An approximation approach for semantic queries of naïve users by a new query language

Ala Djeddai, Hassina Seridi-Bouchelaghem and Med Tarek Khadir

LABGED Laboratory, University Badji Mokhtar Annaba, Po-Box 12, 23000, Algeria

{djeddai, seridi, khadir}@labged.net

Abstract. This paper focuses on querying semi structured data such as RDF data, using a proposed query language for the non-expert user, in the context of a lack knowledge structure. This language is inspired from the semantic regular path queries. The problem appears when the user specifies concepts that are not in the structure, as approximation approaches, operations based on query modifications and concepts hierarchies only are not able to find valuable solutions. Indeed, these approaches discard concepts that may have common meaning, therefore for a better approximation; the approach must better understand the user in order to obtain relevant answers. Starting from this, an approximation approach using a new query language, based on similarity meaning obtained from WordNet is proposed. A new similarity measure is then defined and calculated from the concepts synonyms in WordNet, the measure is then used in every step of the approach for helping to find relations between graph nodes and user concepts. The new proposed similarity can be used for enhancing the previous approximate approaches. The approach starts by constructing a graph pattern (GP) from the query and finalized by outputting a set of approximate graph patterns containing the results ranked in decreasing order of the approximation value level.

Keywords. Graph matching, RDF, Naïve user, Graph pattern, Semantic Queries, Regular Path Queries, Approximation, Similarity, Ranking and WordNet

1 Introduction

In recent years, the amount of information on the web grows increasingly and the classic information retrieval is not able to find the answer which satisfies the user queries, therefore, the semantic search may be a proposed solution for such situations. Most users have not much knowledge about the querying language in the semantic web, they are not aware of target knowledge base; so the user query does not match necessary the data structure. It is very hard and difficult to understand intend of naïve users.

In this paper we propose an approach for answering a new query language inspired from the conjunctive regular path queries [1], the user query is transformed to a graph pattern. We use a new method to calculate the approximation level between the paths of the graph data and the query paths; approximation is enhanced using the

WordNet database so the method is based on a proposed meaning similarity between concepts from WordNet

We consider the problem of querying the semi-structured data such RDF data which is modeled by a graph $G = (V, E)$ and an ontology $O = (Vo, Eo)$. Where each node in V is labeled with a constant and each edge e is labeled with a label drawn from a finite set of symbols S, V contains nodes representing entity classes or instances or data values (values of properties), the blank nodes are not considered, the edges between the class nodes and the instance nodes is labeled by 'type', E represents the relations between the nodes in V, $V \subset Vo$ and $E \subset Eo$.

Users specify their request by a proposed language inspired from the conjunctive regular path queries CRP which have the next format:

$$Q : q(X1 \dots Xn) :- (Y1\ R1\ Z1), \dots, (Yn\ Rn\ Zn) \quad (1)$$

- Each Yi or Zi is a variable or a constant. The variable is specified by? , we make a simple modification to the constants for specifying the choices so the user is able to specify constants which are not necessarily appearing in G and he is able to use many constants by using the symbol '|'so Yi or Zi is a variable or a constant or expression (in our approach).
- Regular path expressions {R1,…,Rn}, which are defined by the grammar:

$$R: = \text{å} \mid a \mid \_ \mid L \mid (R1.R2) \mid (R1|R2) \mid R*, \quad (2)$$

Where ε is the empty string, "a" is a label constant, "_" denotes any label and L is a label variable.

- $X1 \dots Xn$ are head variables and the result is returned in these variables.

In this paper, for helping the naïve users, we propose a new simple query language, we focus on the regular expression which has a simple format (using only the '.' and the '|'), the query $Q1$ is an example of the proposed language, We construct from the user query a graph patterns GP for finding a set of sub graphs in G (approximate graph patterns) whose nodes matches the nodes in GP and its paths have a level of approximation to the paths in GP.

Example1. We assume that a user writes a query $Q1$ for finding the publications and the authors in 'California' university or 'Harvard' university in the 'ESWC 2012'conference:

$(?pub, ?author) :-\ (?pub, writer, ?author),$
$(?author, his\_univ.name|location, California|Harvard),$
$(?pub, conf.name, ESWC2012).$

Figure 1 shows a GP constructed from $Q1$, the separate points between symbols represented by non-labeled nodes, the query paths 1 2 3 correspond to user paths 1 2 3 of $Q1$. The variable nodes are specified with '?' to indicate that only these nodes are shown in the answer. In our work, the answers for the query is a set of approximated graph patterns ranked in order of decreasing the approximation level value, every one contains nodes that correspond to the user variables, the paths in every approximate graph pattern are an approximation of the paths in GP (every path in GP is corresponding to a single conjuncts query [4]). We use the graph patterns as answers, for

giving to the user the ability to explore the results for more information about the result nodes.

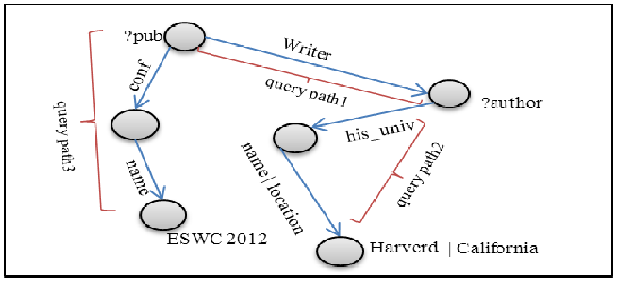

Fig.1. A graph pattern GP constructed from Q1

In section 2 related works are discussed and Section 3 presents WordNet and the new proposed similarity meaning. In section 4 the approximation approach is detailed. Section 5 is dedicated to the approach implementation and experimentation, whereas the conclusion and future works are presented in section 6.

2 Related works

Many approaches, methods and query language are proposed for the search in the semantic web search, and may be classified as follows:

1. Approaches consider structured query languages, such as: Corese [9], Swoogle [11] and ONTOSEARCH2 [4].

2. Approaches for naive users, these approaches can themselves be divided into:

- ✓ Keyword-based approaches, such as QUICK [8], where queries consist of lists of keywords;
- ✓ Natural-language approaches, where users can express queries using natural language, such that PowerAqua [10].

In this work we are interested by using the regular path queries with simple regular expression, this helps the naive users to use the query language as they are able to write simple regular expression. Our approach combines the two previously cited classes, so the naïve user queries the system using simple structure and the user constants are seen as keywords. Many works are proposed for the approximation such as [1] and [2], where the approximation is applied to the conjuncts queries. The ISPARQL [3] is a similarity based approach which added the notion of similarity to SPARQL queries, where another technique in [7] calculates the approximate answer from RDF graph using an evolutionary algorithm. Despite their efficiency, the approaches discard the user influence and opinion. The obtained results do not, therefore, often satisfy the latter. In addition to the above approaches, our work propose a new query language inspired from conjunctive queries, using a technique for the approximation based on meaning similarity from WordNet for a better understanding of the user query as well as finding the correspondences between its concepts and the graph data. The answers are a set of approximate graph patterns ranked in decreasing

order approximation level so the user can explore these results in order to acquire more knowledge.

3 Using WordNet

WordNet [5] is a lexical resource for the English language; it groups terms (name, verbs, adjectives etc.) in sets of synonyms called Synsets. Approaches based on characters strings become insufficient when concepts are systematically close to each other and when their names are different (example: « car » and « automobile »), the interrogation of a linguistic resource such as WordNet may indicates that two concepts are similar . For the calculation of the linguistic similarity, the function Syn(c) calculates the set of WordNet Synsets related to the concept c.

3.1 Definition of a new WordNet Meaning Similarity

In this section we define a new WordNet meaning similarity, this measure is used in the process of discovering the nodes mapping from the user query and graph data.
Let $S\_com = Syn(c1) \cap Syn(c2)$ the set of common senses between $c1$ and $c2$ to be compared, the cardinality of S_com is : ë $(S\_com) = |Syn(c1) \cap Syn(c2)|$, we use the following measure:

Let min(|Syn(c1)|, | Syn(c2)|) be the minimum cardinality between the two sets Syn(c1) and Syn(c2) for the concept c1 and c2 respectively, thus our similarity measure is constructed from analyzing of the next metric [7]:

$$\text{Sim1}(c1, c2) = \frac{\lambda\,(\text{S\_com})}{\min(|\text{Syn}(c1)|, |\text{Syn}(c2)|)} \quad (3)$$

This metric based on common senses of c1 and c2, it return 1.0 if c1 is synonym of c2 but if the set of senses for c1 (or c2) are including in the set of senses of c2 (or c1) so this metric return again 1.0, for example the concept "machine" has 8 senses and "motorcar" have 1 sense (included in the 8 sense of "machine"), utilizing this metric: $Sim1(machine, motocar) = \frac{1}{min(8,1)} = \frac{1}{1} = 1$, so "machine" is the synonym of "motorcar" but this is wrong because "machine" is the generalization of "motorcar", so from this idea we propose the next new measure which is based on the different senses between two concepts:
Let $S\_dif = ((Syn(c1) \cup Syn(c2)) - (Syn(c1) \cap Syn(c2))$: the set of different senses between c1 and c2, so $S\_Dif = union - intersection$. $|S\_dif|$ =|(Syn(c1) ∪ Syn(c2)) | – |(Syn(c1) ∩ Syn(c2))|, the set of union is defined as: $U = Syn(c1) \cup Syn(c2)$, our metric is:

$$Sim2(c1, c2) = 1 - \frac{|different\ senses|}{|union|} = 1 - \frac{|union| - |intersection|}{|union|} \quad (4)$$

$$Sim2(c1, c2) = 1 - \frac{|S\_dif|}{|U|} = 1 - \frac{|U| - |S\_com|}{|U|} \quad (5)$$

If $Syn(c1) = Syn(c2)$ (no different senses c1 is synonym of c2) then $Sim2(c1, c2) = 1 - 0 = 1$. $Sim2(machine, motorcar) = 1 - \frac{7}{8} = 1 - 0.87 = 0.13$ (7 common senses).

In this paper we use the next measure which takes advantage of Sim1 (common senses) and Sim2 (deferent senses):

$$Sim\_senses(c1, c2) = ù1 * \text{Sim1} + ù2 * \text{Sim2} \quad (6)$$

where, $ù1$ and $ù2$ are the widths associated to Sim1 and Sim2 respectively, $ù1 = 0.5, ù2 = 0.5$ by default i.e. same importance, we adjust $ù1$ and $ù2$ according the preference of the user.

*Example 2.*Table 1 shows values of similarity for some pair of concepts. We cannot find a significant similarity between these concepts if we use a metric based on syntax only, the $Levenshtein$ similarity indicates that "house" and "mouse" are similar but this is wrong, this highlights the importance of the proposed measure as it is used to find relationships between terms of the semantic regular path queries and the nodes of the graph data.

Concept1	***Concept2***	***Sim1***	***Sim2***	***Sim_sense***	***Levenshtein***
Car	*Automobile*	*0.5*	*0.16*	*0.33*	*0.0*
Location	*Placement*	*0.33*	*0.16*	*0.245*	*0.22*
House	*mouse*	*0.0*	*0.0*	*0.0*	*0.86*

Table 1. Some similarity values calculated using Sim_sense and $Levenshtein$

4 Approximating the naïve user queries

We start by defining the problem that is: how to satisfy the user in case if he specifies concepts that do not exist in the graph data? This is a big difficulty, as the approximation is the solutions for finding results and approximating the user query. However, it must take into account the concept meaning, this is the goal of the new proposed query language and the meaning similarity. This helps to better understand the user and helps the discovery a set of concepts in the structure which are relevant to user concepts in order to begin the process of exploration and finding the responses for the variables.

The proposed approach may be divided in three steps:

1- Discovering nodes which correspond to discovering user concepts in GP.
2- Finding for every query path its approximate paths in the graph data.
3- Generation of the results which are a set of approximate graph patterns with its approximation level value, these graph patterns contain the nodes results corresponding to the projection of the user variables.

The procedure is based on the following objectives:

- ✓ Giving to ability to the naïve user to take advantage from the power of semantic search, in this case we let him specify his needs by writing simple regular paths.
- ✓ Understanding the naïve user query by finding relationships between the user paths and the knowledge base (RDF graph). Most user concepts do not appear in the structure, for this reason, we propose a new query language and a meaning similarity leading to a better understanding of the user needs on one hand and discovering the correspondences between the query concepts and the graph nodes on the other hand. The user, however, still plays an important role in the query answer paradigm.
- ✓ The outputted answer must be understandable for the user and it should be simple.

We make clear the procedures have been omitted, in the rest of the paper, because of pages limitation; we cannot describe the approach in detail so only the main steps are mentioned.

4.1 Mapping from Nodes in GP to Nodes in G

The mapping process is necessary to find the correspondences of the nodes in GP (variables and constants in the conjuncts query); these nodes are used for finding the set of the approximate paths in $\boldsymbol{G}$. Because the user have lack knowledge of the graph data structure so he is able to use concepts not necessarily appearing in the graph and the process of mapping is important for discovering the nodes matches these concepts using WordNet. In order to enhance the matching we use a similarity metrics based on syntax (characters strings) (like: Levenshtein, NGram, JaroWinkler) and our meaning similarity (using the WordNet ontology) for discovering the senses (the meaning) commons between the concepts.

Definition 1. Two concepts $c1 \in GP$, $c2 \in G$are similar if $Sim\_senses(c1, c2) > T$ (WordNet similarity), T is predefined threshold, if $Sim\_senses(c1, c2) = 0$ then we test $sim\_synt(c1, c2) > T$, the values of Sim_senses, Sim_synt and T is defined in [0,1].

Sim_senses and Sim_synt (any syntax similarity) use the labels of nodes and edges.
In the rest of the paper we use $sim(c1, c2)$ for the value returned by Sim_senses or Sim_synt.

For finding the sets of node mapping the procedure get_nd_map returns for every node $n_i^j \in GP(i \in \{1,2\}$ i.e. the first or the last node in the query path QP_j), the set $NdMap_i^j$contains the nodes in G which are similar to n_i^j using its label by the similarity based senses (or based syntax), in addition this procedure use a strategy for discovering another nodes in G from the first and last edge in the query path QP_j .

4.2 Computing the Approximate Paths

In this section we introduce the notion of approximation level between two paths and describe the method for calculating its valuesapx_lev, this section is for the computation of the approximate paths from G, the finale answers (approximate graph patterns) are calculated in the next sub section. This calculation is started after the generation of the set of nodes mapping $NdMap_i^j$ for every node $n_i^j \in QP_j$ .The procedure get_apx_paths take as input a query path QP_j and outputting the set of tuples answer tup_ans_j ,every tuple (V_1, V_2, p, apx_lev) containing two node : V_1 is first node in the approximate path p, V_2 is the last node and ap_lev is the value of the approximate level between QP_j and p, the sets of tuples answer are used for constructing the approximate graph patterns for GP.
We consider the next points in the calculation process of $\boldsymbol{apx\_lev}$ **:**

- The number of edges in p similar to the edges in QP_j (similarity ≠ 1), each similar edge in p is a non-direct substitution for its corresponding edge in QP_j so we added the value of substitution to apx_lev .

- The number of additional edges in p ($cost\_ad$), not appearing in QP_j , each additional edge in p is an insertion.
- We also take into account the two values: similarity value between the first node in p and the first in QP_j, similarity value between the last node in p and the last in QP_j.
- The order of edges in the query path QP_j for respecting the preference of the user.

Our approach considers common and similar edges, therefore common edges are not associated to a value of '0' but '1', as well as the similarity values for similar edges. Before starting the process of finding the approximate local answer, the procedure get_apx_paths generates the set of all paths $ApxPths_j$ from the two sets of nodes mapping $NdMap_1^j$ and $NdMap_2^j$.

Definition 2. Let p a path in G, QP a query path in GP , p is an approximate path for QP if the value of the approximation level between p and QP is higher than T_apx (predefined threshold of approximation), $T\_apx \in [0,1]$.

The procedure get_apx_path use the similarity obtained between two nodes v_1 and v_2,$v_1 \epsilon\ QP_j$, $v_2 \epsilon\ path\ p$ .If v_1 is labeled with more than one term by the symbol '|' so all terms are compared to the label of $v_2$and only one value of similarity is returned i.e. the MAXvalue.

Example 3. Figure 2 shows the computation of the approximation level apx_lev for the paths $P \in G$ and $P' \in G$ for the query path QP_3. For $QP_3 \in GP$: the first node n_1^3 is labeled with the variable '?pub' and has the set of nodes mapping $NdMap_1^3$ = {publication, pub1, pub2}, the last node is labeled with constant 'ESWC2012' and has the set of nodes mapping $NdMap_2^3$ = {ESWC2012, ISWC2012}.The similar edges by discontinued line, additional edges by double line, first and last nodes by the dark circle; the values of similarity between edges and nodes are in italic,). The common edges are represented by single continued line. In the path P, number of similar or common edges is 2 (with two values of similarity: 0.95, 1), $1 = sim(?pub, pub2) = 0.90$, $sim2 = sim(ESWC2012, ESWC2012) = 1$, the approximate level associated with P is:

$$apx\_lev = \left(\frac{\sum val_{similarity}}{nb\ val_{similarity}} + \left(1 - \frac{cost_{ad}}{p.length}\right) + sim1 + sim2\right)/4 \quad (7)$$

$$apx\_lev = \left(\frac{0.95+1}{2} + \left(1 - \frac{0}{2}\right) + 0.90 + 1\right)/4 = 0{,}97 \quad (8)$$

The tuple answer corresponding to P is: $(pub2, ESWC2012,\ P,\ 0{,}97)$.
In the path P' there is one additional edge the (the edge *type*) so $cost\_ad = 1$ and, number of similar or common edges is 2 (with two values of similarity: 0.95, 1), $sim1 = sim(?pub, publication) = 0.70$, $sim2 = sim(ESWC2012,\ ISWC2012) = 0.20$, so the approximate level associated with P' is :

$$apx\_lev = \left(\frac{0.95+1}{2} + \left(1 - \frac{1}{3}\right) + 0.70 + 0.20\right)/4 = 0{,}64 \quad (9)$$

The tuple answer corresponding to P is: $(pubublication,\ ISWC2012,\ P',\ 0{,}64)$.
apx_lev for P is greater than apx_lev for P'so the path P have a good approximation than P'.

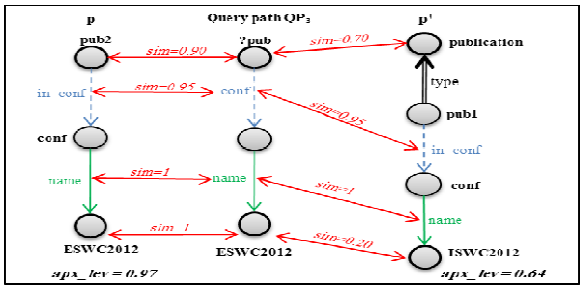

Fig. 2. Computing the approximation level apx_lev for the paths P and P′

4.3 Computing the Approximate Graph Patterns

In this section we describe how the final answers (Approximate graph patterns) are computed from the approximate paths discovered in the previous step, The final answers for the approximation is returned in form of tuples and every tuple represented by $(v_1, v_2, \ldots, v_n, apx\_gp, apx\_gp\_lev)$, where v_1 to v_n are the nodes corresponding to the nodes variable in GP (the nodes answers corresponding to the variables in the user query). apx_gp is the approximate graph pattern constructed from the approximate paths returned in tup_ans_j and apx_gp_lev is the approximation level between GP and $ApxGP$ i.e. the mean of the values apx_lev of the approximate paths used for the construction of apx_gp.

In [1] and [2], the final answers are a set of nodes corresponding to the variable in the query, in addition; as our approach based on graph patterns, a graph pattern with each nodes result is returned for a better answers understanding.

For computing the final answer we must generate the set of tuples answer tup_ans_j for every path $QP_j \in GP$, exploring the paths in every tuple and combining same paths for the generation of the graph patterns answer $\in G$.

Definition 3. Let GP a graph pattern constructed from a regular conjunctive query$Q$, Let GP' a graph pattern$\in G$. GP' is an approximate graph pattern for GP, if the value of approximate level apx_gp_lev between GP and GP' is higher than T_apx_gp (predefined threshold of approximation for GP), $T\_apx\_gp \in [0,1]$.

The procedure $final\_ans$ is called with the set tup_ans_1 and its first tuple. The procedure $final\_ans$ explores all tuples in any tup_ans_j to generate all approximate graph patterns, added them in the final set $final\_ans$ with its nodes variables and its approximation level. For the process of ranking, the value apx_gp_lev is used to rank the tuples in $final\_ans$, the tuples are outputted ranked in a decreasing order.

5 Implementation and Experimentation

Our approach is implemented in Java and Jena API, we use JAWS (Java API For WordNet Searching) for the implementation of the proposed meaning similarity. The RDF data set used is a sub set from the *SwetoDblp* ontology which is large size ontology focused on bibliography data of Computer Science publications where the main data source is DBLP, it has 4 millions of triples. The used subset contains a collection

of books and its book chapters. For making the execution faster, an offline phase which contains: RDF triples normalization, (getting triples that are closely to the natural language), building 2 indexes, is computed in order to allow quick finding of the approximate paths. The thresholds T, T_apx, T_apx_Gp are automatically initialized and updated according the query structure, this update allows the reduction of the found answers number.

For experimentation purposes and because our query language is inspired from the conjunctive path queries for helping the naïve (non-expert) users, a query benchmark is created. The benchmark contains a set of queries, with different intends that are executed over the RDF subset. For every query, from the subset; we computed, manually, the set of the relevant solutions (*RS*) for evaluating *Precision* and *Recall*:

$$Precision = \frac{\text{the number of relevant solutions returned (included in RS)}}{\text{number of solutions found by the system}} \quad (10)$$

$$Recall = \frac{\text{the number of relevant solutions returned(included in RS)}}{\text{the number of solutions in RS}} \quad (11)$$

In comparison with SPARQL, our work can be used by a non-expert users and it allows specifying a query paths between variables and constants for a better understanding of the user intend. It is difficult for the naïve user to use SPARQL efficiently because its complexes structure. Table 2 includes some queries, used for the evaluation whereas Figure 4 shows the precision and recall for some queries, proving the effectiveness of the approach.

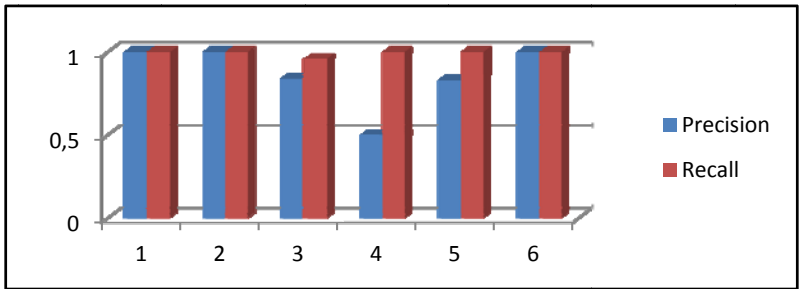

Fig. 3. Evaluation results for some queries

user query	Nb answers in RS	Nb system answers	Nb Relevant answers	**Precision**	**Recall**
(?Book_Chapter , title.contains, web) User intend: *Find all Book_chapters that have title contains «web »*	52	59	50	**0.84**	**0.96**
- (?Book_Chapter, book chapter included in the book, Prolog and Databases) - (?Book_Chapter, pages number, ? pages) User intend: *Find all Book_Chapters included in the book: «Prolog and Databases », associated with the pages number.*	20	20	20	**1.0**	**1.0**

- (?Book, year of publication, 2000) - (?Book, book isbn, ?isbn) - (?Book, has publisher, ?publisher) User intend: *Find Books published in 2000, associated with its isbn and the publisher.*	5	6	5	**0.83**	**1.0**

Table 2. Some user queries used for the evaluation

6 Conclusion and Future Works

In this paper a novel approach for query approximation based on meaning similarity from WordNet is proposed, using a proposed query language inspired from the conjuncts queries. Using this technique, the naive users are able to write simple queries that not necessarily match the data structure. Our approach can be used as an extension to other approaches for a better understanding of the user query and obtaining results that satisfies the user's needs. It has been shown that the answers are a set of graph patterns ranked following the approximation level decreasing order. The work, is not considering only RDF graph but it can be seen as a general approach which may be applied to any semi-structured data that is modeled as graph, Future work will consist in applying the proposed approach to specific domains such as geographic, medical, biologic and bibliography, using query interface and building new indexes for scaling a huge number of triples.

References

1. A. Poulovassilis and P. T. Wood Combining Approximation and Relaxation in Semantic web Path Queries. In Proc. ISWC, 2010.
2. C. A. Hurtado, A. Poulovassilis, and P. T. Wood. Ranking approximate answers to semantic web queries. In Proc. ESWC, pages 263–277, 2009.
3. C. Kiefer, A. Bernstein, and M. Stocker. The fundamentals of iSPARQL: A virtual triple approach for similarity-based semantic web tasks. In Proc. ISWC, pages 295–309, 2007.
4. E. Thomas, J. Z. Pan, and D. H. Sleeman. ONTOSEARCH2: Searching ontologies semantically. In Proc. OWLED-2007, CEUR Workshop Proceedings 258. CEUR-WS.org, 2007.
5. Eyal Oren, Christophe Guéret, Stefan Schlobach. Anytime Query Answering in RDF through Evolutionary Algorithms, International Semantic Web Conference pp.98-113, 2008.
6. Fellbaum, C.: WordNet, an electronic lexical database. MIT Press, Cambridge (1998)
7. Fellah, A., Malki, M and Zahaf, A., « Alignement des ontologies : utilisation de WordNet et une nouvelle mesure structurelle CORIA 2008 - Conférence en Recherche d'Information et Applications, Trégastel, France, 2008.
8. G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl. From keywords to semantic queries -Incremental query construction on the Semantic Web. J. Web Sem., 7(3):166–176, 2009.
9. O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker. Querying the Semantic Web with Corese search engine. In Proc. ECAI-2004, pp. 705–709. IOS Press, 2004.
10. Lopez, V., Fernndez, M., Motta, E., Stieler, N.: PowerAqua: Supporting Users in Querying and Exploring the Semantic Web Content. Semantic Web Journal. IOS Press (2011).
11. T. W. Finin, L. Ding, R. Pan, A. Joshi, P. Kolari, A. Java, and Y. Peng. Swoogle: Searching for knowledge on the Semantic Web. In Proc. AAAI-2005, pp. 1682–1683.