Querying the Semantic Web with Corese Search Engine

Olivier Corby, Rose Dieng-Kuntz, Catherine Faron Zucker

Querying the Semantic Web with Corese Search Engine

Olivier Corby [1] and **Rose Dieng-Kuntz** [2] and **Catherine Faron-Zucker** [3]

Abstract. This paper presents an ontology-based approach for web querying, using semantic metadata. We propose a query language based on ontologies and emphasize its ability to express approximate queries, useful for an efficient information retrieval on the web. We present the Corese search engine dedicated to RDF(S) metadata and illustrate it through several real-world applications.

1 INTRODUCTION

The present Web comprises a huge amount of heterogeneous data (structured data, semi-structured data, textual data, multimedia data), dedicated to human users of the Web. The Semantic Web [1] aims at enabling the semantic contents of Web resources to be also processed by automated tools. It relies on rich metadata, also called semantic annotations, offering explicit semantic descriptions of Web resources and built on domain ontologies.

In this paper, we focus on information retrieval (IR) on the semantic web. This specific kind of IR is needed in web applications such as web browsing, digital libraries, knowledge management (KM), E-learning, e-commerce, etc. Web users aim at retrieving resources or services satisfying specific criteria or constraints.

IR on the Semantic Web can be addressed according to three different points of view: *developers of ontologies* focusing on the representation of domain knowledge, *annotators of web resources* creating semantic annotations based on ontologies, and *end-users* asking ontology-based queries for searching the web. Previous work on ontology-guided IR (SHOE [16], OntoBroker [8], OntoSeek [13], WebKB [17], Corese [3, 4]) mainly focused on ontology knowledge representation (KR) languages. In this paper, we rather focus on the *query processing point of view* and we address the problem of a *dedicated ontology-based query language*.

After showing how ontologies ensure an efficient retrieval of web resources by enabling inferences based on domain knowledge, we present the Corese search engine and its query language dedicated to the retrieval of web resources annotated in RDF(S). Then we describe Corese's approximate query processing capabilities. Last, we present real-world applications of Corese.

2 ONTOLOGY-BASED IR

2.1 A Logic based Approach

Ontology-based IR stems from a logical model as defined in [20]: given (1) a model for the descriptions of documents, (2) a model for the queries, and (3) a matching function that defines how a query is matched with any description, a document D is relevant for a query Q if the description of D logically implies Q (D $\rightarrow$ Q).

In this model, a query is viewed as a set of constraints on the description of the documents to be retrieved and then correspond to a search problem to be solved. The matching function thus implements the strategy chosen for solving this problem. It differs from an IR system to another, depending on the KR formalism chosen for the document descriptions and the queries.

For IR on the semantic web, ontologies enable to take into account in the query processing some knowledge implicit in the annotations of the web resources. It comprises subsumption links between domain concepts and between domain relations, other semantic links between domain concepts, domain axioms or rules enabling deductions on semantic annotations. This domain knowledge enables to retrieve web resources while using in the query terms maybe different from - but semantically related to - those of the annotation, and to perform inferences improving document retrieval.

The use of ontological knowledge in the query processing is expressed in the following IR model: (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 that defines how a query is matched with any annotation. Given this model, a web resource R is relevant for a query Q iff R satisfies Q according to the ontology O from which both the annotation of R and the query Q are built. This means that the annotation of R and the ontology O together logically imply Q : $O \wedge R \rightarrow Q$.

2.2 Corese and its Query Language

Corese is an ontology-based search engine for the semantic web: it is dedicated to the retrieval of web resources annotated in RDF(S) [26] by using a query language based on RDF(S). Corese ontology representation language is built upon RDFS, that enables representation of ontologies provided with a concept hierarchy and a relation hierarchy. Corese thus takes into account subsumption links between concepts and between relations when matching a query with an annotation. Corese ontology representation language also enables to represent domain axioms which are taken into account when matching a query with an annotation. Annotations are represented in RDF and related to the RDF Schema representing the ontology they are built upon. The query language is also built upon RDF; for each query, an RDF graph is generated, related to the same RDF Schema as the one of the annotations to which it is to be matched.

The Corese engine internally works on conceptual graphs (CG). When matching a query with an annotation, according to their common ontology, both RDF graphs and their schema are translated in the CG model [24]. Through this translation, Corese takes advantage of previous work of the KR community leading to reasoning capabilities of this language.

[1] INRIA Sophia Antipolis, France email: olivier.corby@sophia.inria.fr
[2] INRIA Sophia Antipolis, France email: rose.dieng@sophia.inria.fr
[3] I3S, University of Nice - Sophia Antipolis, France email: catherine.faron@sophia.inria.fr

2.2.1 RDF(S) and Conceptual Graphs

The RDF(S) and CG models share many common features and a mapping can easily be established between RDFS and a large subset of the CG model. An in-depth comparison of both models was the starting point of the development of Corese [3, 4].

Both models distinguish between ontological knowledge and assertional knowledge. In both models, the assertional knowledge is positive, conjunctive and existential and it is represented by directed labeled 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. property) 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 relevant the mapping of RDFS and CG models, contrarily to object-oriented language, where properties are defined inside classes. For sake of room, we don't detail the few differences between the RDF(S) and CG models in their handling of classes and properties but they can be easily dealed with when mapping both models.

The projection operation is the basis of reasoning in the CG model. A query is thus processed in the Corese engine by projecting the corresponding CG into the CGs translating the annotations. The retrieved web resources are those for which there exists a projection of the query graph into the annotation graph. For example the following query graph :

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

can be projected on the two following annotation graphs:

```
[TechReport]-(createdBy)-[Researcher]
            -(subject)-[CognitiveScience]
```

and:

```
[Book]-(createdBy)-[Professor]
      -(topic)-[SocialScience]
```

In the ontology shared by these annotation graphs and the query graphs, both *TechReport* and *Book* are `subClassOf` *Document*, *Researcher* and *Professor* are `subClassOf` *Person*, *CognitiveScience* and *SocialScience* are `subClassOf` *Science* and *topic* is `subPropertyOf` *subject*. The two previous graphs thus annotate web resources answering the above query and will be retrieved by Corese when processing this query.

2.2.2 Domain Axioms

In addition to a concept hierarchy and a relation hierarchy, a richer ontology is provided with domain axioms that enable to deduce new knowledge. However RDF Schema is not provided with such a feature. Hence we have proposed an RDF Rule extension to RDF and Corese integrates an inference engine based on forward chaining production rules [4]. The rules are applied once the annotations are loaded in the system and before the query processing occurs. Hence, the annotation graphs are augmented by rule conclusions before the query graph is projected on them.

The production rules of Corese implement CG rules [21]: a rule $G_1 \Rightarrow G_2$ is a pair of lambda abstractions $(\lambda x_1, ..., \lambda x_n G_1, \lambda x_1, ..., \lambda x_n G_2)$ where the x_i are co-reference links between generic concepts of G_1 and corresponding generic concepts of G_2 that play the role of rule variables.

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 (if needed, we can add that `?p != ?m` in the condition) :

```
?m rdf:type c:Person
?m c:head ?t
?t rdf:type c:Team
?t c:hasMember ?p
?p rdf:type c:Person
=>
?m c:manage ?p
```

A rule $G_1 \Rightarrow G_2$ applies to a graph G if there exists a projection π from G_1 to G, i.e. G contains a specialization of G_1. 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 specialize the types of some concepts, to create relations between concepts and to create new individual concepts (i.e. concept without variable).

3 APPROXIMATE IR

3.1 Why do We Need Approximation?

The implicit vision of the Semantic Web in the previous section relies on three strong hypotheses:

1. it is possible to design standard conceptual vocabularies (so-called ontologies) to describe a domain objectively,
2. it is possible to describe web resources using these vocabularies,
3. it is possible for users to search information using the same vocabularies as the annotators.

In other words, we have supposed that an ontology designed to describe a domain is useable to both annotate web resources of this domain and retrieve them by semantically querying the web.

Reality is more contrasted. The viewpoint of the designers of ontologies, the viewpoint of the designer of annotations describing web resources and the viewpoint of the user performing IR may not completely match.

Ontologies are models of reality that may be complex. They are built according to some goals, among which (1) identify and describe the objects and relations of a domain in order to promote reuse and shareability, and (2) ease IR of web resources of this domain. Usually, an ontology is built by specialists of the domain, not by specialists of the IR task in this domain, i.e. the users. The user may not share or not understand the viewpoints of the designers: the technical domain modeling does not necessary meet the IR management. There may be some mismatch between the needs of a clean reusable formal ontology and an effective guideline for IR. Sometimes, distinctions made from the ontology viewpoint are not significant from the user viewpoint. Hence, it is difficult to master an ontology of hundreds of concepts.

Some experiments of Corese with the O'CoMMA [9] ontology give us good examples of misunderstanding or misuse by the user of concepts stated by the ontologist: the user used the *Commerce* concept instead of *Business* or *KnowledgeDissemination* instead of *Education.*

Users may not use the *right* concepts - those of the ontologist - when writing a query, and this mismatch may lead to an empty answer to the query.

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. S/he may even appreciate to retrieve a research group working on a similar subject, instead of no answer at all.

So, the core query language of Corese presented above was extended to address this problem of mismatch between the design of ontologies and annotations and the IR activity. Corese is able to provide the user with approximate answers to a query, the semantic distance being computed by using the ontology and the approximation being controlled with comparison operators.

3.2 Semantic Distance

The principle of the Corese approximation is to evaluate the semantic distance of classes or properties in the ontology hierarchies: two brother classes or relations are closer than two cousins, etc. Based on this semantic distance, Corese does not only retrieve web resources whose annotations are *specializations* of the query, it also retrieves those whose annotations have a structure upon which the query can be projected but whose concepts and relations are not necessarily subsumed by those of the query: they are just close enough to them in the ontology hierarchies. The projection of the query upon annotations is thus done free from the subsumption relations between classes and between relations; for each retrieved web resource, its distance to the query is then computed and finally the resources whose semantic distance does not overpass a given threshold are eventually presented to the user, sorted by increasing distance. Furthermore, Corese generates a specific markup on approximate concepts in the output, to ease up their identification and to enable their enhancement at presentation time (with another color or another font).

We define the distance of a web resource to a query as the sum of the distances of its concepts to those of the query that project upon them. If a target concept is a specialization of the query concept that projects upon it, its distance is 0. Otherwise, the distance between two concepts can be defined as the distance between their classes, the distance between two classes being the sum of the distances between each of them and their deepest common super class [10].

But low level classes are semantically closer than top level classes. For example, *TechnicalReport* and *ResearchReport* are closer than *Event* and *Entity*: two brothers are closer at depth 10 than at depth 1. In other words, the distance between classes decreases with depth: the deeper the closer. As a result, following [28], we define the distance between a class and a direct super class of it (separated by a path of length 1) by $1/2^d$, where d is the (maximum) depth of the upper class. Let us note that, because of multiple inheritance, a class may be associated with several depths and we chose to take into account its maximum depth.

Our semantic distance is generic and applies to homogeneous corporate ontologies. The handling of distributed ontologies would require further researches to refine the semantic distance by taking into account the heterogeneous depths of the ontology parts.

3.3 Operators for Tuning Approximation

In approximate mode, Corese basically approximates each concept of the query. However, it is sometimes useful to require specialization of some concepts and only approximate the others. Hence, Corese enables to define which concepts can be approximated and which ones must be found exactly. More generally, it enables to specify conditions on the types that are acceptable and those that are not. For this purpose, we have introduced in the Corese query language a set of type comparison operators that can be associated to each query concept:

`<:` strict subtype, `<=:` subtype, `=:` same type, `>=:` super type, `>:` strict super type. These operators can also be combined with a `!` negation operator, e.g. : `!<:`, `!<=:`, etc.

By using these operators, Corese is able to retrieve, for instance, the persons interested in *KnowledgeEnginering* (or something close) and member of a *Project* (or something close) by processing the following query written in Corese query language :

```
?person c:interestedBy ?k
?person <=: c:Person
?k rdf:type c:KnowledgeEngineering
?person c:member ?project
?project rdf:type c:Project
```

In this query, the `<=:` specialization operator indicates that the class *Person* is required, while *KnowledgeEnginering* and *Project* may be approximate.

The Corese query language offers type operators that support variables to enable the comparison of concepts types in a query. For example, the following query asks to retrieve two documents that must be of the same class:

```
?d1 rdf:type c:Document
?d2 rdf:type c:Document
?d1 =: ?d2
```

3.4 Approximation using related classes or relations

3.4.1 See Also between Classes

The semantic distance of classes in ontologies is not always sufficient to express the proximity of some concepts. In our experiments of Corese, we have often encountered some concepts which were somehow distant from each other in their hierarchy but which shared some features making them closer from IR 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. For example, when querying for *KnowledgeDissemination*, one may want to retrieve *KnowledgeEngineering* resources in case of failure.

Hence, Corese has a second approximation capability by means of the standard `rdfs:seeAlso` property. The effect of a `rdfs:seeAlso` property between two classes is to shorten the actual semantic distance between the two classes in the approximate query processing. For instance, shortening the semantic distance between *KnowledgeDissemination* and *KnowledgeEngineering* is simply achieved by setting a `rdfs:seeAlso` property between these two classes, as shown below:

```
<rdfs:Class rdf:ID='KnowledgeDissemination'>
   <rdfs:seeAlso
     rdf:resource='#KnowledgeEngineering'/>
</rdfs:Class>
```

```
<rdfs:Class rdf:ID='KnowledgeEngineering'/>
```

The semantic distance between two classes linked by a `rdfs:seeAlso` property is shortened to the distance between brothers. In the particular case of a `rdfs:seeAlso` property set between brother classes, the distance between them is divided by two, which makes these brothers closer than other brothers.

The following rule

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

enables to propagate the `seeAlso` property to subclasses.

Corese enables to add a `rdfs:seeAlso` property to an existing RDF Schema, for a specific purpose such as a class of users or specific IR tasks. Hence, an existing ontology can be parameterized to b etter fit a specific IR task.

3.4.2 See Also between Properties

Like classes, some properties may share a semantic proximity from an IR point of view. For instance *isInterestedBy*, *hasForWorkInterest* and *hasForPersonalInterest* are close properties. As for close classes, a `rdfs:seeAlso` property can be set between close properties. Corese handles them by authorizing the occurrence of one of them instead of the other when matching a query with an annotation. For instance, let us put a `rdfs:seeAlso` property in the declaration of the *hasForWorkInterest* property:

```
<rdf:Property rdf:ID='hasForWorkInterest'>
    <rdfs:seeAlso
        rdf:resource='#isInterestedBy'/>
</rdf:Property>
<rdf:Property rdf:ID='isInterestedBy'/>
```

Hence, when answering a query involving the *hasForWorkInterest* property, Corese may return a resource annotated with a *isInterestedBy* property in approximate mode.

The semantic distances in the relation hierarchy of an ontology are computed similarly to the semantic distances in the class hierarchy.

To sum up, `rdfs:seeAlso` property allows an approximation on both the query relations and the query concepts. The relevance of the final answer is guided by the connectedness of the query, its structure, which can always be projected on the structure of any approximate answer.

3.5 Approximation through Relation Paths

Sometimes, the user may search, without success, for resources linked by a conceptual relation. In addition to the approximation of concepts and relations, Corese can search for relation paths of variable lengths between concepts. When querying for x R y, Corese can also generate the query x R z R y (with a relation path of length 2 between x and y), the query x R z R t R y (with a relation path of length 3 between x and y), etc. The length of these relation paths is bounded by a constant value in the query. For instance, the organizations related to Human Science by a relation path of maximum length 3 are retrieved in response to the following query:

```
?org c:relation[3] ?topic
?org rdf:type c:Organization
?topic rdf:type c:HumanScience
```

The default behavior of Corese is to compute the answers with the shortest successful path and to stop. It can also compute all the possible answers with the following syntax : `all::c:relation[3]`. When used with a generic property, this enables to retrieve all connected resources at a given depth.

4 EXPERIMENTATION AND EVALUATION

4.1 Architecture

Corese[4] is implemented in Java and uses the Notio Conceptual Graph API [23]. Corese includes an RDF parser and pretty printer, a query processor, support for a subset of XML Schema datatypes. It also includes an inference rule language and its forward chaining engine. Corese can also be smoothly embedded into a web server.

4.2 Applications

The Corese search engine and its query language have been tested on several real world applications:

- *SAMOVAR*: vehicle project memory system for Renault car manufacturer [12]. The ontology has 792 concepts and 4 relations, and annotates 4483 problem descriptions. Corese answers queries such as: *"Find all fixing problems that occurred on the dashboard in a past project"*.
- *CoMMA*: a multi-agent system for corporate memory management (integration of a new employee and technological watch). The O'CoMMA ontology comprises 472 concepts used for annotating documents or people in an organization [9].
- *KMP: Knowledge Management Platform*, a RNRT project for cartography of skills in telecommunications for Sophia Antipolis firms. The KMP ontology comprises 542 concepts and 45 relations. Corese answers queries such as: *"Who are the possible industrial partner knowing how to design integrated circuits within the GSM field for cellular/mobile phone manufacturers?"*.
- *Ligne-de-Vie*: A virtual staff for a health network relies on an ontology comprising 26432 concepts and 4 relations. It guides several physicians for discussing among alternative therapies for a given pathology, according to the patient's features.
- *MEAT*: a memory of experiments of biologists on DNA microarray relies on annotations on scientific articles, using UMLS as an ontology. Corese can answer queries such as *"Find all the articles asserting that HGF gene plays a role in lung disease"*.

Corese has also been tested with existing RDF Schemas such as the Gene ontology (13700 concepts, 950000 relations), IEEE LOOM, W3C CC/PP, etc. Let us illustrate the expressivity of the Corese query language by giving a concrete example of approximate retrieval taken from the experiments of Corese with the O'CoMMA ontology.

The query below:

```
select more where
?p rdf:type c:Person
?p c:HasForActivity ?a
?a rdf:type c:KnowledgeDissemination
```

returns the following approximate answer:

[4] `http://wwww.inria.fr/acacia/corese`

```
[Employee, Researcher http://www.inria.fr/a.g]
   (gDistance)      [Literal 0.0078125]
   (HasForActivity) [Research]
   (HasForActivity) [Education]
   (IsInterestedBy) [KnowledgeEngineeringTopic]
   (IsInterestedBy) [CognitiveSciencesTopic]
```

Corese palliates the absence of exact retrieval by using the `rdfs:seeAlso` link between properties - it thus considers the *IsInterestedBy* property instead of *HasForActivity* property asked in the query - and between classes - it thus considers *KnowledgeEngineeringTopic* and *CognitiveSciencesTopic* instead of *KnowledgeDissemination*.

4.3 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. [11] details the scenario-based approach used for evaluating several Corese-based applications (in particular, CoMMA and KMP).

Moreover Corese achieves good performances, due to its very efficient projection operator. For example, with an RDF graph of 18000 relations, Corese answers in 0,01s in exact mode (0 answer) and in 0.02s in approximate mode (5 answers) to the following query, where the $\sim$ operator means *contains*.

```
?doc rdf:type c:TechnicalReport
?doc c:Designation ?t
?t ~ knowledge
?doc c:CreatedBy ?p
?p rdf:type c:Person
?p c:Designation ?x
```

In approximate mode, it finds an interesting answer which is a *Lecture* organized by a *GatheringEntity*, instead of a *TechnicalReport* created by a *Person*. Such examples give an idea of the power of the approximate search mechanism of Corese.

5 CONCLUSIONS

In this paper we presented an ontology-based IR system, Corese, and its query language for searching the semantic web, possibly with approximate queries. A semantic distance between classes and properties of the ontology enables to sort the approximate answers by relevance according to the ontology. Approximate answers can also be retrieved by using the `rdfs:subClassOf` and `rdfs:seeAlso` properties when matching a query with annotations and by querying for variable length relation paths between concepts.

Corese can be compared to query languages or tools dedicated to RDF such as RQL [14], Triple [22], SquishQL [18], Sesame[2], KIM [15], or the tools described in [7, 27]. But, to our knowledge, *Corese is the only RDF(S)-dedicated engine that offers both inference rules and approximate search*.

As a further work, we are studying Corese extension to handle annotations represented in OWL [25]. Our choice of RDFS is mainly historical since the first implementations of Corese preceded the emergence of OWL. However, in most applications of Corese, the expressivity of the RDFS language is sufficient - if extended with inference rules and approximation in the query language. Our previous work on Description Logics and CGs [5] [6] convinces us that Corese should and could handle the OWL Lite features.

We also currently explore user profile features for integration into Corese query processing.

ACKNOWLEDGEMENTS

We thank Olivier Savoie for his implementation work on Corese, the Acacia team for research and applications around Corese and the INRIA for the support of Corese development.

REFERENCES

[1] T. Berners-Lee, J. Handler, O. Lassila. The Semantic Web, Scientific American, May, 2001.
[2] J. Broekstra et al. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In ISWC'2002, pp. 54-68, Sardinia, Italy, 2002
[3] O. Corby et al. A conceptual graph model for W3C Resource Description Framework. In Proc. of ICCS'00, pp. 468-482, 2000.
[4] O. Corby, C. Faron-Zucker. Corese: A Corporate Semantic Web Engine, In Proc. of the WWW'2002 Workshop on Real World RDF and Semantic Web Applications, Honolulu, Hawai, USA, 2002.
[5] P. Coupey, C. Faron. Towards Correspondences between Conceptual Graphs and Description Logics. In Proc. of ICCS'98, LNCS 1453, Springer Verlag, pp. 165-178, Montpellier, France, August 1998.
[6] A. Delteil, C. Faron. A Graph-Based Knowledge Representation Language. In Proc. of ECAI'2002, pp. 297-301, Lyon, France, 2002.
[7] A. Eberhart. Automatic Generation of Java/SQL Based Inference Engines from RDF Schema and RuleML. ISWC'2002,pp. 102-116, 2002.
[8] D. Fensel et al. On2broker. Semantic-Based Access to Information Sources at the WWW. In Proc. of Webnet'99, pp. 366-371, 1999.
[9] F. Gandon et al. Semantic Web and Multi-Agents Approach to Corporate Memory Management, 17th IFIP World Comp. Congr., p. 103-115, 2002.
[10] F. Gandon. DAI and KM: ontologies and MAS for a corporate semantic web. PhD, 2002, UNSA.
[11] A. Giboin et al, Assessment of Ontology-based Tools: Systemizing the Scenario Approach, Proc. of EON2002, Siguenza, Sept. 2002, pp. 63-73
[12] J. Golebiowska et al, Building and Exploiting Ontologies for an Automobile Project Memory, Proc. of K-CAP, Victoria, October 23-24, 2001.
[13] N. Guarino et al. Ontoseek: Content-based access to the Web. In IEEE Intelligent Systems, vol. 14(3), pp. 70-80, 1999.
[14] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl. RQL: a declarative query language for RDF. In Proc. of WWW'2002, pp. 592-603, Honolulu, Hawaii, USA, 2002.
[15] A. Kiryakov et al, Semantic Annotation, Indexing and Retrieval. In Proc. of ISWC'2003, pp. 484-499, Florida, Oct. 2003
[16] S. Luke et al. Ontology-based Web agents. In Proc. Of the 1st Int. Conf. On Autonomous Agents, 1997.
[17] P. Martin, P. Eklund. Knowledge Retrieval and the World Wide Web. IEEE Intelligent Systems, 15(3):18-25, 2000
[18] L. Miller et al. Three Implementations of SquishQL, a Simple RDF Query Language. In ISWC'2002, 2002
[19] K. Patel, G. Gupta. Semantic Processing of the Semantic Web. In Proc. of ISWC'2003, LNCS 2870, pp. 80-95, Sanibel Island, Florida, USA, 2003.
[20] C.J. Rijsbergen. A new theoretical framework for information retrieval, In Proc. of the ACM Conf. on Research and Dev. in IR, pp. 194-200, 1986.
[21] E. Salvat. Theorem Proving Using Graph Operations in the Conceptual Graph Formalism, In Proc. of ECAI'98, pp. 356-360, Brighton, UK, 1998.
[22] M. Sintek, S. Decker. Triple: A Query, Inference and Transformation Language for the Semantic Web. ISWC'2002, pp. 364-378, Sardinia, 2002
[23] F. Southey and J. G. Linders, Notio - A Java API for Developing CG Tools, 7th ICCS, pp 262-271, 1999,
[24] J.F. Sowa. Conceptual structures: Information Processing in Mind and Machine. Addison-Wesley, Reading, Massachusetts, 1984.
[25] W3C. Web Ontology Language. `http://www.w3.org/sw/WebOnt`.
[26] W3C. Resource Description Framework, `http://www.w3.org/RDF`.
[27] J. Wielemaker, G. Schreiber, B. Wielinga. Prolog-Based Infrastructure for RDF: Scalability and Performance. ISWC'2003, pp. 644-658, 2003
[28] J. Zhong et al, Conceptual Graph Matching for Semantic Search, ICCS'2002, pp. 92-106, Borovets, July 2002.