shown that RDF Schema's expressivity is sufficient in many applications—if extended with inference rules and approximation in the query language. We think that OWL Lite features are sufficient to handle most knowledge representation problems encountered in Semantic Web applications. Corese provides OWL value restrictions, class intersection, subclasses, and algebraic properties such as transitivity, symmetry, and inverse. It also provides annotation, versioning, and ontology OWL statements. Corese doesn't yet provide loops in a subsumption hierarchy or statements from OWL such as cardinality restrictions, property and class equivalences, or the sameAs statement.

These extensions to OWL features are based on domain axioms, and Corese integrates an inference engine based on forward-chaining production rules. Corese applies the rules once it loads the annotations, before query processing, so it enriches the annotation graphs before projecting the query graph. This is the key to Corese's scalability to the Web application in which we've used it.

Furthermore, Corese implements CG rules. For instance, the following CG rule states that if a person *?m* is head of 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's 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 might lead to specializing the types of some concepts or to creating new individual concepts or relations between concepts.

We based the Corese rule language on RDF's triple model. For instance, the CG rule we just presented translates this 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>
```

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, variable, or comparison operator; the third is a variable, value, or 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—for example, dc as http://purl.org/dc/elements/1.1/.

Variable names begin with a question mark. Values are typed with the XML Schema data types: numerical, xsd:string, xsd:boolean, and xsd:date. We can specify the language of the literal's value using an @ operator and the constants defined for xml:lang. For instance, we can request that a thesis's title be in English using ?doc kmp:Title ?t@en.

The comparison operators for equality and difference (=, !=), ordering (<, <=, >, >=), and string inclusion and exclusion (~, !~) let us compare a variable with a constant or another variable. For instance, we can state the title must include the term "Web" using ?t ~ "Web".

Type comparators (<:, <=:, =:, >=:, >:) and combinations with the ! (negation operator) let us specify constraints on some types in a query. For instance, we can constrain the document to be a strict specialization of a thesis (such as a PhD or an MSc thesis) using ?doc <: kmp:Thesis.

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

Corese's core query language addresses possible mismatches between end-user and ontologist concepts by approximating the query's semantics, structure, or both.

The Corese query language supports queries on ontologies just as it does on annotations because 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, a Corese query returns all the variables' values. A select operator lets us list the values desired in the answers. For instance, we can choose to only return the documents' title and authors using select ?t ?n.

A group operator lets us group the retrieved answers according to one or more concepts instead of separately listing answers about the same concepts. For instance, when looking for documents on a specific subject written by an author, a group on the ?doc variable will avoid returning a document written by several authors for each of the authors.

A count operator, combined with group, counts the different answers retrieved. For instance, to count the number of documents written by a person, Corese applies count to the variable ?doc and group to the variable ?p. Finally, we're integrating SPARQL syntax for the query language into Corese.

Approximate Semantic Web search

We have extended Corese's core query language to address possible mismatches between end-user and ontologist concepts. Corese can cope with queries for which no exact answer exists by approximating the query's semantics, structure, or both.

Ontological approximation

The first principle of the Corese semantic approximation is to evaluate semantic distances between ontological types. On the basis of this ontological distance, Corese retrieves not only Web resources whose annotations are specializations of the query but also those whose annotations are *semantically close*.

Ontological distance. To evaluate conceptual relatedness, Corese relies on the ontology's structure. In CG, structured-based distances are the key to defining a nonbinary projection—for example, 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.

We start with the fact that in an ontology, low-level classes are semantically closer than top-level classes. For example, *TechnicalReport* and *ResearchReport*, which are brothers (that is, subclasses of the same class) at depth 10, are closer than *Event* and *Entity*, which are brothers at depth 1. So, 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 $d_H(\mathrm{T}) = 0$, $\forall x \in H$, $d_H(x) \leq d_H(\bot)$, and $\forall (x, y) \in H^2$, $y < x \rightarrow d_H(y) < d_H(x)$; T and $\bot$ being the hierarchy's root and the bottom).

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

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

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 supertype:

$$D_H(t_1, t_2) = \min_{\{t \geq t_1, t \geq t_2\}} (l_H(< t_1, t >) + (l_H(< t_2, t >))$$

We proved that D_H is a semidistance, a distance that doesn't match the triangle inequality in the general case.[2]

Contextual closeness. The ontological distance between two classes isn't always sufficient to render the closeness of some concepts. We've encountered cases where concepts are distant in the ontology but share some features that make them close from the search viewpoint. 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 the rdfs:subClassOf link doesn't express. When querying for *KnowledgeDissemination*, a user might want to retrieve *KnowledgeEngineering* resources in case of failure. Similarly, some properties might 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.

Corese can express relatedness using the standard rdfs:seeAlso property. It can add this property to any existing RDF Schema. So, it can parameterize a given ontology to better fit a specific Web search task or a particular user class. This addition not only improves browsing capabilities but also shortens the semantic distance and tunes approximate matching. It's worth considering having the subclasses and subproperties inherit the rdfs:seeAlso property. So, 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
```

Approximate projection. On the basis of the ontological distance, Corese distinguishes between *exact answers* for which there exists a projection of the query upon the answers' annotations and *approximate answers* for which there exists an *approximate projection* of the query upon the answers' annotations. These annotations have a structure upon which the query can be projected, but their concept and relation types aren't necessarily subsumed by those of the query: they are considered "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. This mapping

- preserves adjacency and order on edges,
- might 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), and
- might decrease the labels of relation nodes or change them to contextually close ones (for which a seeAlso property stands).[2]

Corese measures the relative relevance of the retrieved annotations by their similarity to the query. It presents to the user annotations whose similarity doesn't exceed a given threshold.

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

Setting an rdfs:seeAlso property between two concept types c_1 and c_2 should shorten the ontological distance between them to a brotherhood distance. Consequently, it should increase the similarity between the 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 an rdfs:seeAlso property between two relation types r_1 and r_2 is also taken into account when computing 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 $d = \max(d_H(c_1), d_H(c'_1))$—that is, the maximum depth of the types c_1 and c'_1 of the neighbor concept nodes of r_1.

Corese measures the relative relevance of the retrieved annotations by their similarity to the query. It presents to the user annotations whose similarity doesn't exceed a given threshold, sorted by decreasing similarity. This threshold is relative to the query's best-found approximation.

Syntactically, the more keyword in a Corese query's select clause asks for approximate answers. In this case, Corese basically approximates every query concept. However, its query language lets us require the specialization of some concepts while approximating others using type comparators. For instance, using the <=: operator, Corese can retrieve people interested in knowledge engineering (or something close) and members of a related 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.

Structural approximation

The ontology-based approximation we've described makes up for the possible divergences between the vocabularies. Another kind of approximation that the Corese query language supports 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—in other words, how the annotator has described it. For example, the user might search for organizations related to *human science*, whatever the relationship is.

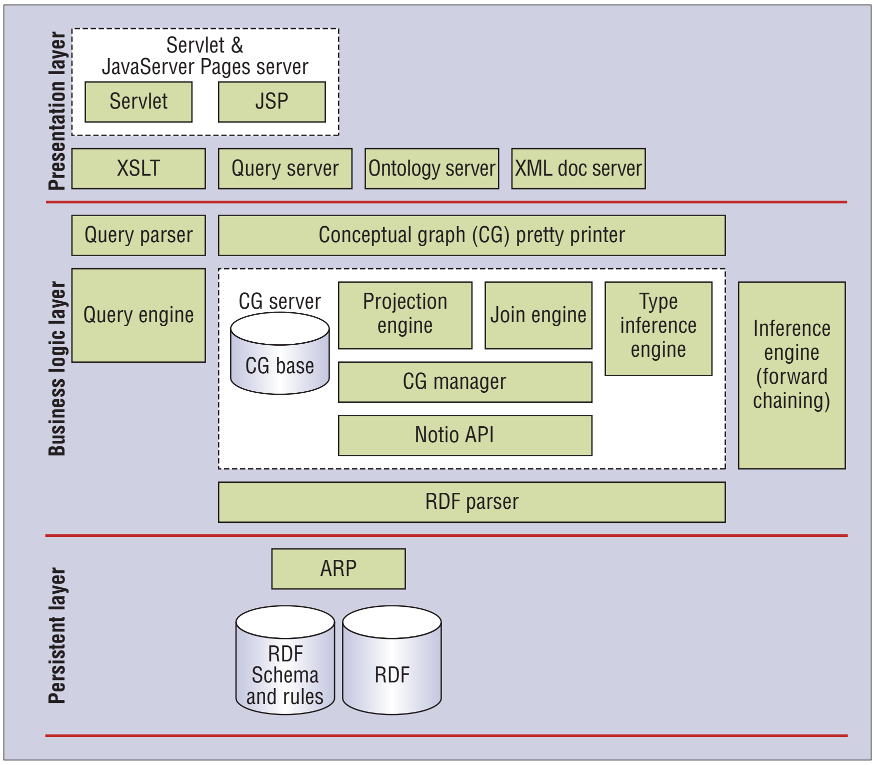

Figure 1. Corese's three-tier architecture.

This kind of approximation concerns the annotations' structure but still remains semantic. We can view it as the approximation of a complex relationship that can't be represented by a single property and that requires a graph to define it.

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

We've extended our definition of an approximate projection of a CG G to a CG H to allow the mapping of a relation node with a path graph. This extension preserves adjacency and order on edges, 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.[2]

Syntactically, in the Corese query language, the path graph expression must be suffixed by the path's maximal length. By default, Corese stops after retrieving one path (with the shortest length). It computes all possible paths when the **all** qualifier prefixes the relation.

Consider the following query asking for organizations related to *human science* by a (nondirected) relation path of a length less than 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 the previous query: they express that the CNRS institute is interested in *human science* and a member of the INRIA institute graduated in *human science*.

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

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

Corese's architecture

We developed Corese in Java, and it's publicly available under the INRIA license at www.inria.fr/acacia/corese (including Java packages, documentation, and the GUI). We've also developed a Corese server according to a three-tier architecture (see figure 1).

The *presentation layer* generates the content that the user's browser will present (ontology views and browsing controls, query edition interfaces, annotation forms, answers, and so forth). This layer relies on a model-view-controller architecture to handle HTTP requests and generate responses fed by the *business logic layer's* appropriate Corese services. The responses are formatted using Extensible Style Sheet Language Transformation or JavaServer Pages templates. Servlets implement the presentation layer, which provides

- the front end of what we call a Semantic Web server—that is, an HTTP server that can solve Semantic Web queries submitted through HTTP requests;
- JSP tags to include Semantic Web processing and render results in Web pages;
- XSLT extensions to perform Semantic Web functions related to XPath expressions, thus improving RDF/XML transformation capabilities; and
- a form description language to dynamically build forms using queries—for instance, to populate the different choices of a drop-down box.

The *business logic layer* consists of a platform that implements three main services accessible through an API: a CG server (using the Notio API, www.cs.ualberta.ca/~finnegan/notio), 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 manages the CG base, the projection and join operators, and the 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 developers can use to add Semantic Web capabilities to their applications.

In the *persistent layer*, Corese can access RDF Schema data using the ARP (Another RDF Parser) parser (www.hpl.hp.com/personal/jjc/arp), and the RDF-to-CG parser translates the data. Rules are saved in separate files and parsed by the rule parser.

A real-world application

We've tested Corese on several real-world, large-scale applications with ontologies (see the "Applications" sidebar).[3] Here we discuss how we used Corese to build a *knowledge management platform* for a mapping of telecommunications skills for Sophia Antipolis firms (see www-sop.inria.fr/acacia/soft/kmp.html). To foster synergies and partnerships in the Telcom Valley community, the KMP system pro-

Applications

In the main text, we discuss how we've used Corese to build a knowledge management platform for a mapping of telecommunications skills in for Sophia Antipolis firms. Other projects using Corese include the Samovar (Systeme d'Analyse et de Modèlisation des Validations des Automobiles Renault) system, which supports a vehicle project memory for Renault. The ontology has 792 concept types and four relation types and annotates 4,483 problem descriptions. Corese answers queries such as, "Find all problems that occurred on the dashboard in a past project."

The CoMMA (Corporate Memory Management through Agents) IST project involves a multiagent system for corporate memory management with two scenarios: integration of a new employee and technological watch. The O'CoMMA ontology comprises 472 concept 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 users who might be interested in the technological news that was just submitted about GSM v3."

The Escrire project involves annotating and searching the Medline database's genetics abstracts. Corese answers queries such as, "Find articles describing interactions where the Ubx gene acts as a target and where the instigator is either an *en* or *dpp* gene."

The Ligne de Vie project involves a virtual staff for a health network relying on an ontology comprising 26,432 concept types and 13 relation types. It guides physicians discussing possible diagnoses and alternative therapies for a given pathology according to the patient's features. It can answer queries such as "Find the past sessions of virtual staff where they chose a given therapy for the patient, and indicate the arguments in favor of this therapy."

The Meat (Memoire d'Experiences su l'Analyse due Transcriptome) project is creating a memory of the experiments performed on a DNA microarray, relying on annotations on scientific articles and using the Unified Medical Language System as an ontology. Corese answers queries such as "Find all the articles asserting that HGF gene plays a role in lung disease."

We've also tested Corese with other ontologies such as the Gene ontology (represented by an RDF graph with 13,700 concept types and 950,000 relations), IEEE Learning Object Metadata, W3C Composite Capability/Preference Profiles, and the Dublin Core.

vides a dynamic map of different stakeholders' competencies. The KMP solution relies on specifying, designing, building, and evaluating an online customizable service.

This service is becoming the main component of a portal for the community of industries, academic institutes, and institutional organizations involved in the Telecom Valley of Sophia Antipolis. The KMP project is a real-world experiment, and the steering committee comprises 10 pilot companies — Amadeus, Philips Semiconductors, France Telecom R&D, Hewlett Packard, IBM, Atos Origin, Transiciel, Elan IT, Qwam System, and Cross Systems.

The KMP provides clustering views to analyze the competencies in the Telecom Valley. The screenshot in figure 2 shows one of these views, also called a *cluster*. This cluster presents a distribution of *grapes* corresponding to resources involved in each competence area (such as telecommunications or computer science). Each grape contains bubbles representing actions (for example, "produce" or "design") involved. The integrated RDF Schema data collected then dynamically generates a scalable-vector-graphics view, which provides a powerful representation for analyzing the Telecom Valley's diversity. The grouping of competencies relies on the ontology-based distance defined in Corese to evaluate the conceptual similarities between competencies.

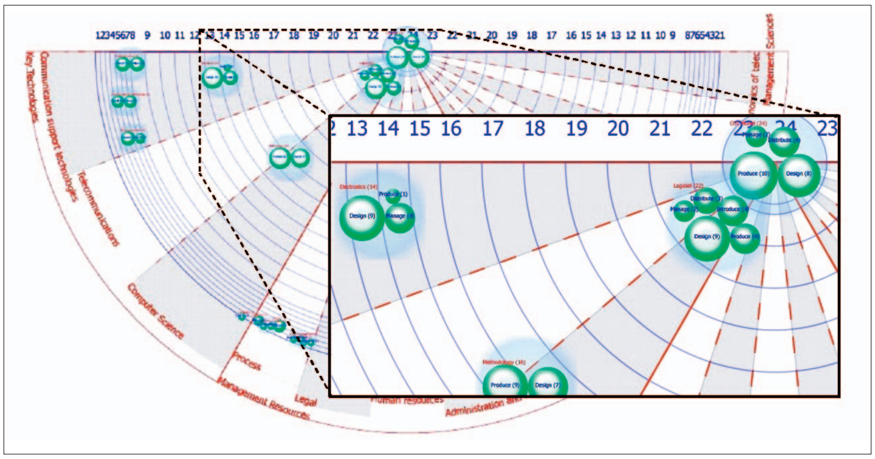

Figure 2. Conceptual clustering of the competencies of the Sophia Antipolis Telecom Valley.

To test Corese's approximate reasoning, at the end of the KMP project, we used a variant of KMP for an intra-enterprise skills management scenario. We ran a query for people who are Java programming experts and are interested in XML. Such a query might aim to find profiles for building project teams or managing mobility in a company. A basic exact retrieval in the annotation base—a base comprising descriptions of competencies of employees in a company—couldn't answer this query, but Corese retrieved eight answers. One exact answer was the result of applying a domain rule of the ontology, stating that the author of a thesis on a given subject is an expert on that subject: Yvonne Duchard wrote a thesis on Java programming, so she is considered an expert in that area.

Moreover, Corese extended this answer using an interesting approximation: in addition to XML, Yvonne Duchard is interested in (aware of) the Wireless Application Protocol, which is close enough to XML for the semantic search in the ontology. This shows how Corese supports serendipity. The seven other answers approximately matched the query, and two had annotations with the same similarity to the query. One was an engineer skilled in both XML and Java programming; the other was a project manager skilled in both XML and Enterprise JavaBeans programming. In both cases, IsSkilledIn approximated the IsExpertIn property.

Evaluation

We evaluated Corese from a systems viewpoint (performance) and from an end-user viewpoint (scenario-based evaluation). For the latter, we used the KMP project.

Corese performance

We measured Corese engine performance on an RDF Schema or RDF base comprising 19,000 properties, 8,000 resources, and 10 rules. The Corese standard test base of 262 queries covering all of the query language's features runs in 9.7 seconds on a laptop. The average answer time is 0.037 seconds per query. The efficient projection operator lets Corese achieve good performances in real-world applications.

Scenario-based evaluation

Corese users gave it a positive evaluation once they received domain axioms, approximate queries, and presentation capabilities (features they really required). (Details for the scenario-based evaluation used for several applications, including CoMMA and KMP, appear elsewhere.[4])

The evaluation of the KMP application involved 10 mediators and approximately 30 users from 17 organizations.

Users emphasized that

- the Corese query language is powerful and effective,
- the ontology-driven user interface forms are useful and user-friendly because they hide the ontology structure's complexity, and
- Corese's approximate search feature was unique and useful, letting them find the best match for any query with the ontology.

Users also suggested several useful improvements for the KMP system. First, some users wanted more dynamic interactions in the query-answer cycle. They wanted to be able to easily refine a query from the answer. Once their query was refined, they wanted Corese to enhance the differences in the answer. They also wanted the system to manage a history of queries.

Second, some users thought we could improve the ordering of approximate answers, and they wanted the system to justify the proposed approximations. Some users also wanted to be able to tune the approximation—for example, which concept can be approximated and how. After receiving a result, they wanted to be able to document the distance of each approximate concept to its query concept.

Finally, some experiments showed that the generic distance wasn't always completely accurate: sometimes a class is closer to its brother class than to its direct ancestor. This led us to more work on distance modeling in ontologies.

Overall, we concluded that although ontology-driven tools are powerful and useful, the user, task, and domain models, not the ontology, should drive user interaction.

In addition to using Corese for an ontology-based Web search, we could also integrate Corese definitions of semantic distances between concepts into existing alignment techniques. We might also benefit from using such alignment techniques to integrate aspects other than simple structural distance and ontology depth into the Corese semantic distance. (In the "Related Work" sidebar, we compare Corese to other work.)

We're 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. We aim to contextualize the distance of the seeAlso property, making it more dependent on user profiles or tasks. This will let us integrate user profile features into the Corese query language.

The Authors

Olivier Corby is a researcher on the ACACIA team at INRIA Sophia Antipolis. His research interests include software development environments for knowledge modeling and Semantic Web technologies for corporate-memory management. He's the designer and main developer of the Corese semantic search engine. He received his PhD in computer science from the University of Nice—Sophia Antipolis. Contact him at INRIA-ACACIA Project, 2004 route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex, France; olivier.corby@sophia.inria.fr.

Rose Dieng-Kuntz is an INRIA research director and ACACIA team leader at INRIA Sophia Antipolis. Her research interests include knowledge engineering, knowledge management, ontologies, the Semantic Web, conceptual graphs, and multiagent systems. She received her PhD in computer science from the University of Paris-Sud. She won the 2005 Irène Joliot-Curie award. Contact her at INRIA-ACACIA Project, 2004 route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex, France; rose.dieng@sophia.inria.fr.

Catherine Faron-Zucker is an associate professor at the University of Nice—Sophia Antipolis. She carries out her research in the M@inline team at I3S and in the ACACIA team at INRIA Sophia Antipolis. Her research interests include knowledge representation (particularly conceptual graphs and description logics), ontologies, the Semantic Web, and e-learning. She received her PhD in computer science from the University Pierre et Marie Curie. Contact her at I3S, M@inline team, 930 route des Colles, BP 145, 06903 Sophia Antipolis Cedex, France; faron@essi.fr.

Fabien Gandon is a researcher at INRIA Sophia Antipolis. His research interests include knowledge engineering, ontologies, the Semantic Web, multiagent systems, Web services, mobile networks, context-awareness, and privacy. He designed and developed the Corese Semantic Web server. He received his PhD in computer science from the University of Nice—Sophia Antipolis. He's a member of the W3C Semantic Web Best Practices and Deployment working group. Contact him at INRIA-ACACIA Project, 2004 route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex, France; fabien.gandon@sophia.inria.fr.

Acknowledgments

We thank Olivier Savoie for his participation in implementing Corese, INRIA for funding him, and Alain Giboin, Nicolas Gronnier, Cécile Guigard, and Karine Delêtre for their participation in the evaluation of the KMP project.

References

1. O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker, "Querying the Semantic Web with the Corese Search Engine," *Proc. 16th European Conf. Artificial Intelligence* (ECAI 04), 2004, IOS Press, pp. 705–709.

2. O. Corby et al., *Ontology-Based Approximate Query Processing for Searching the Semantic*

Related Work in Query Languages and Ontologies

Here we review some of the many query languages and ontology-based search tools designed for the Semantic Web.

RDF query languages

The Corese RDF query language is close to the RDF Data Query Language, the Sesame language SeRQL[1] (www.openrdf.org/doc/users/ch06.html), and SPARQL (Simple Protocol and RDF Query Language, www.w3.org/TR/rdf-sparql-query), which might become a W3C recommendation to query RDF.

Like SPARQL, the Corese query language is based on a Boolean combination of triples that computable expressions can constrain. Corese also processes data-typed RDF literals, optional patterns, and alternatives. Corese returns an RDF/XML graph or an XML binding format, and the bindings are available through an API. Corese also provides the select, distinct, and order statements, and an equivalent of limit statements but not the construct, describe, and ask SPARQL statements (although Corese can simulate the last two).

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

Ontology-based Web search applications

One previous ontology-based Web search application is OntoBroker, which expresses ontologies and queries in Frame Logic and translates them into Horn Logic.[2] Others include Sesame[1] and RDQL,[3] which rely on database management systems to store and query RDF Schema or RDF.

In addition to these general-purpose reasoners, WebKB[4] and OntoSeek[5] are search-oriented applications based on conceptual graphs. WebKB interprets statements expressed in a CG linear notation and embedded in HTML documents; it can query lexical or structural properties of HTML documents. OntoSeek focuses on lexical and semantic constraints when encoding resources into CG and building queries.

WebKB, OntoSeek, and Corese are all built on 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 handles RDF Schema or RDF data like Corese does, and they don't handle rules in their ontology representation language. Moreover, both focus on annotation and ontological problematics and lack an expressive query language.

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's the only Web search application addressing structural approximation of queries. Some recent methods address ontological approximation for searching the Web; one example approximates overlap between RDF Schema concepts based on Bayesian networks.[6] This method suggests applying such approximation to define a semantic distance between concepts and sort the answers to an ontology-based search. However, this method hasn't actually been applied to Web search and focuses on overlap rather than subsumption.

The PASS (Personalized Abstract Search Services) system searches abstracts of research papers, using a fuzzy ontology of term associations for query refinement.[7] 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. Also, the measure of the so-called *narrower* and *broader relations* between terms would correspond to our semantic distances only between the concepts related by see-Also relations, not between any two concepts in the ontology.

Ontology alignment or versioning

Our work might be considered as having some analogies with the mapping between classes of two ontologies to be aligned or with the comparison of two versions of the same ontology. The various ontology alignment approaches (see the state of the art on current alignment techniques provided by the Knowledge Web network, http://knowledgeweb.semantic) or the PromptDiff algorithm heuristic matchers[8] for finding the differences between two versions of the same ontology could be useful if Corese aimed to find an alignment between the ontology and the user's (implicit) personal ontology, but it doesn't. Instead, Corese focuses on finding the RDF annotations closest "semantically" (that is, with regard to the ontology and our ontological distance) to the user's query.

References

1. J. Broekstra, A. Kampman, and F. van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," *Proc. Int'l Semantic Web Conf.* (ISWC 02), Springer, 2002, pp. 54–68.
2. S. Decker et al., "OntoBroker: Ontology-Based Access to Distributed and Semi-structured Information," *Semantic Issues in Multimedia Systems*, Kluwer Academic, 1999, pp. 351–369.
3. L. Miller, A. Seaborne, and A. Reggiori, "Three Implementations of SquishQL, a Simple RDF Query Language," *Proc. Int'l Semantic Web Conf.* (ISWC 02), LNCS 2342, Springer, 2002, pp. 423–435.
4. P. Martin and P. Eklund, "Knowledge Retrieval and the World Wide Web," *IEEE Intelligent Systems*, vol. 15, no. 3, 2000, pp. 18–25.
5. N. Guarino, C. Masolo, and G. Vetere, "OntoSeek: Content-Based Access to the Web," *IEEE Intelligent Systems*, vol. 14, no. 3, 1999, pp. 70–80.
6. M. Holi and E. Hyvönen, "A Method for Modeling Uncertainty in Semantic Web Ontologies," *Proc. 13th Int'l Conf. World Wide Web* (WWW 04), ACM Press, 2004, pp. 296–297.
7. D.H. Widyantoro and J. Yen, "A Fuzzy Ontology-Based Abstract Search Engine and Its User Studies," *Proc. 10th IEEE Int'l Conf. Fuzzy Systems*, IEEE Press, 2001, pp. 1291–1294.
8. N.F. Noy and M.A. Musen, "Ontology Versioning in an Ontology Management Framework," *IEEE Intelligent Systems*, vol. 19, no. 4, 2004, pp. 6–13.

Web with Corese, research report RR-5621, INRIA, 2005.

3. R. Dieng-Kuntz and O. Corby, "Conceptual Graphs for Semantic Web Applications," *Proc. 13th Int'l Conf. Conceptual Structures* (ICCS 05), LNAI 3596, Springer, 2005, pp. 19–50.
4. A. Giboin et al., "User Assessment of Ontology-Based Tools: A Step Towards Systemizing the Scenario Approach," *Proc. EON 2002: Evaluation of Ontology-Based Tools, OntoWeb-SIG3 Workshop 13th Int'l Conf. Knowledge Eng. and Knowledge Management,* 2002, pp. 63–73.