International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:2, 2008

PmSPARQL: Extended SPARQL for Multi-paradigm Path Extraction

Thabet Slimani, Boutheina Ben Yaghlane, and Khaled Mellouli,

Abstract—In the last few years, the Semantic Web gained scientific acceptance as a means of relationships identification in knowledge base, widely known by semantic association. Query about complex relationships between entities is a strong requirement for many applications in analytical domains. In bioinformatics for example, it is critical to extract exchanges between proteins. Currently, the widely known result of such queries is to provide paths between connected entities from data graph. However, they do not always give good results while facing the user need by the best association or a set of limited best association, because they only consider all existing paths but ignore the path evaluation. In this paper, we present an approach for supporting association discovery queries. Our proposal includes (i) a query language PmSPRQL which provides a multi-paradigm query expressions for association extraction and (ii) some quantification measures making easy the process of association ranking. The originality of our proposal is demonstrated by a performance evaluation of our approach on real world datasets.

Keywords—Association extraction, Query Language, relationships, knowledge base, multi-paradigm Query.

I. Introduction

INCREASINGLY the organizations converge towards the use of semantic Web technologies, in other word the choice depends on the adequacy of their needs with the support provided by these technologies. Consequently, the possibilities of the semantic requests and information storage become recommended to provide a uniform space of research allowing the mining and the flexible exploitation of semantic data. In the field of the semantic Web, current exploitation technologies, mainly concentrate on the entities, resources, and/or associations identification (interesting relations or path in RDF graph). The research techniques of path offer effective significance to answer requests of type ”is there a semantic relationship between entities A and B?”. The concept of relation or semantic association refers to a complex path which connects two entities contained in a knowledge base [1] [2] [3]. The origin of semantic associations was appeared with the theories and the methods coming from the research of LSDIS laboratory at the university of Georgia [1]. The semantic association discovery is a process of path extraction which connects two specified entities. The approaches and the methods which offer a language of association extraction and estimation of their qualities became increasingly recommended. Certainly, there are attempts to making possible an approach of associations extractions and ranking [4] [1] [2] [3] [5]. The association identification and ranking are a very interesting topic adapted in the context of the semantic Web. However, there is no remarkable profit of perspicacity about what gives good associations and in the manner of finding them effectively. Query languages, starting from RDF bases, such as RDQL [6] and its Successor SPARQL [7] support the definition of the graph models and allow restrictions on the entities and the properties which take part in certain definite models. Nevertheless, no support is provided by these languages with regard to the extraction of paths or associations in RDF graphs. Recently, two languages [8] and [9] who are interested in the extraction of the paths in RDF graph have been developed. These languages offer a support for path extraction, but they miss the evaluation process of the extracted paths. It is within this framework where our work is, which proposes a new PmSPARQL language which doesn't only make it possible to achieve the needs for semantic associations extraction, but also to provide, in some cases, the possibilities of arranging them to distinguish the most suitable association with the user's need. Moreover, we propose to provide a space of needs specification in a request in such a way that the result will not be presented exclusively in the form of the assertion ”entities A and B are connected through such and such a way”, but also by the extraction of the best way according to some specified dimensions.

The paper is organized as follows. In Section 2, we give a motivation for querying semantic association discovery in RDF graph. In Section 3, we introduce the related work and contribution. In Section 4, we present the specification of semantic association in RDF base. In Section 5, we propose the analysis of semantic association quality. Section 6, introduces our proposed PmSPARQL language, the concept of a path in PmSPARQL, shows the syntax and semantics of the language, and describes its algorithm for path identification. Finally, in section 7, we discuss the performance and the experimental results of our implementation.

II. Motivations

In spite of the quick change with regard to the development of the storage and extraction means in the field of semantic Web, there remains/stills a radical gap between the support provided by the developed mechanisms and the needs for certain categories of applications. In particular, the analytical and investigatory applications in some fields such as national security, economic intelligence, bioinformatics, scientific research, require the associations and interactions identification

T. Slimani is with the Department of Computer Science, High Institute of Management, Tunisia, CA 2000, Bardo Tunisia (Phone: 0021697498557; email:Thabet.slimani@gmail.com).

B.B. Yaghlane and K. Mellouli are with the Department of Computer Science, Institute of High Commercial Study, CA 2016, Carthage Precidency, Tunisia (emails: {boutheina.yaghlane,khaled.mellouli}@ihec.rnu.tn)

[1]http://lsdis.cs.uga.edu/

which exist between data. However, it is crucial to invent methods that make possible to answer questions of the type ”How two entities A and B are connected?”, ”what’s the best connection?” and ”there exists a fashion of presenting these connections by order of merit?”. Such request refers to the extraction of paths which connect two specified entities or which connect a single source entity with unknown target entity. Moreover, the developed language provides the possible requests which include multi-paradigm choice with regard to the nature of the associations wished by an extraction process. We present here, a definite real task of analysis which can adopt these types of requests:

Biological Science: some examples of semantic associations can be generated in the sequences of metabolic pathways. The metabolic pathways are composed by sequences of chemical reactions occurring in a cell. For example, the pathway N-Glycan Biosynthesis [10] is an example quite clear of the metabolic pathways. The mining of metabolic sequences is a difficult problem which requires a request of regular path. The results produced by these requests allow the researchers in the field of biology to extract some ordered sequences of specific reactions which can occur starting from a specified sequence in entry to the desired final product.

- Analysis of genes interaction implied in advanced ovarian cancer treated in [11]. The researchers will like to analyze these genes with regard to the biological pathways which they participate to facilitate the comprehension of the mechanism which leads to the disease. The identification of these interactions requires only a language which makes it possible to express specific constraints of concerned genes, for example expressed in epithelial cells.

*National Security:*To ensure the safety concerning the threats that can occur on certain flights. The type of requests which can be adopted in this task related to the identification of some relationships between terrorist peoples who purchased tickets according to a well specified period.

Scientific research: To evaluate the degree of connectivity of a specific researcher, such as its field of expertise and for example the degree of relevance of a paper subjected to a conference with the field of expertise of a reviewer. The request type which can be adopted in this case must have constraint to the level of the identification of the frequent properties which have a relationship to the resources related to the fields of research. Some works concerning the analysis of links to discovering social relationships in academic communities were presented in [12].

III. Related work and contributions

The topic of semantic associations can be regarded as a fundamental layer in various real-world applications. This topic approached in various sectors; especially, in the national security sector [13], the extraction of biomedical acts [14], the geospatial semantic analysis [15],...Also, the extraction of the semantic relationships was presented in the work of [16] which implies the discovery of new paths between entities, new loops and nodes significantly connected. As an attempt at efficient presentation of an association to a user, the semantic associations ranking was also presented in the work of [4] [1] [2] [3] [5], as well as of the effective algorithms concentrating on the execution and the effectiveness were presented in work of [17] [18]. The credibility measures of semantic associations starting from multiple sources were proposed in the work of ([19] [20]).

Works presented earlier does not offer a flexible model of association extraction which is applicable for various fields. We are interested by paths extraction languages. In the literature, several query languages designed for semi-structured databases in order to support the definition of regular paths requests. Among these languages, we can state the language G [21], G+ [22] and Graphlog [23]. In the field of semantic Web, the paths extraction starting from RDF graphs is treated by *SQL-Like* SerQL languages [24], SPARQL, RDQL. *Rule-based* languages such as TRIPLE [25], N3 and traversal graph languages such as Versa [26] and PxPath [27]. These languages offer a partial support for the paths extraction, but do not include possibilities of regular paths extraction. Recently, an important SPARQeL language [9] which represents the extension of SPARQL language has been developed specifically for semantic association extraction starting from RDF graph. This language incorporates paths models making it possible to capture the needs of semantic and structural association. In this paper, the semantic association extraction looks like the SPARQel language in the manner of association extraction which connects two resources, but which includes additional functionalities (association evaluation and ranking).

IV. Specification of Semantic Association in RDF Base

RDF (Resource Description Framework) is based on a model of triple (subject: property: object) whose values are either resources, literal or blank nodes. Each element (subject or object) of the triple constitutes an entity or a resource of RDF graph. A property/predicate is a binary relation between two entities (resources). Let *I*, *B* and *L* (IRIs, Blank nodes, and literals) pairwise disjoint finite sets. The triple (subject,predicate,object) $\in (I \cup B) \times I \times (I \cup B \cup L)$ is called an RDF triple. An RDF graph (RDF dataset) is a set of RDF triples. A path in RDF graph represents a variety of explicit relations. The existence of a path through which the entities are connected represents the foundation of the assumption which affirms the existence of semantic associations between these entities. We can distinguish several types of semantic associations. In the work of [5], Anyanwu and Sheth present an approach based on the ρ operator for semantic association extraction. More specifically, the ρ operator provides a mechanism for semantic associations reasoning which exist in a knowledge base (RDF graph). Two entities contained in a knowledge base are semantically connected when they are associated through the ρ-path, ρ-join or ρ-iso operator.

Figure 1 shows some examples of semantic associations. Being given two entities e_1 and e_2, in Figure 1 (a), ρ-path identifies direct paths which connect e_1 to e_2. The ρ-join operator of Figure 1(b) seeks if there exist direct paths between

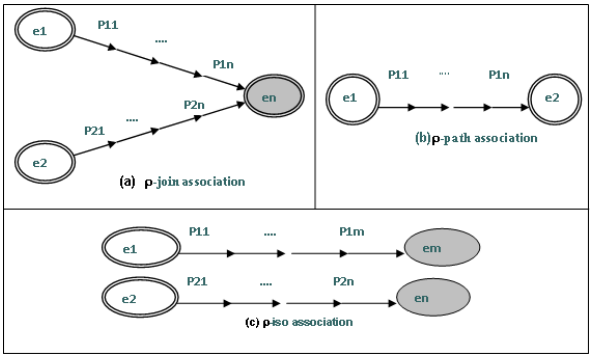

Fig. 1. Semantic Association types.

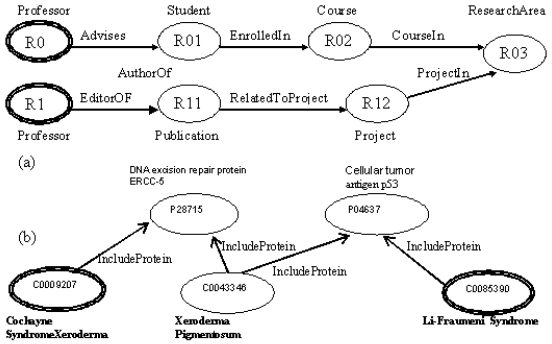

Fig. 2. Indirect Association Example.

e_1 and e_2 which meets some node e_n. Finally, Figure 1(c) seeks pairs of directed sub-graphs of which the roots are, respectively, represented by e_1 and e_2 and both sub-graphs are semantically similar. In addition to semantic associations through direct paths (example of relation between the entity of the type professor (R0) and the entity of the type ResearchArea (R03))(Figure 2), it is important to drive semantic associations with indirect paths. A good illustration of this observation is schematized by the Figure 2-a, describing indirect association between the entities of the type professor R0 and R1. In spite of the inexistence of direct path which exists between these two entities, R0 and R1 are semantically associated. Semantic association results from the existence of the direct path which exists between R0 and R1 towards the same entities R11 and R03. In the same manner, in the Figure 2-b, it is interesting to seek indirect associations to identify the network of proteins shared by two different diseases. In the example of the Figure 2-b, we can conclude that the disease "Li-Fraumeni Syndrome" and the disease "Cockayne syndromeXeroderma" have proteins in common, which make it possible to facilitate the identification of the fundamental molecular base shared between these two diseases [28].

In this paper, a semantic association is a direct or indirect path connecting two entities contained in RDF graph. We present in this section some formalism concerning the definition of semantic associations by taking into account some connections types between entities.

Being given an RDF database (Ω). Let Ωp and Ωe two sets respectively indicating the entities (class instances, literal) and the properties (predicates) contained in Ω. Each property pi contained in the Ωp set has Re (pi) range and **D**(pi) domain. **D**(pi) and Re (pi) are obligatorily included in Ωe. A path (association) in Ω is formed by a sequence of entities contained in Ωe connected by a sequence of properties included in Ωp.

Definition1: direct association Two entities e_0 and e_n are semantically connected by a direct association if there exists a direct path between e_0 and e_n of the following form: ($e_0 \xrightarrow{p_1} e_1 \xrightarrow{p_2} e_2 ... \xrightarrow{p_{n-1}} e_{n-1} \xrightarrow{p_n} e_n$). Where e_i ($1 \leq i \leq n$) is an entity of order i and P_j($1 \leq j \leq n$) is a property of order j. The properties set p_1, p_2,..., p_{n-1} indicate the sequence of the properties (PS) which connects the entities sequence ES (e_1, e_2,..., e_{n-1}, e_n). The association length is n which represents the number of triplets in association. In this paper we explore only simple direct associations (the entities which form an association must be distinct). If an association contained a repeated entity, then association is complex and can involve infinite loops (especially if the maximum number of repetition is not fixed).

Definition 2: indirect association: Two entities e_0 and e_n are indirectly associated if there exists a semantic connectivity between e_0 and e_n independently of direction of the properties (arcs carried out by e_0 and e_n). For example, $e_0 \xrightarrow{p_1} e_1 \xleftarrow{p_2} e_2 ... \xrightarrow{p_n} e_n$ is an example of indirect paths between the entities e_0 and e_n. In this type of association, we consider only simple indirect paths.

Definition 3: Two entities e_0 and e_n are semantically associated if there exists a direct or indirect path in Ω making it possible to connect these two entities.

V. Analysis of Semantic Association Quality

In the work of [29], Tartir and Arpinar provide some ontology evaluation techniques by using statistics metric related to the knowledge contained in ontology. The authors divided their evaluation techniques into two categories: metric based on schema ontology and metric based on knowledge base (instances). A semantic association is a path in knowledge base and thereafter, if the knowledge base evaluation is possible, then a path in a knowledge base can be also evaluated. From these intuitions, we can derive our evaluation metric to evaluate associations based on knowledge base metrics.

A. Association connectivity

As defined in the work of [29], the metric of class connectivity represents an indicator of class importance contained in ontology while being based on the graph instances paths. This measure is adopted to measure the class importance compared to other classes contained in ontology. In a similar manner to class, it is important to identify association significance in a knowledge base compared to other associations having the same source and target entities.

Definition 4: The class connectivity ($\mathcal{C}(c_i)$) is defined by the relationship number of class instances with other classes

instances (RNCI(C_i)).

$$\mathcal{C}(c_i) = |RNCI(C_i)|. \quad (1)$$

Definition 5: The association connectivity ($\mathcal{C}$(A_i)) is defined by the average of class connectivity of association entities.

$$\mathcal{C}(A_i) = \frac{\sum_{i=1}^{N} \mathcal{C}(c_i)}{N}. \quad (2)$$

Where N represents the entities number contained in association. The association connectivity makes it possible to identify the paths having an important connectivity in a knowledge base.

B. Association Importance

This metric allows the percentage calculation of the instances contained in a structure of sub-class tree of a well defined class compared to the full number of instances contained in a knowledge base. This measure is adopted to specify the focal part of a defined ontology.

Definition 6: The class importance ($\mathcal{I}$(C_i)) contained in ontology is defined by the ratio of instances number (in (C_i)) contained in subclasses graph of the root C_i and the sum of instances number of the classes contained in the knowledge base ($\mathbf{B}$(IC)).

$$\mathcal{I}(C_i) = \frac{In(C_i)}{\mathbf{B}(IC)}. \quad (3)$$

Definition 7: The importance of semantic association ($\mathcal{I}$(A_i)) in a knowledge base is defined by the ratio of class importance sum of classes (n) relative to association entities by the number of class instances ($\mathbf{B}$(IC)) contained in a knowledge base.

$$\mathcal{I}(A_i) = \frac{\sum_{i=1}^{n} \mathcal{I}(C_i)}{(\mathbf{B}(IC))}. \quad (4)$$

The association importance can be adopted to identify the most important path or association in a knowledge base.

C. Association Richness

The class richness measure is important because it makes it possible to identify how much properties defined for a class C_i in the schema ontology are used by the instances I_i of C_i on the level of knowledge base.

Definition 8: The class richness $\mathcal{R}$(C_i) is defined by the proportion of the properties number used by its instances (P (I_i, I_j)) divided by the properties number defined for C_i in the level of the schema ontology PC(I_i, I_j).

$$\mathcal{R}(C_i) = \frac{|P(I_i, I_j)|}{|PC(I_i, I_j)|}. \quad (5)$$

Definition 9: The association richness $\mathcal{R}$(A_i)is defined by the proportion of classes richness of entities (n entities) contained in association divided by the number of entities contained in a defined association (T).

$$\mathcal{R}(A_i) = \frac{\sum_{i=1}^{n} \mathcal{R}(C_i)}{T}. \quad (6)$$

The association richness is very important to identify semantically rich path in a knowledge base.

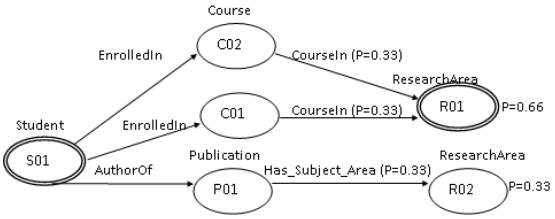

Fig. 3. Exemple 1: identification of frequent entity.

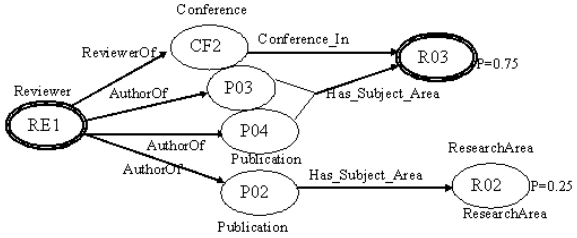

Fig. 4. Exemple2: Association guided by the most frequent entity.

D. Association based entity frequency

Intuition 1: The entity description is made up by the probability of the properties transmitted by the predecessor entities.

Example 1: It is a question to identify the research field of a student through the use of semantic associations.

In Figure 3, let us suppose that the entity source of association is known (S0<rdf:type>student) and that the target entity is unknown (only the entity type is known: (?<rdf:type>:ResearchArea)). But, while knowing the properties predecessors ($\xrightarrow{CourseIn}$, $\xrightarrow{CourseIn}$, $\xrightarrow{HasSubjectArea}$) coming from the entities C02, C01 and P01, we can specify the target entity. In the case of Figure 2, the entities predecessors transferred some amount of information (frequency of the properties) to R01 and R02, for that we can obtain some knowledge for the choice of the target entity.

In addition to the mentioned descriptions earlier, it is necessary to formalize these ideas. The probability value of a defined entity is obtained by the sum of the probabilities resulting to the properties coming from the predecessors entities (in Figure 3, P (R01) =0.66, P (R02) =0.33). The selected entity is the entity which has the maximum value of probability. In the example of Figure 3, the target entity became R01 since it represents the most frequent entity.

Example 2: It is a question to identify the expertise field of a *reviewer* for a well defined conference through the use of semantic associations (Figure 4). In this example, the target entity which must be selected is R03 since it presents a maximum frequency value.

VI. PMSPARQL LANGUAGE

Being based on the needs remarked on the level of knowledge extraction starting from structured data under format of a graph or network, such as biological networks or social

networks, we released some specification for the development of PmSPARQL language. PmSPARQL is an extension of SPARQL intended to discover semantic associations located in a knowledge language (OWL ontology, RDF graph). Our intuition is to make minimal changes on the level of SPARQL syntax and semantics. Our approach achieves two fundamental characteristics. Firstly, PmSPARQL requests are entirely compatible with SPARQL requests. Secondly, PmSPARQL request makes possible the semantic associations extraction based on some specified criteria defined by a request. These PmSPARQL characteristics facilitate the task of associations ranking and presentation to a defined user according a specified criteria. The association extraction through PmSPARQL is based on the construction of path patterns (triple pattern created with the use of a path variable in a place of the property) including direct or indirect paths. The syntax of PmSPARQL represents a little modification of the SPARQL language grammar. The new functionalities added by PmSPARQL make possible the following operations:

- Find located direct paths between two entities specified without interest of associations values (the interest is carried to discover all semantic association without specification of length or importance).
- Find located paths by associations filtering through regular expressions relating to associations properties.
- Find located paths with informational content value of a specified property or entity delimited by a well defined interval.
- Find located associations by imposing constraint on the length of path.
- Find all paths which include the presence or the absence of a property with a defined position in association.
- Find all ordered paths according to the value of an association with a specified criterion by user.

A. the syntax of a path pattern in PmSPARQL

In this section, we give an algebraic formalization of the syntax of PmSPARQL over a simple graph RDF, without RDFS vocabulary and literal Rules (for details on RDF formalization see [30], and on RDFS vocabulary see [31]. In a likewise manner of [32] we denote by *IL* the union $I \cup L$, and by *T* the union $I \cup B \cup L$. Assume additionally the existence of an infinite set V of variables disjoint from the above sets. A tuple from $(IL \cup V) \times (I \cup V) \times (IL \cup V)$ is a triple pattern. In a likewise manner, the expression of a *path pattern* is similar to the triple pattern described in SPARQL [32] except that on the level of the property position will be placed an entity of type path (association expression: <subject: path: object>). An entity of type path is a sequence of several RDF entities. The sequence is composed by a not ordered list of properties connecting the subject and object entities, similar to the definition of semantic association presented earlier. In a more formal manner, A path pattern expression could consist of a SPARQL triple pattern [32] with a path variable in the predicate position (called a path pattern) and some built-in path filter conditions. Let *RP* and *PV* (regular path and path variables respectively) two pairwise disjoint set of variables that are disjoint from $(I \cup L \cup B)$. A triple pattern is a tuple in $(IL \cup RP) \times (I \cup RP) \times (IL \cup RP)$. To specify a desired path pattern it will be necessary to exploit regular expression over triple pattern. If we considers E as a set of regular expressions and *re* an expression included in E, then *re*, *re**, *re+*, *re*? are also regular expressions. In a likewise manner, if *re1*, *re2* are a regular expressions then [*re1* AND *re2**] and [*re1* OR *re2**] are also regular expressions. Let the expression T*re* a regular expression applied to a triple pattern or an extension of regular expression of the form ($[e_1*]$, *P*, $[*e_n]$) where (e_1, *P*, e_n) is a triple path pattern. T*re* looks for paths including arbitrary entities whose subject of first triple is e_1 and the object of last triple is e_n and of which all the properties (predicates) are represented by *P*. The * after e_1 and before e_n means all the intermediate nodes connecting the source and target entities and all the predicates on the path are represented by *P*. Let *RT* represent the set of extended regular expression over *T*. In the following section we provide some developed built-in path functions:

- PropertyFrequence (PV, R*e*, interval) ⟶boolean: is a Boolean function which indicates if there exist paths including a predicate "Pr"$\subset I$ with a specified informational contents value.
- Size (PV) ⟶Number: is a function of type integer which identifies the size of association path (number of properties).
- MatchingResource (PV, R*e*) ⟶boolean: This function returns if a resource specified by a regular expression on a triple is contained in a path or not.
- MatchingPattern (PV, *RT*) ⟶boolean.
- MatchingProperty (PV, *p*, pos) ⟶boolean:This function returns if a property is contained in a path at a position pos.
- MatchingProperty (PV, *p*+) ⟶boolean:This function returns if a path includes only the property p.
- IndPath(PV,[*p1-P2*]+)⟶boolean:This function returns indirect path which includes a sequences of pairs of properties: *p1* followed by the inverse of the property *p2*.

Definition 10: A Path Pattern Expression (*PaP*) is defined recursively as follows:

- A tuple from $(IL \cup PV) \times (RP) \times (IL \cup PV)$ called a *path triple pattern* is a (*PaP*).
- If *TP* is a SPARQL triple pattern and (*PaP*) is a path pattern then (*TP* AND *PaP*) is a path pattern.
- If *PaP* is a path pattern and *PC* is a path built-in condition, then the expression (*PaP* PFilter *C*) is a path pattern.

Remark: A Path built-in condition is constructed using elements of the set $V \cup IL \cup PV, logicalconnectives(\neg, \wedge)$, inequality symbols ($\leq$,$\geq$, <, >), equality symbol (=) and path built-in functions in the following manner: Given a variable path @P (a name beginning with the @ caracter), a constant c, i$\subset I$, T*re* a regular expression, the following are path built-in conditions:

1. PropertyFrequence (@P, R*e*)= c, Size(@P)= c, MatchingResource(@P, R*e*), MatchingPattern (@P, *RT*) and

MatchingProperty (@P, *p*, pos),MatchingProperty (PV, *p*+), IndPath(@P,[*p1-P2*]+).

2. If PC1 and PC2 are path built-in conditions, then (¬ PC1) and (PC1 ∧ PC2) are path built-in conditions.

B. Semantics of path Queries in PmSPARQL

In [32] the semantics of graph pattern expressions is defined as a function [[.]] which takes a pattern expression and returns a set of mappings μ. A mapping μ is defined as a partial function from *RP* to *T* defined earlier. The function dom(μ) is used to define the subset of *RP* in which μ is defined. This definition will be used as an extension of PmSPARQL's path patterns.

Let X the set of possible triple from T. We introduce the notion of *pmapping*(path mapping) ω as a function from (*RP* ∪ *PV*) to (*X* ∪ *T*) such that ω(*pv* ∈ *PV*)=p ∈ *X* and ω(*rp* ∈ *RP*)=X. Then, let *pt* a path triple pattern and ω(*pt*) the tuple obtained by remplacing any variable *rp* ∪ *pv* in *pt* according to ω. The dom of ω is the subset of *PV* ∪ *RP* in which ω is defined and is denoted by dom(ω). In addition, the compatibility used in [32] will be extended, in this paper, to include compatibility between a mapping μ and pmapping ω. When for all x ∈ dom(μ) ∩ dom(ω), we have μ(x) ∈ ω(x), then μ and ω are compatible. After that, the join defined in [32] will be extended as the join between a set of mappings Υ and a set of pmappings Ψ in the following manner:

$\Upsilon \lhd\rhd \Psi = \{\mu \cup \omega \mid \mu \in \Upsilon, \omega \in \Psi \text{ are compatible}\}$.

We are ready to define the semantics of path triple pattern expressions as a function $[[]]_D$ which takes a pattern expression and returns a set of pmappings.

Definition11: Let D be an RDF dataset over T, *pt* a path triple pattern whose variables are defined by var(*pt*) and GP a graph pattern. The solution of a path pattern *PaP* over D, obtained by $[[]]_D$ is defined recursively as follows:

1. $[[pt]]_D = \{\omega \mid$ dom(ω)=var(*pt*) forms a path in D$\}$.
2. $[[PaP \text{ AND } GP]]_D = [[PP]]_D \lhd\rhd [[GP]]_D$.

For the path patterns with PFILTER expressions, we say that a pmapping ω satisfies a built-in condition BC or $\omega \models$BC if given I' a subset of I (set of IRIs) and *tr* a tp-regular expression,

1. BC is MatchingResource(@P, I') and @P ∈ dom(ω) and I' ∩ ω(@P)≠∅.
2. BC is MatchingPattern(@P, I') and @P ∈ dom(ω) and I' ⊆ $\omega(@P)$.

C. PmSPARQL Query Examples

This section gives a feel of PmSPARQL Query examples:

Query1 (Simple Path Query): The following select path query, involves the extraction of all possible associations with two fixed path pattern (two authors):

```
SELECT @P WHERE
{<http://dblp.uni-trier.de/rec/bibtex/books/aw/AbiteboulHV95>
@P <http://dblp.uni-trier.de/rec/bibtex/conf/vldb/AroraC78>};
```

By default, the matched paths must be directed. The variable @P is bound to the located path, represented as a properties sequence and the connecting resources specified by the request.

Query2 (Path Query With Single Resource): Find all paths connecting the resource s="AbiteboulHV95" of type "Publication" with any other resources/entities:

```
SELECT @P WHERE
{<http://dblp.uni-trier.de/rec/bibtex/books/aw/AbiteboulHV95>
@P ?x};
```

The following single source SELECT query locates associations reachable from resource s. The analogous form of the above query relies on the inverse path pattern of the form {?x @P <r>}. This pattern matches all paths from which the resource r is reachable.

Query3 (Path Query including a specified resource): matches any semantic path between the resources x and y, provided the path which includes the resource e. For example, find any associations including a biologic molecule BAND-OF the molecule "cbo.400.8q21.1":

```
SELECT @P WHERE {?x @P ?y
?e cbo:BAND-OF <cbo.400.8q21.1>
PFilter (MatchingResource(@P,?e)
};
```

Query4 (Path Query including the specified entity with a max frequency): Matches the association having the entity with the maximum incoming predicate frequency. For example, find the path and the appropriate research area of specified reviewer (Reviewer1):

```
SELECT @P, ?y WHERE {<Reviewer1> @P ?y
?x Has-Subject-Area ?y
PFilter (Max(PropertyFrequence (@P, ?y))
};
```

Query5 (Path Query extracting ranked association according to the Richness value): Matches ordered associations according the richness value defined earlier. For example, find all ranked associations reachable between source and target entities s and t according to the ordered richness values(desc or asc):

```
SELECT @P WHERE {< s > @P < t >
PFilter (RichnessRanking(@P, asc/desc))
};
```

If the ranked associations concern only a specified fixed number of best or bad associations, it will be necessary to include the number of associations to return like this expression (PFilter(RichnesRanking(@P, n, asc/desc))). In a likewise manner of the richness function, the expression of association ranking query according to the connectivity, the importance and the uncertainty functions are similar to the query 5. The PFilter condition is defined, respectively as follows: ConnectivityRanking(@P, asc/desc), ImportanceRanking(@P, asc/desc) and UncertaintyRanking(@P, asc/desc).

Query6 (Path Query with Path length constraint):Find all associations between the publication 0-201-53771-0 and all publication having cited AptBW88 with a length <5 hop (predicates):

```
SELECT @P WHERE {?x @P ?y
?x pub:isbn <0-201-53771-0>
?y pub:cites <http://dblp.uni-trier.de/rec/bibtex/books/mk/minker88/AptBW88>
PFilter (Size(@P)<5)
};
```

Query7 (Path Query with path pattern constraint): Find existing social relationships between SarahWhite and Zackary-Black:

```
SELECT @P WHERE {?x @P ?y
?x foaf:name ”SarahWhite”
?y foaf:name ”ZackaryBlack”
PFilter(MatchingPattern (@P,[?x*] foaf:ElectedLeader [?y*])
};
```

Query8 (Path Query including predicate at a specific positions): Matches all semantic paths between two resources r1 and r2, provided the path including a property p in a position i:

```
SELECT @P WHERE {<r1> @P ?y <r2>
PFilter(MatchingProperty (@P, p, i))
};
```

Query9 (Path Query including a single repetitive property): Matches all semantic paths between two resources r1 and r2, provided the path including a unique property ”cbo:BandOf”:

```
SELECT @P WHERE
{<r1> @P ?y <r2>
PFilter(MatchingProperty (@P, ”cbo:BandOf+”))
};
```

Query10 (Path Query including indirect associations): The located paths may not be fully directed, but guided by specified individual properties. This may be requested by a suitable path expression, as in the query example:

```
SELECT @P WHERE
{<r1> @P ?y <r2>
PFilter(size(@P)>=6 AND size(@P)<=8 AND IndPath(PV,[p1-P2]+))
};
```

D. Prototype Implementation of PmSPARQL Language

The implementation of PmSPARQL uses *Powl* [2], the Web based platform for collaborative semantic Web development as our RDF storage system. Our prototype implementation is based on low *Powl*'s level API to iterate over triples (subject, predicate, object) with several criteria. Each graph pattern is obtained by a path iterator, according to a specific query. The developed path iterator, useful for path and pattern matching, has been implemented based on Bidirectional breadth-first search (BBFS) strategy. In [17], the authors show that a bidirectional breadth-first search strategy is the most efficient method in practice for finding all simple paths up to some hop limits. The path search uses the steps from BBFS to recursively find the entities used for the path construction. For the research of indirect path, it will be necessary to use two paths iterator. A forward path iterator which locates a path (P1) from the source resource frontier (r1) and a backward path iterator which loactes a path (P2) from the target resource frontier (r2). A candidate indirect path between r1 and r2 is located when an entity from the forward frontier matches an entity from the backward frontier. Figure 5 shows our system for multi-paradigme RDF querying useful for pattern matching and path queries. The first step in our prototype is to load RDF graph into our *Powl* RDF data store. Then, according to a paradigm request different processing steps are performed on the data which produces appropriate data stores on the result of each paradigm. The result of pattern matching request is stored in *Pattern Store*, the result of path matching request is stored in *path store* and the result of top K-ranked paths is stored in a *rank store*. Our different storage layer uses adoDB [33] data storage system compatible with any relational databases.

[2]http://powl.sourceforge.net/swc

E. Path Query Processing in PmSPARQL

```
Algorithm Path-Finder (Node r, Node s, Paths[])
i=0, Nbhop=0, HopMax=value, j=0;
Begin
ListNode[i]=r;
ConstructPath[i]=r;
ListHop[i]=0;
    While(!empty(Head(ListNode)))
     Nbhope= ListHop[i];
    Begin
      r=Head(ListNode);
      if(!empty(triple-finder(r))
      listp(r, p', s')←—triple-finder (r);
      else
      Delete(ListNode[i]);
      if(!empty(listp))
          foreach r, p', s' ⊂ listp
          Begin
          Nbhop=Nbhop+1;
          ConstructPath[i] = ConstructPath[i] ∪ p' ∪ s'
            if( s'=s or HopMax=Nbhop )
              Begin
              Paths[j]=ConstructPath[i];
              j=j+1;
              End
              else
              Begin
              i=i+1;
              InsertNode(ListNode, HeadPos, s');
              ListHop[i]=Nbhop;
              End
          End
          else
          Begin
            DeleteNode (ListNode, HeadPos);
            DeletePath (ConstructPath, HeadPos);
          End
    End
return Paths;
End
```

The path extraction in PmSPARQL is based on the *Path Iterator* which returns a final path expressions (fp-expression) by a decomposition process of successively retrieved relevant path expression (p-expression) from disk. The *Path Iterator* uses the following *Path Finder* algorithm. The *Path Finder* algorithm begins by initializing a set of variables. The Path Query algorithm returns a list of fp-expression (Paths[]) depending on the subgraph in which the source and the target node (r and s) of the query is located. The Path Finder algorithm needed to process recursively over rdf graph having a root node r and a leaf node s. This algorithm looks to construct all possible fp-expressions (Paths) having a maximum length size

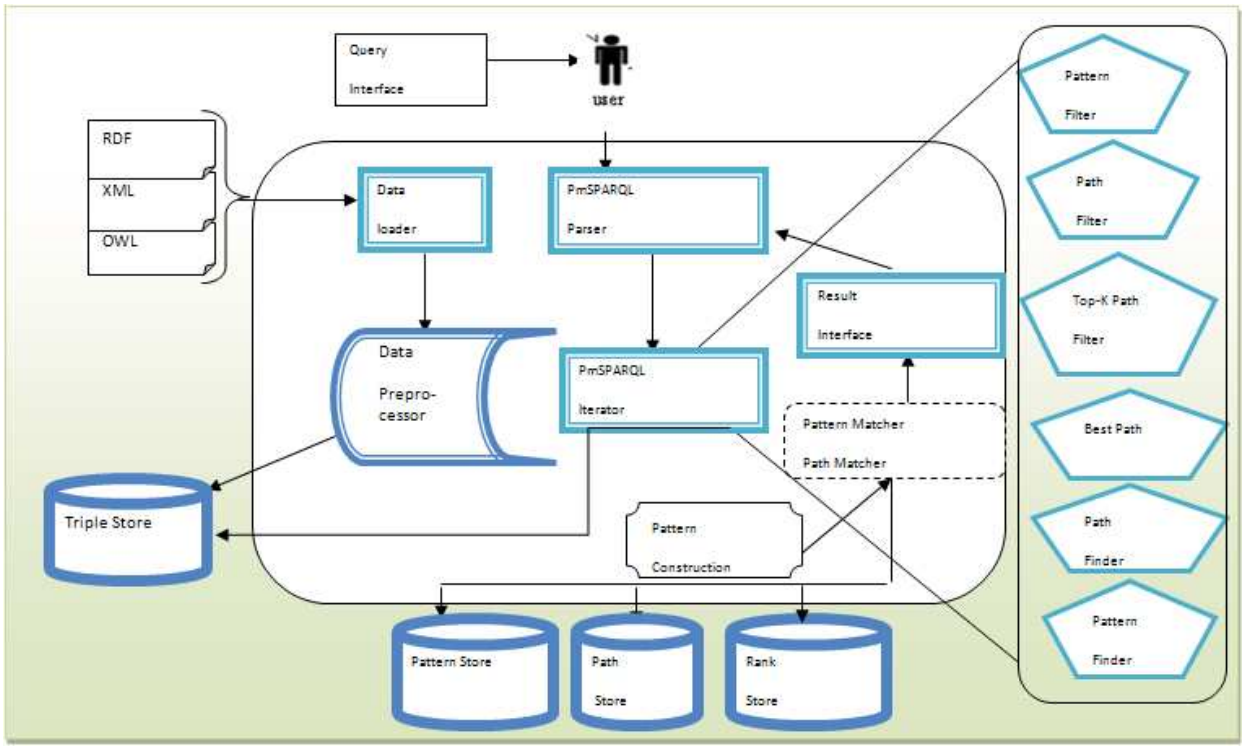

Fig. 5. Prototype Architecture.

equal to *HopMax* by a union operator of existing p-expression (listp (r, p' , s')) obtained from a triple-finder function begin with the source (r) up to the destination (s'), p-expression (obtained by triple-finder) are prepended to the fp-expression in ConstructPath[i] ∪ p' ∪ s'. The variable i represents the current fp-expression index currently in construction. Our Path-Finder algorithm operates over the head of three lists: ListNode, ConstructPath and ListHope. The first list is used to stores the last node (at the head) that will be used in a possible decomposition process of the last fp-expression of the second list (at the head) and the Listhop store the last length of the last fp-expression of the second list. For example, if the last fp-expression (Head of the ConstructPath list) is represented by the expressions ([r,p,r'],p',s'],p1,s1]), then the last node (Head) stored in ListNode is s1 and the value stored in the head of the ListHope is the number of predicates=|p,p',p1|=3. If fp-expression is terminal (the last node in ListNode=s or HopMax=Head(ListHope)) then the decomposition process is stopped and the current fp-expression in the head of ConstructPath will be stored in the final path list (Paths[]), else the process of path constrcution will be continued. A similar solution is used for path patterns with other paradigms but using a minimal modification.

VII. Experimental Design and Evaluation

In this section, we describe our implementation of PmSPARQL by an empirical evaluation of our query processing approach and the performance evaluation of multi-paradigm queries.

TABLE I
Properties of the DataSets and Knowledge bases.

Dataset and kBase	Number of RDF Statement	Number of Classes	Number of relat-ionships	disk file size
SwetoDBLP-Aug-2007	305000	21	22	25M
CBO	31893	4069	8463	3.02M
GlycO-0-9-full-enzyo	15259	338	81	1.25M
SwetoDBLP-apr-2008-p1	610360	21	22	50M

A. Implementation

Our tests are performed on machine with Intel(R) Core(TM) 2 Duo 2.2GHZ CPUs and 3GB memory. We used Powl, an efficient and easy platform for collaborative semantic Web development as our RDF main-memory storage. We implemented our algorithms using php 5.

B. Data Sets

We used a real world SwetoDBLP-August-2007 dataset, SwetoDBLP-april-2008-part1 dataset [3], an extract of CBO dataset [4], GlycO-0-9-full-enzyo dataset [5].

Table 1 above shows the properties of the datasets and knowledge bases. Glyco knowledge base represents information about glycans and includes a comprehensive schema as well as instances and is loaded in disk from owl file. Currently, the ontology has 338 classes and 81 specialized properties.

[3]http://lsdis.cs.uga.edu/projects/semdis/swetodblp/
[4]http://clinbioinformatics.org/cbopublic
[5]http://lsdis.cs.uga.edu/projects/glycomics/2006/GlycO-0-9-full-enzyo.Owl

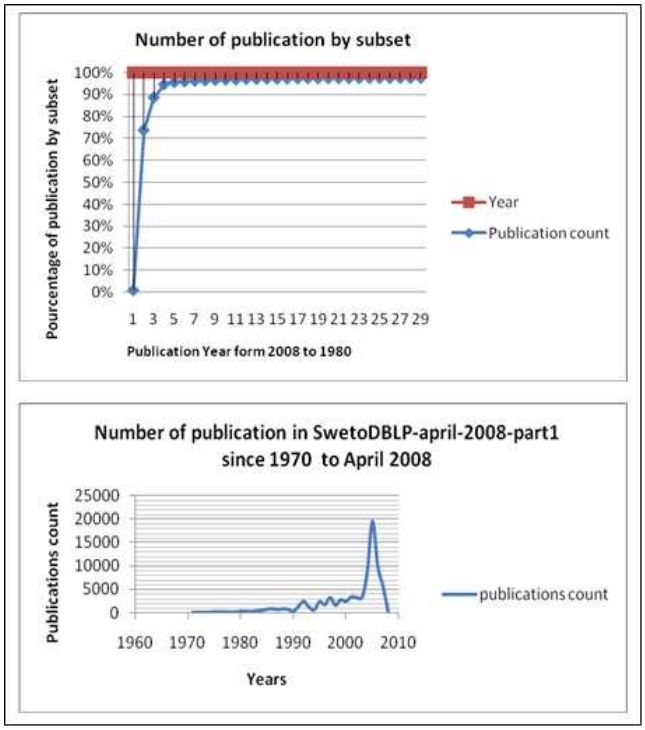

Fig. 6. Number of publications in SwetoDBLP-april-2008-part1 starting from year 1970 and sizes of used subsets in datasets.

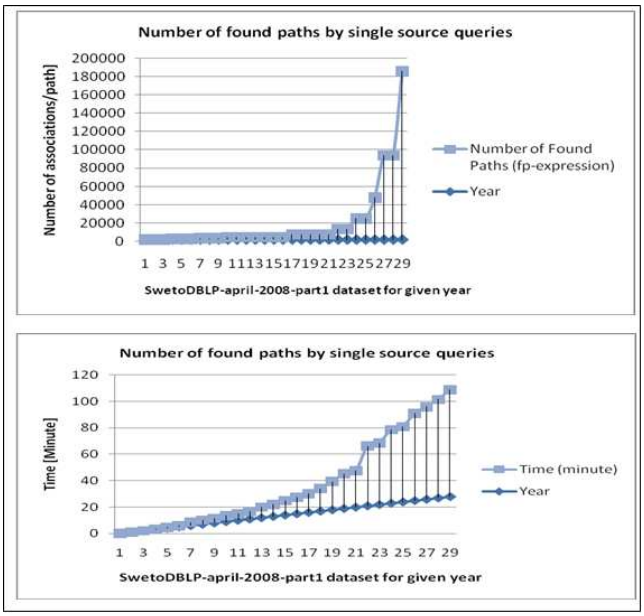

Fig. 7. Number of found paths in SwetoDBLP-april-2008-part1 and execution time for single source path query.

The CBO dataset (Clinical Bioinformatics Ontology) is a semantically structured controlled vocabulary for molecular diagnostics that addresses the need for consistent representation of clinically relevant molecular biological and cytogenetic entities. To test the scalability of our experiments it is useful to use a much bigger data set. For this purpose we use two versions of DBLP ontology generated from the data available in SwetoDBLP-August-2007 and SwetoDBLP-april-2008-part1. The data contains information about authors, published papers, articles, year of publication, etc. The property *cites* is useful to derive associations (paths), but it is assigned to few documents (3067 occurrences), rendering this set unsuitable for scalability test purposes.

To be able to search for long and meaningful associations, we have added to the SwetoDBLP-april-2008-part1 dataset a new list of randomly created citations (1 to 12 random citations to papers selected randomly from some previous years in the knowledge base, using a normal distribution). The total number of randomly inserted citations in the full dataset reached almost 1000000 statements. The SwetoDBLP-april-2008-part1 dataset contains over 80657 publications with some publish year. To test the scalability of our algorithm, we used some publications between 1980 and 2008. It contains 78080 publications subdivided into 28 subset, each one includes publications starting with 2008 and ending with 1980 (the smallest subset included only 2008 publications (20 publications) and the largest one included publications from years 1980-2008 (77830 publications)). Figure 6 presents numbers of publications in SwetoDBLP-april-2008-part1 datasets (starting from year 1970) and sizes of all used subsets in datasets (starting from 1980).

C. Performance Test on SwetoDBLP-april-2008-part1 dataset

We evaluate the performance of the techniques by observing 1) the size of the returned associations i.e. the number of fp-expressions brought into memory from disk and 2) the query processing time including the time of fp-expression construction. In order to test the scalability of our algorithm, we randomly chose two papers published in 2008 and executed single specified resource query to find all paths leading to papers they cited, using the relation/property "*cites*". We present in the following paragraph an example of PmSPARQL query with single resource:

```
SELECT @P WHERE
{<http://dblp.uni-trier.de/rec/bibtex/books/sp/Helmert2008> @P
?x
PFilter(Size(@P)<=28 AND MatchingProperty(@P,
"opus:cites+"))
};
```

Our tests use queries performed on increasingly larger datasets, starting with publications in 2008 and ending with publications during 2008-1980. The result of those queries is presented by the plot in Figure 7 which includes the execution time for all queries for each dataset and the number of located paths. In the obtained results, the number of paths increased exponentially as the publications from the previous years were added. The execution time presented in the Figure 7 also followed the exponential growth, but even for the longest path (length=28) did not exceed 81 minutes to generate 184320 paths.

We compare the execution time of our performed tests with the tests described in [9] on query applied to articles published

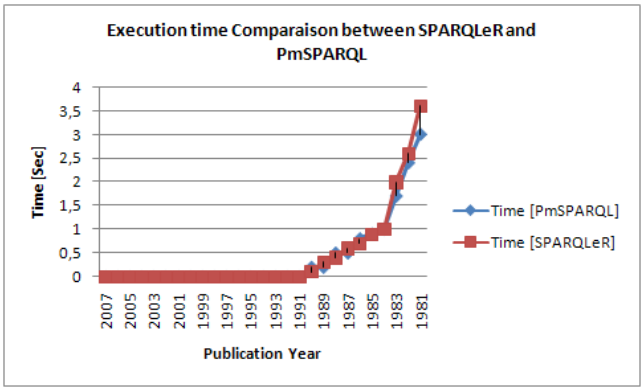

Fig. 8. Execution Time Comparaison between SPARQeL and PmSPARQL

only in 2006 and ending with articles published during 1981-2006. The maximum of path count in each test is equal to 6. The plot in Figure 8 gives information about our algorithm performance.

In addition to the tests applied to requests with single source, we perform some experimentation for finding semantic associations between two specified entities (source and target entities). An example of PmSPARQL Query with two specified entities is given below:

SELECT @P WHERE

{<http://dblp.uni-trier.de/rec/bibtex/books/infix/Makoui2007> @P

<http://dblp.uni-trier.de/rec/bibtex/books/sp/Wood80>

PFilter(Size(@P)<=28 AND MatchingProperty(@P, ”opus:cites+”))

};

We identify 4 target entities for each of the two starting entities from the year 2008. These 4 entities are chosen as target entities for path queries between two resources. The queries were performed on increasingly larger dataset having a length increases from 1 to 28. For each query execution we specify the source entity s=”http://dblp.uni-trier.de/rec/bibtex/books/infix/Makoui2007” and the target entity (r) applied for a given year which has a path from s to r. For the year 1980, as mentioned in the above query the entity r=”http://dblp.uni-trier.de/rec/bibtex/books/sp/Wood80”. The plots in Figure 9 present the execution time and number of located paths which represents the case of two specified entities path query.

In the performed experimentations, the search space was become larger than the search request for single source queries. But, the execution time did not exceed the 19 seconds, which represents an indicator of a good result for path extraction with 27 hops length. The results represent a good proof for the usability of our PmSPARQL language for a large and long path extraction. This results of our scalability experiments demonstrate that in some practical cases, path extraction can be returned with a reasonable time execution, even for specified long paths.

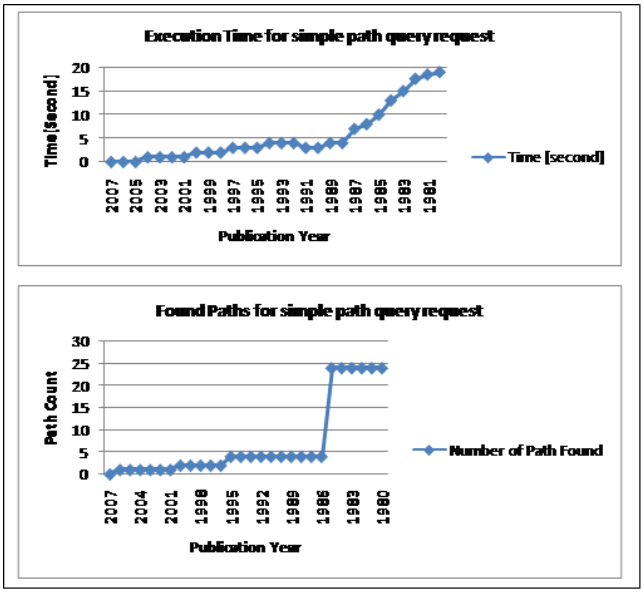

Fig. 9. Number of found paths and execution time in SwetoDBLP-August-2007-part1 for Simple path Query Request

D. Ranking Association based on connectivity measure

Ranking associations are inherently different from ranking documents. There exist several ways to measure and rank relevance of association. To relevantly rank the resulting associations, it will be necessary to define a flexible query for association evaluation. In our context an association rank is a function of specified criteria. Due to the space limitations here, we will only discuss the association rank which takes into account the association connectivity contained in association (defined above). The Queries performed to ranking associations are applied to the SwetoDBLP-april-2008-part1 dataset. An example of PmSPARQL Query for ranking associations is:

SELECT @P WHERE

{<http://dblp.uni-trier.de/rec/bibtex/books/ws/BMW07-papers/BandyopadhyaySMM07> @P ?t

PFilter (ConnectivityRanking(@P, desc) and MatchingProperty(@P, ”opus:cites+”) AND Size(@P)=3)

};

To be able to search for meaningful ranked paths, we have created a list of randomly citations between the publications of the years 2008 (of type Book-Chapter) and the publications of the year 2007 (of type Scientific-Publication). In a likewise manner, we have created a list of randomly citations between the 2007 publications (of type scientific-publication) and some 2006 publications (of type Book-Chapter). Finally, we have created a list of randomly citations between the 2006 publications (of type Book-Chapter) and a mixed 2005 publications (of type scientific-publication and Book-Chapter). The result of the performed tests is presented in the plot of Figure 10. The results in the Figure 10 represent the connectivity of some classes contained in the SwetoDBLP-april-2008-part1 dataset and the ranked associations of the returned paths with length 3. The intuition behind ranking associations is to provide measurable and quantifiable associations which can be

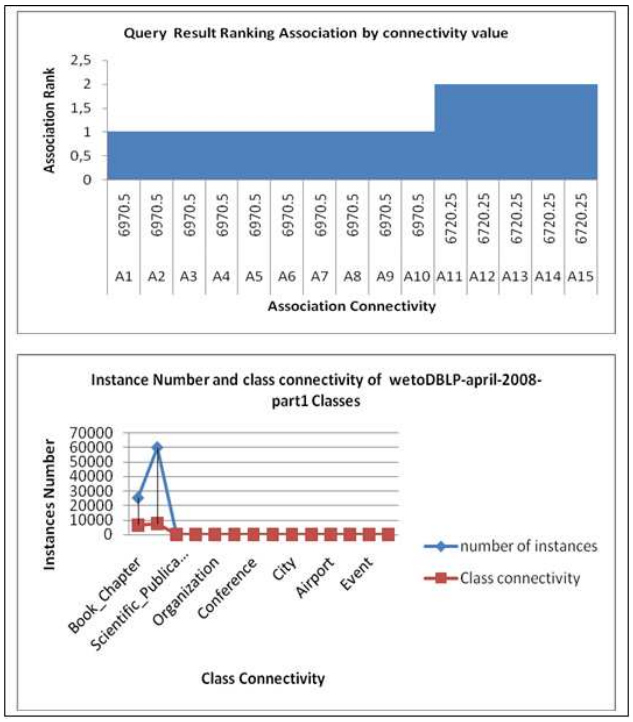

Fig. 10. The classes connectivity contained in Sweto-DBLP ontology and query results applied to ranked associations based on the association connectivity

classified to the end-user according to the rank value. In the case of the SWETO ontology the results of ranked associations return the most connected associations in the ontologies, it can be seen that the results also include good information about publications paths other than publications paths.

In addition to the tests applied to SwetoDBLP-april-2008-part1 dataset, we have performed some tests applied to the glyco ontology for ranking pathways based on some criteria. To avoid the complexity of presentation, the results of the performed tests are not presented here

E. PmSPARQL Test in the Biomedical domain

Our tests are performed on a wide set of RDF base to evaluate the functionality of PmSPARQL for finding long and complex paths. Among those tests, we choose to discuss a representative query in the biomedical domain, which try to finding reactions in the Glycan biosynthesis pathway. We propose in this query test to search reactions between N-Glycan-G00020 and N-Glycan-G00022. An example of PmSPARQL Query is presented in this example:

SELECT @P WHERE
{<glyco:N-Glycan-G00020> @P ?y <r2>
PFilter(size(@P)<=8 AND IndPath(PV,[*has-product-has-reactant*]+))
};

The glycan N-Glycan-G00020 is a predecessor of N-Glycan-G00022. These entities are semantically associated, even though there is no direct path connecting them. In fact, the entire pathway links the starting N-Glycan-G00020 substance and the final product, N-Glycan-G00020, using a sequence of reactions returned by the request. This query locates a pathway of length 8, consisting of several reactions contained in the GlycO-0-9-full-enzyo dataset. This test's results demonstrated the effectiveness of the proposed PmSPARQL language.

VIII. Conclusion

This paper addresses the issue of providing support for association identification and evaluation queries in RDF databases. For this task, we have presented PmSPARQL, an extension of SPARQL language designed for finding associations with several paradigms dependant to user needs, and describes its syntaxic and semantic implementation. The inclusion of path pattern queries in our experimentations has demonstrated the powerful results of PmSPARQL queries, its effectiveness for finding results in a reasonable time execution. In addition to finding long associations, our implementations demonstrate the importance of the evaluation measure to rank associations according to some new defined measures (richness, connectivity, importance,...). For the reason that the timing for query results is good, we think to optimize the PmSPARQL queries for path extraction which is very important for practical use of this language. The presented measures for associations evaluation are important, because the same query with different measures should returned different results and consequently should provided a different association classification.

References

[1] B. Aleman-Meza, C. Halaschek, B. Arpinar, and A. Sheth, Context-aware semantic association ranking. In First International Workshop on Semantic Web and Databases,in *First International Workshop on Semantic Web and Databases*, Berlin, Germany, September 2003, p. 33–50.

[2] K. Anyanwu, A. Maduko, and A. Sheth, Sem-rank: Ranking complex relationship search results on the semantic web, in *International World Wide Web Conference*, *14*, ACM, Chiba, Japan, 2005, p. 117–127.

[3] B. Aleman-Meza, C. Halaschek-Wiener, I. Budak Arpinar, C. Ramakrishnan, and A. Sheth, Ranking complex relationships on the semantic web, in *IEEE Internet Computing*, *03*, 2005, p. 37–44.

[4] B. Aleman-Meza, P. Burns, M. Eavenson, D. Palaniswami, and A. Sheth, An ontological approach to the document access problem of insider threat. *IEEE International Conference on Intelligence and Security Informatics*. Atlanta, Georgia, USA, 2005, p. 486–491.

[5] K. Anyanwu and A. Sheth, The rho operator: Computing and ranking semantic associations in the semantic web. SIGMOD Record, 2002.

[6] A. Seaborne, *RDQL A Query Language for RDF, WWWConsortium, Member Submission SUBM-RDQL-20040109*, 2004.

[7] E. Prud'hommeaux and A. Seaborne, *SPARQL:Query Language for RDF*, 2005.

[8] A. Kemafor, M. Angela, and S. Amit, SPARQ2L: Towards support for subgraph extraction queries in rdf databases, in *WWW 2007*, Banff, Alberta, Canada, 2007, p. 797– 806.

[9] J. Krys and Maciej.J, SARQLeR: Extended sparql for semantic association discovery, in *4 th European Semantic Web Conference.* Innsbruck, Austria, 2007.

[10] A. Helenius and M. Aebi, Roles of n-linked glycans in the endoplasmic reticulum, in *Annual Review of Biochemistry*, *73* , 2004, p. 1019–1049.

[11] H. Donninger, T. Bonome, M. Radonovich, Pise-Masison, C. A., J. H. Brady, J.and Shih, J. Barrett, and M. J. Birrer, Whole genome expression profiling of advance stage papillary serous ovarian cancer reveals activated pathways. Oncogene **23**, 8065, 8077 (2004).

[12] T. Miki, S. Nomura, and T. Ishida, Semantic web link analysis to discover social relationships in academic communities. *Symposium on Applications and the Internet*, 2005.

[13] A. Sheth, B. Aleman-Meza, I. Arpinar1, C. Halaschek, C. Ramakrishnan1, C. Bertram, Y. Warke, D. Avant, F. S. Arpinar, K. Anyanwu, and K. K., Semantic association identification and knowledge discovery for national security applications. Special Issue of Journal of Database Management on Database Technology for Enhancing National Security, L. Zhou and W. Kim (Eds.) **16**, 33-53 (2005).
[14] S. Mukherjea and B. Bamba, Biopatentminer: An information retrieval system for biomedical patents. *Thirtieth International Conference on Very Large Data Bases.* VLDB, Toronto, Canada, 2004, p. 1066–1077.
[15] I. Arpinar, A. Sheth, C. Ramakrishnan, E. Usery, M. Azami, and M. Kwan, Geospatial ontology development and semantic analytics, in *Handbook of Geographic Information Science., 4*, edited by J. P. Wilson and A. S. F. E. vol 10. Blackwell Publishing, 2004.
[16] S. Lin and H. Chalupsky, Unsupervised link discovery in multirelational data via rarity analysis. *ICDM 2003*, 2003, p. 171–178.
[17] M. Janik and K. Kochut, A work-bench rdf store and high performance memory system for semantic association discovery. *4th International Semantic Web Conference.* Galway, Ireland, 2005.
[18] W. Milnor, C. Ramakrishnan, M. Perry, A. Sheth, J. Miller, and K. Kochut, Discovering informative subgraphs in rdf graphs. Technical report, LSDIS Lab, Computer Science,University of Georgia, CS Technical Report 05-001.
[19] V. Paliwal, N. R. Adam, H. Xiong, and C. Bornhovd, Web service discovery via semantic association ranking and hyperclique pattern discovery, in *wi, IEEE/WIC/ACM* ,IEEE Computer Society, 2006, p. 649–652.
[20] H.-J. Chu and R. Chow, Reaching semantic interoperability through semantic association of domain standards, in *11th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS07), ISSN:1701-0483, 0-7695-2810-4*, IEEE Computer Society, Washington, DC, USA, 2007.
[21] I. Cruz, A. Mendelzon, and P. Wood, A graphical query language supporting recursion. in acm sigmod international conference on management of data, in *ACM SIGMOD International Conference on Management of Data*, San Francisco, California, United States, 1987, p. 323–330.
[22] I. Cruz, A. Mendelzon, and P. Wood, G+: Recursive queries without recursion. *2nd International Conference on Expert Database Systems*, 1988, p. 355–368.
[23] M. Consens and A. Mendelzon, Graphlog: a visual formalism for real life recursion. *ACM Symposium On Principles of Database Systems.* 1990, p. 404–416.
[24] J. Broekstra and A. Kampman, SERQL: A second generation rdf query language. In *SWAD-Europe Workshop on Semantic Web Storage and Retrieval.* SWAD-Europe Workshop on Semantic Web Storage and Retrieval, 2003.
[25] M. Sintek and S. Decker, Triple - an rdf query, inference, and transformation language. In *Deductive Databases and Knowledge Management.* Tokyo, Japan, 2001.
[26] U. Ogbuji, *RDF Query using Versa Thinking XML: Basic XML and RDF techniques for knowledge management, Part 6*, 10 April 2002.
[27] A.Souzis, *RxPath specification proposal. http://rx4rdf. liminalzone.org/RxPathSpec.*, 2004.
[28] L. Sam, L. yang, L. Jianrong, C. Friedman, and Y. Lussier, Triple - an rdf query, inference, and transformation language. In *12me Pacific Symposium on Biocomputing.* 2007, p. 76-87.
[29] T. Samir and I. Budak Arpinar, Ontology evaluation and ranking using ontoqa. *The first IEEE International Conference on Semantic Computing.* Irvine, California, USA, September 17-19, 2007, p. 185-192.
[30] C. Gutierrez, C. Hurtado, and A. Mendelzon., Foundations of Semantic Web Databases. *Foundations of Semantic Web Databases. In PODS 2004, p. 95106.*, 2004.
[31] D. Marin, Rdf formalization. Technical report, Santiago de Chile. TR/DCC-2006-8. http://www.dcc.uchile.cl/ cgutierr/ftp/draltan.pdf.
[32] P. Jorge, A. Marcelo, and G. Claudio, Semantics and complexity of sparql. *International Semantic Web Conference.* Athens, GA, US, 2006.
[33] J. Lim, *ADOdb Library for PHP, http://php.weblogs.com/ADODB.*, 2007.

Thabet Slimani Received his Master degree in Computer Sciences from the High School of Management (HIM), University of Tunisia, Tunisia 2003. He is currently member of LARODEC Laboratory and a PHD student at the same school working on Semantic Web. His research interest Data Mining: Association Rule Mining, Data Warehouse, Semantic Associations, Similarity measures and ontology alignment/merging.

Boutheina Ben Yaghlane Doctor in Computer Science Applied to Management. She is currently an assistant Professor at the Institute of High Commercial studies (IHEC Carthage) and member of LARODEC Laboratory. Her research interests include, Uncertainty Management, Semantic Web, Data Mining and learning under uncertainty. She has published research papers at international journals and conference proceedings.

Khaled Mellouli Received his PHD in management from the School of Business, University of Kansas USA. He is currently a full professor at the Institute of High Commercial studies (IHEC Carthage) and he's the director of LARODEC Laboratory of HIM institute (Tunisia University). His research interests are related to Uncertainty Management, Decision problem analysis, Semantic Web, classification, learning under uncertainty and operational research and their application. He is author of a great deal of research studies published at national and international journals, conference proceedings as well as chapters of books.