Ranked Answer Graph Construction for Keyword Queries on RDF Graphs without Distance Neighbourhood Restriction

Parthasarathy K
Indian Space Research Organisation
Bangalore, India
parthas@isro.gov.in

Sreenivasa Kumar P
Indian Institute of Technology Madras
Chennai, India
psk@cse.iitm.ac.in

Dominic Damien
Indian Space Research Organisation
Bangalore, India
dominic@isro.gov.in

ABSTRACT

RDF and RDFS have recently become very popular as frameworks for representing data and meta-data in form of a domain description, respectively. RDF data can also be thought of as graph data. In this paper, we focus on keyword-based querying of RDF data. In the existing approaches for answering such keyword queries, keywords are mapped to nodes in the graph and their neighborhoods are explored to extract subgraph(s) of the data graph that contain(s) information relevant to the query. In order to restrict the computational effort, a fixed distance bound is used to define the neighborhoods of nodes. In this paper we present an elegant algorithm for keyword query processing on RDF data that does not assume such a fixed bound. The approach adopts a pruned exploration mechanism where closely related nodes are identified, subgraphs are pruned and joined using suitable hook nodes. The system dynamically manages the distance depending on the closeness between the keywords. The working of the algorithm is illustrated using a fragment of AIFB institute data represented as an RDF graph.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms

Keywords RDF, RDFS, Keyword search, Answer Graph

1. INTRODUCTION

Query processing over graph data has attracted considerable attention recently as an increasing amount of data which is available on the web, XML data sources and relational sources can be modeled in the form of graphs. RDF as a framework for web resource description appears to have gained a greater momentum on the web and an increasing collection of repositories of data are modeled using RDF framework. Notable examples are biological databases[1], Personal Information Systems[14] where emails, documents and photos are merged into a graph and enterprise information management (EIM) systems like launch vehicle design data where details about vehicle stages, parameters and stage sequence events is modeled as graph data. The largeness and complexity of data sets in these domains makes their querying a challenging task. One interesting class of queries relevant in these data sets is *relatedness queries* i.e to identify a chain of relationship between graph elements. The following query describe typical scenarios where finding such relationships will be relevant.

- *Find the sequence of events in the launch vehicle events chain, between any two major events, in the second stage of a launch.*

Keyword queries provide an easy mechanism for querying underlying data repository. They do not require the users to know complex query language or know details regarding the underlying schema. In *relatedness queries*, the keywords corresponds to anchor points in the data graph. We need to extract a subgraph connecting these anchor points.

Much work has been carried out on keyword search on relational data [4, 13, 3], tree structured data[8] and recently on graph structured data [12, 10, 11, 9]. In order to limit the traversal during graph exploration step, [12, 9] fixes a distance d to restrict the exploration for finding nodes related to the keywords of the query. Within this *d-neighbourhood* of entities, possible subgraphs which connect the entities are computed. The following points highlight two major issues related to this approach.

- Fixing the *distance* d is to restrict exploration space and also to justify the fact that smaller distances are likely to contain the missing links the user looks for. But an optimal d covering keywords is difficult to arrive at. If d happens to be small then the graph constructed becomes disconnected and chained relationship links cannot be extracted. The user has to experiment with different d for exploring the graph.

[1]BioCyc(http://biocyc.org)

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0637-9/11/03.

- In the example illustrated in [12], the keywords *Philip X-Media publication* are spaced evenly. Given the set of keywords, some keywords could be closer and some could be farther(from a graph perspective). The distance between the pair of keywords is not exploited during the graph exploration phase.

In this paper we propose a novel approach for answer graph construction to keyword queries on RDF data represented as a graph, specifically addressing the above mentioned issues. The approach does not fix any distance before hand to identify the vicinity of exploration. We have extended our earlier approach presented in [6] by suitably modifying the pruning step of the algorithm to exploit the distance information between the mapped nodes. For each mapped node or edge for the keyword, neighbourhood class clusters are formed. Nodes from the clusters which cannot contribute to the overall answer graph for the query, are removed. If the keyword clusters do not have class nodes in common implying that keywords are not too close, the pruning step adds a chain of class nodes for processing. Intuitively the clusters are enlarged selectively based on the individual distance between the keywords. The algorithm then explores the new set of clusters to find suitable hook elements for joining and then builds the answer graph for the keyword query.

The paper is organized as follows. Section 2 describes the preliminaries of the problem. Section 3 presents the algorithm, Section 4 presents the illustration of the working of the algorithm, Section 5 presents ranking, Section 6 presents related works and Section 7 presents future works and conclusions.

2. DEFINITION

Given the directed data graph G of an RDF data set containing triples, we are concerned with querying the graph using keywords.

Data Graph The data graph G = (N,E) where

- N is a finite set of nodes which is a disjoint union of *C-Nodes(representing classes), EN-Nodes(representing entities) and D-Nodes(data values) i.e N=C-Nodes* $\uplus$ *EN-Nodes* $\uplus$ *D-Nodes.* In the RDF fragment shown in Figure 1., *Person, Organisation* are examples of *C-Nodes*, *AIFB, Efficient Algorithms* are examples of *D-Nodes* and *pub1, proj1* are examples of *EN-Nodes.* EN-nodes will not be used for queries as they are internal to the RDF graph.

- E is the finite set of edges connecting nodes n_1,n_2 with n_1, n_2 ϵ N. The different types of edges are *IE-Edges* (inter-entity edges), *EA-Edges* (entity-attribute edges) and *Class/SubClass* edges. In *Figure 1 Author, CarriesOut are examples of IE-Edges and Title, Journal are examples of EA-Edges.*

- L is a labeling function which associates a label l for an edge. $L = L(IE\text{-}Edges) \uplus L(EA\text{-}Edges) \uplus \{Class, SubClass\}$ *where L(IE-Edges) represents labels for inter-entity edges, L(EA-Edges) represents labels for entity-attribute edges.* The following restrictions apply on l:
  - $l \in L(IE\text{-}Edges)$ *if and only if* n_1,$n_2 \in$ *EN-Nodes*
  - $l \in L(EA\text{-}Edges)$ *if and only if* $n_1 \in$ *EN-Nodes and* $n_2 \in$ *D-Nodes*

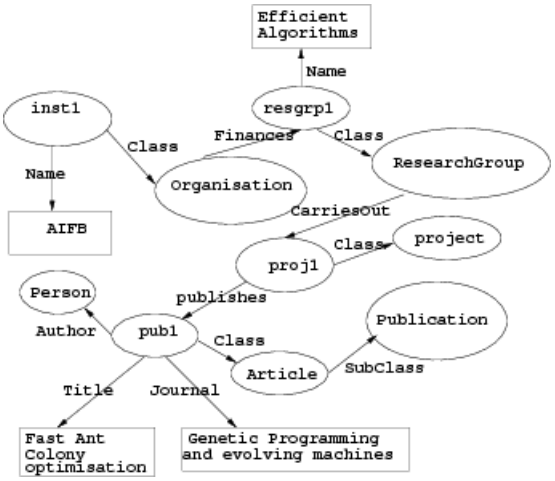

Figure 1: RDF Graph fragment from AIFB Institute Data

  - l = *SubClass if and only if* n_1,$n_2 \in$ *C-Nodes*
  - l = *Class if and only if* $n_1 \in$ *EN-Nodes and* $n_2 \in$ *C-Nodes.*

Class and *SubClass* are two predefined type of edges which captures class membership of an entity and class hierarchy. In Section 3 we use a term *CR-Node* for algorithm description. This is defined by the following property:

CR-Node is a C-Node which has an inter-entity relationship with another C-Node. For example the C-Node Researchgroup is a CR-Node for the C-Node Organisation and vice-versa.

Figure 1 shows a fragment of RDF graph containing data taken from AIFB institute, University of Karlsruhe[2] which will be used for illustration of the algorithm in this paper. The fragment models the information *related to financing of Research group by Organisation AIFB, Projects carried out by the Research group and publications from those projects.* This fragment models a chain of relationship.

Queries A keyword search query Q consists of a list of keywords $\{k_1, \ldots k_n\}$. Given this list of keywords the answer graph to the query is a minimal possible subgraph A such that

- Every keyword is contained in at least one node or edge in G
- The graph consists only of keyword nodes, class nodes connected by inter-entity and entity-attribute edges. The edge labels also form part of the answer graph
- Answer A is minimal in the sense that no subgraph of A can be an answer to Q. If keyword node is removed then that keyword is not matched. If a non keyword node is removed the graph becomes disconnected.

It may be noted that the answer graph is not a result set to the keyword query, but can be used to generate structured queries using query frameworks like SQL and SPARQL.

[2]http://km.aifbunikarsruhe.de/ws/eon2006/ontoeval.zip

Problem We focus on efficient algorithm for construction of ranked answer graphs to the keyword query on RDF data represented as a graph, that exploits graph semantics(subclass relationship) and does not impose distance neighbourhood restriction.

3. ALGORITHM DESCRIPTION

The term mapping step maps each keyword k_i to a set of nodes/edges of the graph corresponding to the data model. By taking one mapped node for each keyword, a node list *NL* is formed which is an input for answer graph construction. For each node list *NL,* an answer graph will be constructed in three steps, namely, *Component_Cluster_Creation, Pruning and Hooking.*

Component_Cluster_Creation Initially the component cluster for each node is empty. A node from NL is taken and the node type is identified. If the node is *C-Node* , then the *C-Node*, its *CR-Nodes* along with the edge labels are added. If the node is a *D-node* then the *C-Node* for the class to which the entity of this *D-node* is associated and *CR-Node*s for the *C-Node* along with the edge labels are added. If the node is mapped to an *IE-Edge,*the corresponding *C-Nodes* are added and if it mapped to *EA-Edge,* a dummy node for the attribute side, *C-Node* for the class to which the entity is associated and *CR-Node*s for the *C-Node* are added. The component clusters obtained for all nodes in NL will act as input to the *pruning* step.

Pruning In this step the algorithm prunes the loosely hanging nodes which possibly cannot be utilised for hooking. For each pairwise component cluster, common nodes are identified. We use the term *similiar nodes* to refer common nodes. Two nodes are *similiar* if they satisfy any one of the following property:

1. They are the same nodes in the graph
2. The nodes are related by SubClass relationship
3. There exists a chain of intermediate *C-Nodes* which connects the two nodes.

In [6], we make use of *property 2* for handling queries. If the intersection of the node list pair is empty, it indicates that the nodes chosen for the keywords are not close neighbours and we enlarge the clusters by using the C-Node chain by using *property 3.* If multiple chains exists, the chain with the least number of *C-Nodes* is used. In this the left half from the middle element in the chain is added to one cluster and right half to other cluster with centre *C-Node* added to both. The union of all the similiar nodes found by considering all pairs of component structures that represents the participating node list. The nodes to be pruned is the compliment of the participating node list with respect to the full node list. The pruned nodes along with its incident edges is removed from the corresponding component cluster. The new list of component clusters obtained will be the input to *hooking* step.

Hooking In this step we start to explore whether *C-Node or CR-node* of a component cluster can be hooked on to a similiar *CR-Node/C-Node* of another component cluster. Initially any component cluster with lowest cardinality of *C-Nodes/CR-Nodes* is chosen for starting the hooking operations. The property of similiar nodes defined earlier is also used for hooking operation. Once nodes to be hooked are identified, the corresponding component clusters are glued together. Nodes which are duplicates are removed and a new glued component cluster is used for further hooking operations. This process continues until no more nodes could be hooked. The final component cluster arrived at is analyzed for loosely hanging nodes and they are cut off. The closely connected cluster thus formed will be the Answer Graph for the keywords presented by the user.

4. ILLUSTRATION OF THE ALGORITHM

Keyword Query: *AIFB journal titles*

- Keyword *AIFB* will be mapped on to the term *AIFB*
- Keyword *journal* will be mapped on to the term *journal*
- Keyword *titles* will be mapped on to the term *title*

In the Component_Cluster_Creation step, using the class nodes and relationship nodes associated with the terms, the component clusters as shown in *Figure 2 and Figure 3* is constructed. Since *AIFB* is the name of an organisation, the *C-Node Organisation* along with *CR-node ResearchGroup* is added. For the term *journal,* which it is mapped to an entity-attribute edge, a *dummy journalname node along with C-Node Article and CR-Nodes Person and Publication are added to form its component cluster.* For the term *title* which is again mapped to an entity-attribute edge, a *dummy titlename node along with Class Nodes Article, Misc, InProceedings, InCollection and CR-Nodes Person and Publication are added to form its component cluster as shown in Figure 3.*

In pruning step, we take the pairwise intersection of the nodes of the components. For the query, the component cluster for *title* and *journal* provide the intersection as shown in first figure of *Figure 4.* The component cluster of *journal* is a subcluster of component cluster of *title* except for the dummy node added. Both these clusters are merged by pushing the extra dummy node to the smaller subcluster. The intersection of clusters of {*title*}, {*AIFB*} and {*journal*}, {*AIFB*} is empty. The algorithm uses the property of similiar nodes to find a chain of *C-Nodes* connected by edges. For the query, the node *ResearchGroup* and *Publication* are connected by the chain as shown in *Figure 4b.* The left set of nodes which is empty is added to AIFB cluster and the right set {article} is added to cluster {journal, title}. The centre node *Project* is attached to both the clusters. We are left with two clusters corresponding to *AIFB and journal* as shown in *Figure 5.*

In the hooking step, the *Project* node of *AIFB* cluster is hooked with the *Project* node of *Journal* cluster and merged. There is no more component to be considered and the final merged cluster will be the answer graph as shown in *Figure 6.*

Compared to earlier approaches for graph querying and also the the approach adopted in [12] our approach has the following enhancements

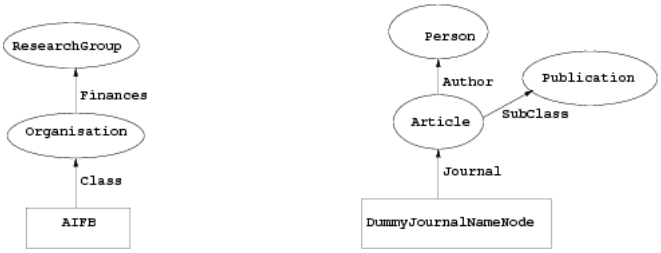

Figure 2: Component Cluster for *AIFB and Journal*

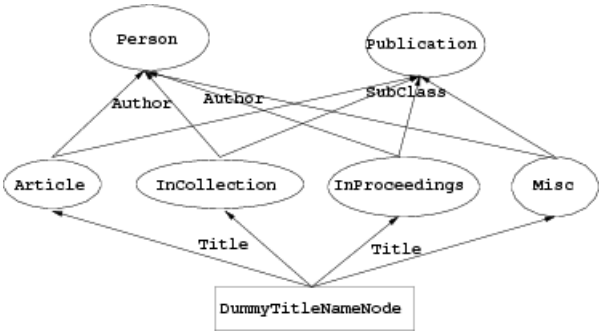

Figure 3: Component Cluster for the term *title*

- Edge mapping is allowed
- Exploits graph semantics(Class/SubClass relationship) during the graph exploration phase
- Distance neighbourhood is not fixed but managed during the pruning phase.

5. RANKING

Since multiple answer graphs could be constructed either through different *term-node* association or through different hook elements, there is a need to meaningfully rank them to identify top answers. Our approach is similiar to [9] where we have used structural compactness as the criteria. We have also added two more factors i.e relationship relevance and node type relevance into our ranking model to align with our approach and also to the RDF graph model.

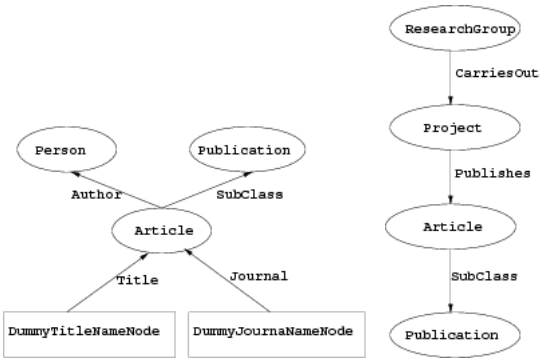

Figure 4: Component clusters formed during the pruning phase

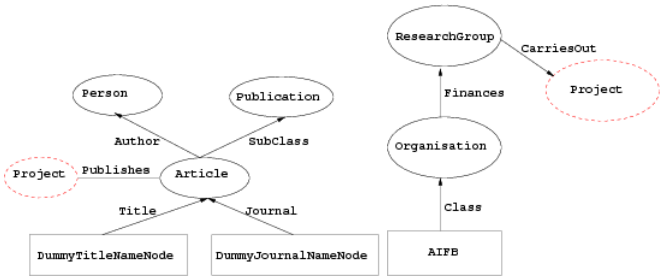

Figure 5: Final Component clusters for hooking step {*Journal, Title*},{*AIFB*}

Figure 6: Final answer graph for the query

Compactness Relevance From a graph perspective, compact answers should be preferred. When the length of the C-Node chain between the mapped nodes is larger, the compactness between them is smaller. If $n_i$and n_j are mapped nodes corresponding to keywords $k_i$and k_j, we define structural compactness for the answer graph AG as follows

$$SC(k_i,k_j|\ AG) = \frac{1}{(chnl+1)^2} \text{ if } n_i\text{and } n_j \text{ are same node}$$
$$= \frac{1}{2(chnl+1)^2} \text{ if } n_i\text{and } n_j \text{ are different nodes}$$

where chnl = number of C-Nodes in the minimum length chain + 1

Term Relevance The textual relevancy of the keywords with reference to the nodes mapped is one of the important attribute for ranking. For example, the term *AIFB* gets mapped to node with project name *AIFB* or it gets mapped to title of a publication which has the string *AIFB*. The term affinity will be different for these mapping from IR perspective. For each keyword element, a matching score TR using standard IR approach can be computed. Combining compactness relevance and term relevance, we have the following

$$RANKVAL(k_i,k_j|\ AG) = SC(k_i,k_j|\ AG) * ((TR(k_i|\ AG) + (TR(k_j|\ AG))$$

Node Relevance The node/edge category to which the keyword is associated also plays a prominent role for ranking. For example the keyword, *publications* gets

mapped to the *C-Node Publication* or to *D-Node abstract* containing the string *publication*. *C-Nodes/IE-Edges* will have highest scores followed by *EA-Edges* followed by *D-Nodes*. The node relevance scores NR for all the mapped nodes is computed.

Relationship Relevance Since the fundamental approach to answer graph construction is identification of missing interconnection nodes, the neighbouring *C-Nodes* that contributes to the answer graph is an important parameter for ranking. It is computed as follows

$$\mathrm{RR} = \frac{Number\ of\ C-Nodes\ in\ AG}{Total\ Number\ of\ C-Nodes} - \frac{Number\ of\ C-Nodes\ added\ AG}{Total\ Number\ of\ C-Nodes}$$

The overall RankVal of an answer graph AG is calculated as follows

$$\mathrm{RankVal(AG)} = \sum_{1 \le i \le j \le n} \mathrm{RankVal(k}_i\mathrm{,k}_j\mathrm{|\ AG)} + \mathrm{NR(AG)} + \mathrm{RR(AG)}$$

6. RELATED WORK

In [12], a generic graph based approach is presented to explore the connections between terms mapped to keywords of the query using knowledge available in ontology. The process of exploration relies mainly on assertional knowledge resulting in a large number of paths that need to be processed. The graph does not model class/sub-class forms of relationship. Ranking is based only on on path length. In [9], the data set is implemented as graphs and the search is modeled as *r-Radius Steiner Graph problem i.e to identify all all r-radius Steiner graphs which contain all the keywords.* In this paper also radius is first fixed. In [10], a system SPARK is proposed for adapting keyword query to semantic search. The system considers all possible term mappings and no pruning is done during query graph construction.

In our approach, we create a fragment of closely related concept clusters and then prune unwanted nodes and edges. We also adopt a guided exploration strategy which exploits other knowledge properties(class/subclass relationship). Also the distance factor is implicitly managed in the pruning phase depending on the closeness of the neighbouring concept nodes found out during the component cluster creation phase. We have also presented a ranking scheme for the answer graphs in line with our approach.

7. CONCLUSIONS AND FUTURE WORK

We have presented an algorithm and illustration for answer graph construction, given a set of keywords and a knowledge repository represented as an RDF graph. It is planned to implement the algorithm on a standard data set. It is also planned to extend the scope of the work to address the following challenging issues

- Specific implementation mechanisms for our algorithmic approach
- Develop efficient indexing techniques to aid the graph exploration
- Handling large RDF graphs by this approach using graph partitioning scheme.

8. REFERENCES

[1] Allemang, D., and Hendler, J. *Semantic Web for the Working Ontologist Modeling in RDF, RDFS and OWL*. Morgan Kaufmann Publishers, Reading, Massachusetts, 2008.

[2] B.Kimfield, and Y.Sagir. Finding and approximating top-k answers in keyword proximity search. In *PODS 2006* (2006), ACM, pp. 173–182.

[3] G.Bhalotia, A.Hulgeri, C.Nakhe, S.Chakraborti, and S.Sudharshan. Keyword searching and browsing in database using banks. In *ICDE 2002* (2002), ACM, pp. 431–440.

[4] H.He, H.Wang, J.Yang, and P.S.Yu. Blinks: Ranked keyword searches on graphs. In *SIGMOD Conference 2007* (2007), ACM, pp. 305–316.

[5] Kasneci, G., Ramanath, M., Sozio, M., Suchanek, F., and Weikum, G. Star: Steiner tree approximation in relationship graphs. In *25th IEEE International Conference on Data Engineering, ICDE 2009* (2009), IEEE, pp. 868–879.

[6] K.ParthasarathY, P.SreenivasaKumar, and Damien, D. Answer graph construction for keyword search on graph structured (rdf) data. In *International Conference on Knowledge Discovery and Information Retrieval(KDIR) 2010* (Oct 2010), INSTICC.

[7] Lei.Y, Uren.V, and Molta.E. Semsearch: A search engine for the semantic web. In *15th International Conference on Knowledge Engineering and Knowledge Management(EKAW),(2006)* (2006), pp. 238–245.

[8] L.Guo, F.Shao, C.Botev, and J.Shanmugasundaram. Xrank: Ranked keyword search over xml documents. In *SIGMOD Conference 2003* (2003), ACM, pp. 16–27.

[9] Li, G., Ooi, B. C., Feng, J., Wang, J., and Zhou, L. Ease: An effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In *SIGMOD 2008* (2008), ACM, pp. 1452–1455.

[10] Q.Zhou, C.Wang, M.Xiong, H.Wang, and Y.Yu. Spark: Adapting keyword query to semantic search. In *ISWC/ASWC,2007* (2007), SWSA, pp. 694–707.

[11] Revuri, S., Upadhyaya, S., and P.SreenivasaKumar. Using domain ontologies for efficient information retrieval. In *International Conference on Management of Data COMAD 2006* (Dec 2006), CSI, pp. 84–89.

[12] T.Tran, P.Camiano, S.Rudolph, and R.Studer. Ontology based interpretation of keywords for semantic search. In *ISWC/ASWC,2007* (2007), SWSA, pp. 523–536.

[13] V.Kacholia, S.Pandit, S.Chakraborti, S.Sudharshan, R.Desai, and H.Karambelkar. Bidirectional expansion for keyword search on graph databases. In *VLDB 2005* (2005), VLDB, pp. 505–516.

[14] Y.Cai, X.Dong, A.Halevy, J.Liu, and J.Madhavan. Personal information management with semex. In *SIGMOD 2005* (2005), ACM, pp. 921–923.

[15] Y.Sure, S.Bloehdorn, P.Haase, J.Hartmann, and D.Oberle. The swrc ontology - semantic web for research communities. In *In Proceedings of the 12th Portuguese Conference on AI(EPIA 2005)* (2005), ECCAI, pp. 218–231.