ISSN:0975-9646

Falcons Concept Search: A Practical Exploring Engine for Net Ontologies

P.Sreedhar Reddy, K.Morarjee

Department of CSE, CMRIT, Dist: R.R,
A.P, India

Abstract—Reuse of knowledge bases and the semantic web are two promising areas in knowledge technologies. Given some user requirements, finding the suitable ontologies is an important task in both these areas. This paper discusses our work on Onto Search, a kind of "ontology Google", which can help users find ontologies on the Internet. Onto Search combines Google Web APIs with a hierarchy visualization technique. It allows the user to perform keyword searches on certain types of "ontology" files, and to visually inspect the files to check their relevance. Onto Search system is based on Java, JSP, Jena and JBoss technologies.

Keywords— Indexing, Ontologies ranking, Ontologies search, snippet generation, virtual document.

I. INTRODUCTION

The Semantic Web is targeted at facilitating data integration across Web applications[1]. Semantic Web data are formatted according to Resource Description Framework (RDF), a triple/graph-based way to represent information. Furthermore, Web Ontologies described in RDF Vocabulary Description Language (RDFS) and the Web Ontology Language (OWL) provide shared Concepts, i.e., classes and properties, for describing domain entities and thus enabling semantic interoperability of different applications. Semantic interoperability depends on reusing or extending existing Ontologies when developing new applications. Therefore, Ontologies search becomes a fundamental service for application developers.

In recent years, several Ontologies search engines have been developed; some of which are still accessible [1]–[3]. Similar to traditional Web search engines, these systems accept keyword queries and re-turn matched Concept s and/or Ontologies. However, for the returned results, they usually provide either only basic metadata (e.g., a human-readable name of each Concept) or all the related RDF description, both of which cannot help users efficiently determine whether a Concept /ontology returned satisfies their needs.

We developed Falcons Concept Search,1 a novel keyword-based Ontologies search engine, as part of the Falcons system. It retrieves Concept s whose textual description is matched with the terms in the keyword query and ranks the results according to both query relevance and popularity of Concepts. The popularity is measured based on a large data set collected from the real Semantic Web[2]. Each Concept returned is associated with a query-relevant structured snippet, indicating how the Concept is matched with the keyword query and also briefly clarifying its meaning. Meanwhile, the system recommends several query-relevant popular Ontologies, which can be used by users to restrict the results to the ones in a specific ontology. Within such a mode of interaction, users can quickly compare Ontologies and deter-mine whether these Ontologies satisfy their needs by checking query-relevant Concept s as well as their contexts, i.e., structured snippets.

The system also provides the detailed RDF description of each Concept and a summary of each ontology on demand. A demonstration of the system is given in the following.

A Panoramic Approach to Integrated Evaluation of Ontologies in the Semantic Web

As the sheer volume of new knowledge increases, there is a need to find effective ways to convey and correlate emerging knowledge in machine-readable form. The success of the Semantic Web hinges on the ability to formalize distributed knowledge in terms of a varied set of ontologies. We present Pan-Onto-Eval, a comprehensive approach to evaluating an ontology by considering its structure, semantics, and domain. We provide formal definitions of the individual metrics that constitute Pan-Onto-Eval, and synthesize them into an integrated metric. We illustrate its effectiveness by presenting an example based on multiple ontologies for a University.

Ontology Ranking based on the Analysis of Concept Structures

In view of the need to provide tools to facilitate the re-use of existing knowledge structures such as ontologies, we present in this paper a system, AKTiveRank, for the ranking of ontologies. AKTiveRank uses as input the search terms provided by a knowledge engineer and, using the output of an ontology search engine, ranks the ontologies. We apply a number of classical met-rics in an attempt to investigate their appropriateness for ranking ontologies, and compare the results with a questionnaire-based human study. Our results show that AKTiveRank will have great utility although there is potential for improvement

Constructing Virtual Documents for Ontology Matching

On the investigation of linguistic techniques used in ontology matching, we propose a new idea of virtual documents to pursue a cost-effective approach to linguistic matching in this paper. Basically, as a collection of weighted words, the virtual document of a URI ref declared in an ontology contains not only the local descriptions but also the neighbouring information to recent the intended meaning of the URI ref. Document similarity can be computed by traditional vector space techniques, and then be used in the similarity-based approaches to ontology matching. In particular, the RDF graph structure is exploited to define the description formulations and the neighbouring operations. Experimental results show that linguistic matching based on the virtual documents is dominant in average F-Measure as com-pared to other three approaches. It is also demonstrated by our experiments that the virtual documents approach is cost-effective as compared to other linguistic matching approaches.

II. System Architecture

Ontology building methodology

In this section, the methodology used to discover and select representative concepts and websites for a domain and construct the final ontology is described.

The algorithm is based on analysing a large number of web sites in order to find important concepts for a domain by studying the initial keyword's neighbourhood (we assume that words that are near to the specified keyword are closely related). The candidate concepts are processed in order to select the most adequate ones by performing a statistical analysis[16].The selected classes are finally incorporated to the ontology[17]. For each one, the main websites from where it was extracted are stored, and the process is repeated recursively in order to find new terms and build a hierarchy of concepts.

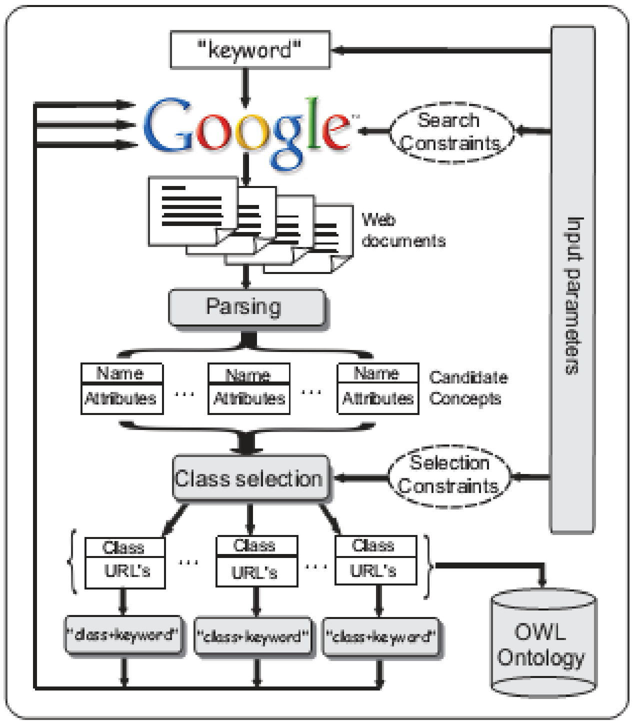

Figure 1: Ontology building algorithm

Semantic Web Frame Work

Information retrieval by searching information on the web is not a fresh idea but has different challenges when it is compared to general information retrieval. Different search engines return different search results due to the variation in indexing and search process. Google, Yahoo, and Bing have been out there which handles the queries after processing the keywords. They only search information given on the web page, recently, some research group's start delivering results from their semantics based search engines, and however most of them are in their initial stages.

Till none of the search engines come to close indexing the entire web content, much less the entire Internet.

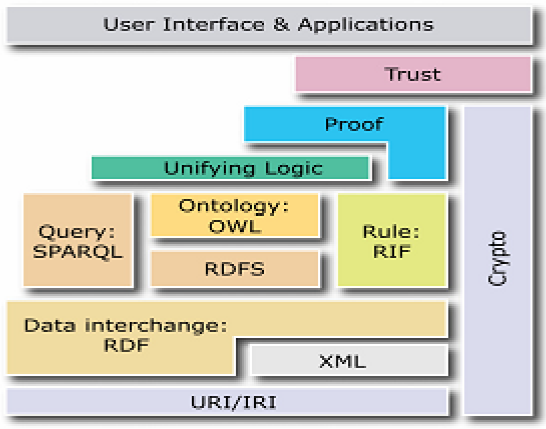

Fig.2: Semantic Web Frame Work

Watson Architecture

The role of a gateway to the Semantic Web is to provide an efficient access point to online ontologies and semantic data[2]. Therefore, such a gateway plays three main roles: 1- it collects the available semantic content on the Web, 2- analyses it to extract useful metadata and indexes, and 3- implements efficient query facilities to access the data. While these three tasks are generally at the basis of any classical Web search engine, their implementation is rather different when we deal with semantic content as opposed to Web pages.

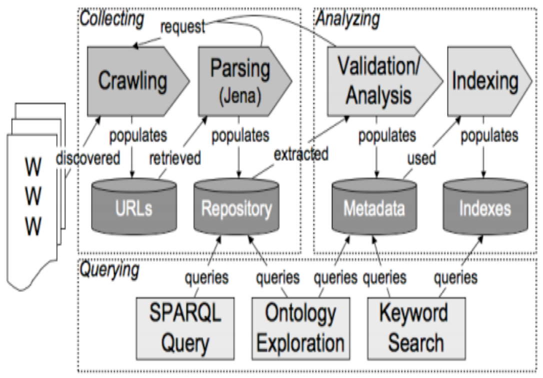

Figure 3: A functional overview of the main components of the Watson architecture.

At a more technical level, these three layers are hosted on a Web server (Apache Web server and Apache Tomcat), relying on a common RDMS (MySQL) to either communicate or exploit information about the collected semantic documents. All the components of Watson are written in Java. The descriptions of the three main tasks of Watson, both at the conceptual and technical levels, are presented in the following sections.

At searching time, popularity of Concept s and term-based similarity between virtual documents of Concept s and the keyword query are combined to rank Concept s. For each Concept returned, a query-relevant structured snippet is generated from the data in the quadruple store. Mean-while, several Ontologies are recommended (cf. Section IV-B) based on top-ranking Concept s. Moreover, for each Concept requested, the browsing Concept s component loads its RDF description from the quadruple store and presents it to the user. For each ontology re-quested, the browsing Ontologies component loads ontology metadata from the quadruple store, loads the lists of classes and properties contained from the metadata database, and presents all of them to the user.

III. Constructing Virtual Documents For Keyword Based Concept Search

Traditional Web search engines build an inverted index from terms in the contents of web pages to their URIs to serve keyword search. However, on the Semantic Web, a Concept has no such contents but is described by RDF triples, from which we need to extract terms to construct its virtual document [4]. Existing

AKTIVERANK

Figure 4 shows the current architecture of AKTiveRank. The main component (no. 2 in the figure) is a Java Servlet that receives an HTTP query from a user or an agent (no. 1). The query contains the terms to search for. Currently it is only possible to search for concepts. In other words, search terms will only be matched with ontology classes, and not with properties or comments.

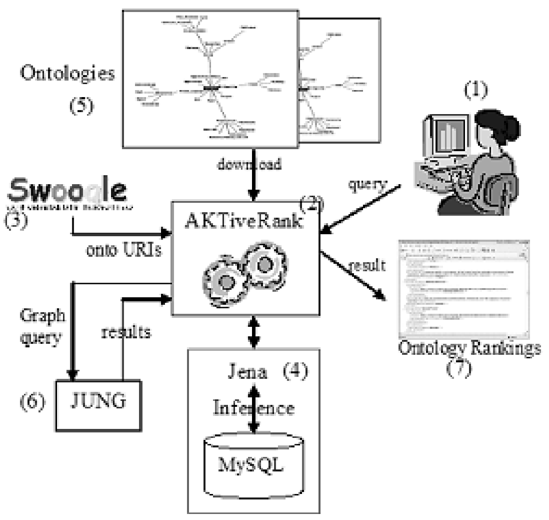

Figure 4: AKTiveRank Architecture

Existing RDF query languages are not well suited for graph queries [3]. To this end, the current version of AK-TiveRank is connected to a purpose-built JUNG servlet (no. 6), which receives an ontology URI and sends back re-sults of JUNG queries in RDF. JUNG (Java Universal Net-work/Graph framework) is a software library for analyzing and visualising network graphs.

AKTiveRank then analyses each of the ontology candi-dates to determine which is most relevant to the given search terms. This analysis will produce a ranking of the retrieved ontologies, and the results are returned to the user as an OWL file containing the ontology URIs and their total ranks.

RDF Sentence

An RDF graph G(T) can be mapped into a set of RDF statements T , composed of URI references, literals and blank nodes, making descriptions about resources. According to the RDF semantics, blank node is a kind of existentially quantied resources whose meaning is in the scope of the graph it occurs. RDF statements sharing a common blank node form a structure providing a joint context of the blank nodes. If such RDF statements are separated into different graphs, the context is broken. The structure is important to certain applications, which will reference or extract a sub-graph of an RDF graph and meanwhile require the extraction to retain meaningful. However, RDF semantics does not provide any intrinsic mechanism to identify this kind of structure.

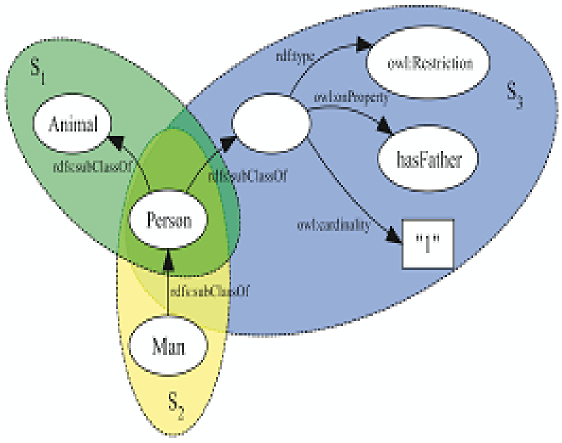

Figure 5: A graph representation of three RDF sentences derived from the Animal Ontology

Here, “term" refers to URI reference which is defined by users, not belonging to the build-in vocabulary of ontology language.

An example is shown in Figure 3, which is a sub graph of the RDF graph derived from the Animal Ontology. The graph can be divided into three RDF sentences: S1, S2 and S3.

IV. Ranking

AKTiveRank applies four types of assessments (measures) for each ontology to measure the rankings. Each ontology is examined separately. Once those measures are all calculated for an ontology, the resulting values will be merged to produce the total rank for the ontology.

In a previous version of AKTiveRank which was reported in [2], one of the measures applied was the Centrality Measure (CEM). That measure aimed to assess how representative a class is of an ontology based on the observation that the more central a class is in the hierarchy, the more likely it is for it to be well analysed and fully represented . How-ever, in some experiments we found a few ontologies that placed our concept of interest as a near-top-level concept. Those few ontologies were entirely focused around the concept we were searching for. This meant that even though such ontologies can be highly relevant to our search, they scored very low in CEM. Furthermore, we also found that CEM values corresponded in most cases to the values of the Density measure, and renders CEM somewhat redundant. The Density measure calculates the information content of a class.

This observation backs Rosch’s studies which showed that classes at mid-hierarchical levels tend to have more detail than others.

Class Match Measure

The Class Match Measure (CMM) is meant to evaluate the coverage of an ontology for the given search terms. AK-TiveRank looks for classes in each ontology that have labels matching a search term either exactly (class label identical to search term) or partially (class label "contains" the search term).

An ontology that contains all search terms will obviously score higher than others, and exact matches are regarded as better than partial matches. For example if searching for "Student" and "University", then an ontology with two classes labelled exactly as the search terms will score higher in this measure than another ontology which contains partially matching classes, e.g. "University Building" and "PhD-Student".

Density Measure

When searching for a "good" representation of a specific concept, one would expect to find a certain degree of detail in the representation of the knowledge concerning that concept. This may include how well the concept is further specified (the number of subclasses), the number of attributes associated with that concept, number of siblings, etc. All this is taken into account in the Density Measure (DEM). DEM is intended to approximate the representational-density or information-content of classes and consequently the level of knowledge detail.

Semantic Similarity Measure

Similarity measures have often been used in information retrieval systems to provide better ranking for query results. Ontologies can be viewed as semantic graphs of concepts and relations, and hence similarity measures can be applied to explore these conceptual graphs. Resin applied a similarity measure to WordNet to resolve ambiguities . The mea-sure he used is based on the comparison of shared features, which was first proposed in . Another common-feature based similarity is the shortest-path measure, introduced by Rada [12]. He argues that the more relationships objects have in common, the closer they will be in an ontology. Rada used this measure to help rank biomedical documents which were represented in a semantic knowledge-base. Variations of these techniques have been used to measure similarity between whole ontology structures.

Betweenness Measure

One of the algorithms that JUNG provides is Betweenness [7]. This algorithm calculates the number of shortest paths that pass through each node in the graph. Nodes that occur on many shortest paths between other nodes have higher betweenness value than others. The assumption is that if a class has a high betweenness value in an ontology then this class is central to that ontology.

Total Score

The total score of an ontology can be calculated once the four measures are applied to all the ontologies that the search engine returned. Total score is calculated by aggregating all the measures' values, taking into account the weight of each measure, which can be used to determine the relative importance of each measure for ranking.

The first rank will be given to the ontology with the highest overall score, the second rank to the second highest score, and so on.

V. Generating Snippets

For each Concept returned, the system presents a query-relevant structured snippet to show how the Concept is matched with the keyword query[15]. The snippet can help users quickly determine the relevance of a Concept to their needs. In this section, we propose a notion of property description thread (PD-thread) as the basic unit of a snippet and then introduce a method of ranking PD-thread and selecting the top-ranking ones into the snippet.

A. PD-Thread: The Basic Unit of a Concept Snippet

The description graph of a Concept is usually too large to be presented entirely so that a sub graph is extracted to form a snippet. How-ever, an RDF triple is not suitable for being the basic unit of a snippet

where_b1is a blank node, gives almost no useful information. Several graph structures [10]–[12] have been proposed to cope with a similar problem called RDF graph decomposition. However, presenting such structures having general topology may take significant space in the result page, which costs users much time to scroll up/down to read other Concept s before/after so that it reduces the efficiency of result checking and comparison.

Therefore, the basic unit of a snippet should be small sized but still meaningful when involving blank nodes. Based on this consideration, we propose a notion of PD-thread as the basic unit. In the description graph of a Concept c, a PD-thread of cis a path in the graph identified as follows: The starting node of the path isc, and the ending node of the path is not a blank node, i.e., a URI or a literal; the internal nodes of the path, if any, are all blank nodes and are distinct from each other.

The latter rule also avoids an infinite number of paths when there exist loops of blank nodes. For example, Fig. 4 contains four PD-threads ofswrc:Studentand one PD-thread of swrc:University. Evidently, a PD-thread, as its name indicates, is a linear structure so that it could be easily presented within one line. In most cases, a PD-thread is a single RDF triple not containing blank nodes. Otherwise, for each blank node contained in a PD-thread, two RDF triples are included to detail its denotation, i.e., in one as the object and in the other as the subject, which can better clarify the meaning than a single RDF triple can.

B. Generating Snippets by Ranking PD-Threads

In the system, a structured snippet of a Concept consists of at most three of its PD-threads. Thus, the problem of generating snippets is transformed into a new problem: ranking PD-threads according to

keyword queries. The ranking algorithm is outlined as follows:

1) Assign a ranking score to each PD-thread candidate.

2) Select the top-ranking candidate into the snippet.

3) If the desired number of PD-threads, which is three here, has not been reached, go back to Step 1). The ranking score of a PD-thread is evaluated by its relevance to the keyword query. A virtual document is constructed for each PD-thread in order to calculate its query relevance, which

includes the following: for each property labelled on the arcs of the path, use its local name and label, and for the ending node, use its local name and label if it is a URI or its lexical form if it is a literal. Finally, the ranking score of a PD-thread keyword query is defined as the cosine of the angle between the vector form of the virtual document of the PD-thread and the vector form of the keyword query.
Further, for a multitier query, the cosine measure may fail to create a snippet of a good coverage of the terms in the keyword query and may lead to a sort of redundancy. For example, for the keyword query "student university," the three selected PD-threads in a snippet may be all matched with "student," but none of them is matched with "university." To deal with this, inspired by previous work on text summarization [13], after a PD-thread is selected into the snippet, the weights of the terms in the vector form of the keyword query that occur in the virtual document of this PD-thread are set to a very small number, i.e., 0.001 in the system. Consequently, in the next rounds, other unmatched terms in the query will dominate the scoring of the remaining PD-thread candidates, and the generated snippet is likely to cover more terms in the query.

VI. EXPERIMENTS

In this section, we first present the results of a preliminary usability evaluation of Falcons Concept Search and then report feedback collected from participants after the experiments. A Usability Evaluation Other than Falcons Concept Search, Swoogle 2 [1], as one of the most famous Ontologies search engines, was also evaluated as reference.

Ontology Crawling and Aggregation

The execution of RDF crawler for 48 hours yielded considerable amount of data, detailed statistics provided in Table 2. Limited computational resources restricted the execution period of RDF crawler to 48 hours. Further, as the number of publicly available ontologies is limited, we consider the dataset of 418 ontologies as a good representative of the entire population.

Number of Relations Discovered	1321
Number of web pages visited	2018412
Number of Concepts crawled	19870
Total Ontologies(after Aggregation)	418

Table 1: OntoKhoj statistics

Ontology Classification

We have performed a series of experiments to determine the most suitable algorithms for the ontology classification. For this purpose, we selected four popular classification algorithms - Na¨ıve Bayes, TFIDF, KNN and PRIND. For the testing dataset, 22 ontologies were selected from five over-lapping domain of interests: Sports, Baseball, Soccer, Uni-versity, and Computer Science. The subject of our interest is the selection criterion, ontologies with a certain degree of overlapping domain were chosen. Each of the 22 ontologies was manually entered into the trained Rainbow Tool [2] to generate classification accuracy for each of the four classification algorithms.

Ontology Ranking

For the given classified ontologies, the OntoRank algorithm subsequently ranks them in descending order of their rank. For performing experiments with the proposed ranking algorithm, we obtained 10 ontologies in Tourism domain through OntoKhoj search interface. A subsequent execution of the algorithm on the dataset yielded results - a ranking of 10 tourism ontologies . A subjective evaluation of the results confirmed the correctness of the OntoRank algorithm. We admit that the subjective interpretation of our results is limited. Our research in the similar direction focused on the development of metric-based ontology ranking method considering the preferences of users. In future, we would like to incorporate a dynamic approach, wherein agents or users would express their own preference through the OntoKhoj portal for raking ontologies. We foresee that user oriented mechanism for ontology ranking would help in improving the practical accuracy of the results.

VII. RELATEDWORK: A COMPARISON OF ONTOLOGYSEARCHENGINES

There is no comparable published work on ontology summarization. But summarizing ontology has been used in some reasoning tasks. Fokouel et al. [9] proposed an approach to summarize Abox in secondary storage by reducing redundancy to make reasoning scalable for very large Aboxes. It is an alternative approach with KAON2 [14], which reduces an SHIQ(D) ontology to a disjunctive data log program and makes it naturally applicable to A boxes stored in deductive databases.

	Animal	Beer	CV
CI	0.626	0.651	0.403
CB	0.415	0.387	0.293
Cp	0.650	0.691	0.500
CH	0.598	0.601	0.492
CF	0.573	0.621	0.310

Table 2: Evaluation of summarization quality measured by vocabulary overlap

	Animal	Beer	CV
Without re-ranking	0.496	0.657	0.251
With re-ranking	0.650	0.691	0.500

Table 3: Evaluation of re-ranking measured by vocabulary overlap (C_P)

The notion of RDF sentence is originated from , where triples with blank nodes are divided into groups according to an equivalence closure for constructing knowledge base from RDF graph. A similar notion is stated in which is called self Minimum Self-contained Graph providing a unit of RDF graph for signing; meanwhile, a notion of RDF molecule is proposed in [4], which is a trackable unit providing prove-nance information. we give an early description of the RDF sentence, and use it as an indication of the depen-dency between domain entities within ontology.

VIII. CONCLUSION

In this paper, we have introduced how to search Concept s and Ontologies with Falcons Concept Search and have detailed its design and implementation. The system integrates Concept -level search and ontology-level search by recommending Ontologies and allowing filtering Concept s with Ontologies. For each Concept returned, its label, type, and a query-relevant structured snippet are provided to help users quickly determine its relevance to their needs. Based on the Concept s returned and their structured snippets, users can quickly learn the relevance and characteristics of an ontology and can also easily compare Ontologies. Detailed RDF description of Concept s and Ontologies has been also provided on demand. The technical contributions of this paper include a mode of inter-action that helps users quickly find desired Concept s and Ontologies as well as a supportive combined inverted index structure, a method of constructing virtual documents of Concept s that includes the names of associated properties and related entities, a way to rank Concept s and Ontologies based on their popularity on the Semantic Web as well as their relevance to keyword queries, and a method of generating query-relevant structured snippets. User interaction is crucial to the usability of a search engine. In future work, we will investigate other query types besides keywords, e.g., controlled natural anguages [14], and improve the method of snippet generation in order to better present ontology structures.

IX. FUTURE ENHANCEMENTS

In future work, It will investigate other query types besides keywords, e.g., controlled natural languages, and improve the method of snippet generation in order to better present ontology structures. It is also interesting to consider other metrics for ontology evaluation and recommendation.

REFERENCES

[1] L. Ding, T. Finin, A. Joshi, Y. Peng, R. Pan, and P. Kolari, "*Search on the semantic web,*" IEEE Comput., vol. 38, no. 10, pp. 62–69, Oct. 2005.

[2] M. d'Aquin, C. Baldassarre, L. Gridinoc, M. Sabou, S. Angeletou, and E. Motta, "*Watson: Supporting next generation semantic web applica-tions,*" inProc. IADIS Int. Conf. WWW/Internet, 2007, pp. 363–371.

[3] C. Anutariya, R. Ungrangsi, and V. Wuwongse, "*SQORE: A framework for semantic query based ontology retrieval,*" in Proc. 12th Int. Conf. Database Syst. Adv. Appl., 2007, pp. 924–929.

[4] C. Watters, "*Information retrieval and the virtual document,*"J. Amer. Soc. Inf. Sci., vol. 50, no. 11, pp. 1028–1029, Aug. 1999.

[5] Nokia, P. Stickler, CBD—Concise Bounded Description. [Online]. Avail-able: http://sw.nokia.com/uriqa/CBD.html

[6] Y. Qu, W. Hu, and G. Cheng, "*Constructing virtual documents for ontology matching,*" inProc. 15th Int. World Wide Web Conf., 2006, pp. 23–31.

[7] X. Zhang, H. Li, and Y. Qu, "*Finding important vocabulary within ontol-ogy,*" inProc. 1st Asian Semant. Web Conf., 2006, pp. 106–112.

[8] G. Wu, J. Li, L. Feng, and K. Wang, "*Identifying potentially important Concept s and relations in an ontology,*" inProc. 7th Int. Semant. Web Conf., 2008, pp. 33–49.

[9] H. Alani and C. Brewster, "*Metrics for ranking Ontologies,*" inProc. 4th Int. EON Workshop, 2006, pp. 1–7.

[10] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost,Y. Peng, P. Reddivari, V. C. Doshi, and J. Sachs.*"Swoogle: A semantic web search and metadataengine*". In Proc. 13th ACM Conf. on Information and Knowledge Management, Nov. 2004.

[11] Doan, A., Madhavan, J., Dhamankar, R., omingos, P., and Halevy, A.Y. *" Learning to Match Ontologies on the Semantic Web"*. VLDB J. 12(4) (2003), 303-319.

[12] Watters, C. "*Information Retrieval and the Virtual Document*". JASIS 50(11) (1999), 1028-1029

[13] Cheng Gong & Qu Yuzhong, "*Searching Linked Objects with Falcons: Approach, Implementation and Evaluation. In International Journal on Semantic Web and Information System*", 5(3), 50-71, July-September 2009

[14] H. Alani and C. Brewster. "*Ontology Ranking based on the Analysis of Concept Structures*". International Conference on Knowledge Capture 2005.

[15] X. Zhang, G. Chang & Y. Qu . "*Ontology Summarization based on RDF Sentence Graph*". In C. Williamson, M. E. Zurko, P. Patel-Schneider, & P.Shenoy (Eds.), Proceedings of the 16th International Conference on World Wide Web (pp. 707-716). New York,NY,USA: ACM ,2007.

[16] O. Ansa, E. Hovy, E. Aguirre, D. Martínez, "*Enriching very large ontologies using the.WWW*", In proceedings of the Workshop on Ontology Construction of the EuropeanConference of AI (ECAI-00),2000.

[17] A. Aldea, R. Bañares-Alcántara, J. Bocio, J. Gramajo, D. Isern,J. Jiménez, A.Kokossis, A. Moreno, and D. Riaño,"*n ontology-based knowledge managementplatform*",Workshop on Information Integration on the Web (IIWEB'03) atIJCAI'03, 177-182,2003.