ANSWER GRAPH CONSTRUCTION FOR KEYWORD SEARCH ON GRAPH STRUCTURED(RDF) DATA

K. Parthasarathy*
Indian Space Research Organisation, Bangalore, India

P. Sreenivasa Kumar
Department of Computer Science and Eng., Indian Institute of Technology Madras, Chennai, India

Dominic Damien
Indian Space Research Organisation, Bangalore, India

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

Keywords: Keyword search, Answer graph, RDF.

Abstract: Keyword search is an easy way to allow inexperienced users to query an information system. It does not need knowledge of specific query language or underlying schema. Recently answering keyword queries on graph structured data has emerged as an important research topic. Many efforts focus on queries on RDF(Resource Description Framework) graphs as RDF has emerged as a viable data model for representing/integrating semi-structured, distributed and interconnected data. In this paper, we present a novel algorithm for constructing answer graphs using pruned exploration strategy. We form component structures comprising closely related class and relationship nodes for the keywords and join the identified component structures using appropriate hook nodes. The Class/SubClass relationships available in RDF schema are also utilized for the answer graph construction. The paper illustrates the working of the algorithm using AIFB institute data.

1 INTRODUCTION

Query processing over graph based data has attracted considerable attention recently as increasing amount of data which is available in web, XML and DBMS can be modeled in the form of a graph. RDF being a framework for web resource description appears to have a greater momentum on the web and an increasing collection of repositories of data are modeled using RDF framework. Notable examples are biological and chemical databases, Web-scattered data, healthcare, Personal Information Systems(where emails. documents and photos are merged into a graph using connection elements) and enterprise information management (EIM) systems like launch vehicle design data where details about vehicle stages, parameters and stage sequence events are modeled. In this class of applications raw data may not be graph structured at the first level but implicit connections will provide a graph structure. The largeness and complexity of datasets in these domains make their querying a challenging task.

As keyword queries do not require the users to know complex query language or know details regarding the underlying schema much work has been carried out on keyword search on databases(Bhalotia et al., 2002; He et al., 2007; Kacholia et al., 2005), tree structured data(Guo et al., 2003) and recently on graph structured data(Tran et al., 2007; Zhou et al., 2007; Revuri et al., 2006). A generic approach first identifies parts of graph containing the keywords of interest, explores for discovering possible interconnections between identified parts. Candidate solutions built out of the connections found are scored and ranked.

Following are some of the specific issues related to the approaches adopted both by existing database and graph based keyword search methods:

- The methods explore all possible paths. Unimportant results are logically pruned by means of scor-

*External Research Scholar, CSE Dept, IIT Madras.

Parthasarathy K., Sreenivasa Kumar P. and Damien D..
ANSWER GRAPH CONSTRUCTION FOR KEYWORD SEARCH ON GRAPH STRUCTURED(RDF) DATA.
DOI: 10.5220/0003067001620167
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2010), pages 162-167
ISBN: 978-989-8425-28-7
Copyright © 2010 SCITEPRESS (Science and Technology Publications, Lda.)

ing and ranking mechanism. They do not attempt to prune the search space in the earlier exploration phase of the graph search.

- Class/Sub-Class and other relations available in RDFS framework are not being exploited suitably for keyword queries. For example in the RDF repository of AIFB institute, if the query {*person, aifb, publication*} is submitted and if the interpretation is to retrieve all *publications* from *persons* related to the institute *aifb*, exploration scheme described in (Tran et al., 2007) will not function properly if some of the RDF instance data related to a publication belongs to a class *TechnicalReport* which is a subclass of *Publication.*
- As input RDF graph becomes bigger, traversing through all the nodes in the exploration phase for all possible paths as adopted in (Tran et al., 2007) will lead to performance issues.

In this paper we propose a novel approach to keyword search in the graph structured data represented as RDF. The approach attempts to provide possible set of answers in an efficient way. For each mapped node or edge for the keyword the closest concept or set of concepts and relationship nodes are identified to form a keyword cluster. Using a pruning strategy nodes from the clusters which cannot contribute to the overall structure of the query are removed. The modified set of clusters are explored to find suitable hook elements for joining and constructing the answer graph for the keyword query.

The paper is organized as follows. Section 2 describes the preliminaries of the problem, Section 3 describes the overall approach and algorithm description, Section 4 presents the illustration, Section 5 presents the evaluation, Section 6 presents related works and Section 7 presents conclusion and future works.

2 PROBLEM DEFINITION

Given a data graph G of an RDF data model, we are concerned with querying the graph using keywords.

Definition 1. A data graph G is a tuple of the form (N,R,E) where

- N is a finite set of nodes which can represent *classes, entities and data values.* Nodes representing entities are referred by specific IDs as shown in Figure 1. These IDs will not be used for queries as they are internal to the RDF graph.

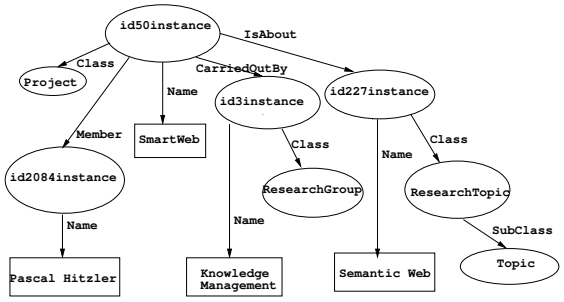


Figure 1: RDF graph fragment from AIFB Institute Data.

- R is the set of edge labels for edges which can represent inter-entity edges, entity-attribute edges or class/sub-class edges.
- E is the finite set of labeled edges of the form e(n_1,n_2) with n_1, n_2 ε N and e ε R.

Class and Subclass are two predefined type of special edges which capture class membership and class hierarchy. *Figure 1* shows a fragment of RDF graph containing data taken from AIFB institute(University of Karlsruhe) which is also the data set used for illustration of the algorithm in this paper. The fragment models the information that *Pascal Hitzler is a member of a Project called SmartWeb carried out by Knowledge Management ResearchGroup on ResearchTopic Semantic Web.*

Queries. A keyword search query Q consists of a list of keywords {k_1,...k_n}. Given this list of keywords the answer graph to the query is the set of minimal possible subgraph A s.t

- Every keyword is contained in at least one node or edge in A.
- 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 present anywhere else in A. 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 only provides a view of the bonding structure of the keywords with respect to the RDF repository. The answer graph could be used to provide more insight into the schema or can be used to generate structured queries using query frameworks like SQL and SPARQL.

Problem. We are concerned with the construction of set of possible answer graphs to the query using RDF data repository represented as a graph.

Assumption. We assume that the mapped elements for the keywords are available for the algorithmic step to proceed with the construction of the answer graph. Mapping refers to a process where a keyword is lexically analyzed and a list of graph elements having labels that are syntactically similiar to the keyword is returned.

3 ALGORITHM APPROACH AND DESCRIPTION

Answer Graph Generation. The terms from the term mapping stage are mapped to a set of nodes/edges of the graph corresponding to the data model. By taking one nodal element for each term a nodelist *NL* is formed which is an input for answer graph construction. In order to generate an answer graph using the mapped elements of the nodelist in the graph, each mapped element is used to construct a closely related cluster consisting of the element class category and list of relationship nodes along with the edges and edge labels(*component structure creation step*). In the next step the algorithm attempts to prune the loosely hanging nodes which cannot possibly be hooked to any other nodes in the cluster(*pruning step*). In the next step which does the hooking operation the algorithm incrementally constructs answer graph by finding suitable hooking elements between the clusters(*hooking step*). The steps mentioned maintains the following class of nodes

- *C-Node (Class Node)* - Node which corresponds to a Class Category
- *D-Node (Data Value Node)* - Nodes which represents data values
- *CR- Node (Relationship Nodes)* - Class Nodes which are connected by inter-entity edges

Component Structure Creation. In this step each node from NL is taken and the characterstic of the node is found. If the node is *a C-Node*, then the inter-entity edges along with the edge labels are added to the component structure which will be initially empty. If the node is a *D-node,* then the *C-Node* for which 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 inter-entity edge, then the corresponding *C-Nodes* are added and if it is mapped to an entity-attribute edge, then a dummy node for the attribute side, *C-Node* for the class to which entity is associated and *CR-Nodes* for the *C-Node* are added. The component structures 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 structure, common nodes which are *similiar* are found. This is found by the intersection of the nodelist pair. Two nodes are considered *similiar* if they are the same node in the graph or they are connected by means of *Class/SubClass* relationship. This concept of *similiarity* can be extended for other scenarios e.g when the nodes are connected by means of a chain of intermediate nodes.The union of all the common nodes found by considering all pairs of component structure represents the participating nodelist. The nodes to be pruned is the complement node list of the participating node list with respect to the full node list. The pruned node along with the edge label to which is connected is removed from the corresponding component structure. The new list of component structures obtained will act as input to *hooking* step.

Hooking. In this step we start to explore whether *CR-node* of a component structure can be hooked on to a similiar *CR-Node or C-Node* of another component structure. Initially a component structure with lowest cardinality is chosen for starting the hooking operations. If there are multiple choices, the component structure in which there is a *CR-Node* which can be hooked to *C-Node* of another component structure is chosen for start of hook operations. The hooking step uses the *node similiarity* feature as defined in pruining step for hook operations. Once nodes to be hooked are identified, the corresponding component structures are glued together. Nodes which are duplicates are removed and a new glued component structure is used for further hooking operations. This process continues until no more nodes could be hooked. The final component structure arrived at is analysed for loosely hanging nodes and they are cut off. The closely connected structure thus formed will be the answer graph for the keywords presented by the user.

4 ILLUSTRATION OF THE ALGORITHM

Keyword Query. ***abstract schmeck 2005***

This query has been chosen for illustration to show that our approach exploits the class/subclass relationship during the graph exploration phase for choosing the relevant nodes. The term mapping step provides one of the following mapping.

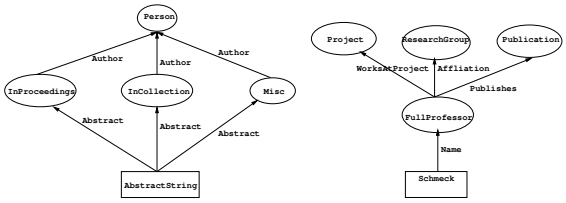
- Keyword *abstract* will be mapped on to the term *abstract*.
- keyword *Schmeck* will be mapped on to the term *Schmeck Blohm*.
- keyword *2005* will be mapped on the term *2005*.

The structural component creation step will form the three components as shown in *Figures 2a,2b and 3*.

In the pruning step the node *Project* with relationship *worksAtProject* and node *ResearchGroup* with relationship *affiliation* associated with cluster for *Schmeck* gets removed. Even though it appears that *Publication* is also an isolated node category, the class/subclass relationship of Publication with InProceedings, InCollection and Misc is used to retain the node.

The hooking step starts with a node in a component structure having smallest cardinality. If there are multiple options any one can be chosen. Starting with the *Person* node in *Figure 3* we explore whether it can be hooked with any node in *Figure 2a, 2b*. Even though *FullProfessor and Person are syntactically different* they are related by *class/subclass* relationship. We use this feature to hook them. In a similiar manner, after this hooking step, the node *Publication* in the merged component structure is hooked with nodes *InProceedings, InCollection and Misc* in component structure of *Figure 3*. Since *Person* node has already been considered and no more component structures need to be considered we get the final joined subgraph as shown in Figure 4.

(Tran et al., 2007) does not capture the Class/SubClass relationship as part of the RDF graph. As such it cannot handle the keyword scenario where the keywords either refer to the superclass or a subclass. We exploit this knowledge to arrive at the correct answer graph reflecting the data repository. Our approach uses schema knowledge rather than relying purely on the graph paths alone.



(a) *abstract.* (b) *Schmeck.*

Figure 2: Structural component for the *Illustration*.

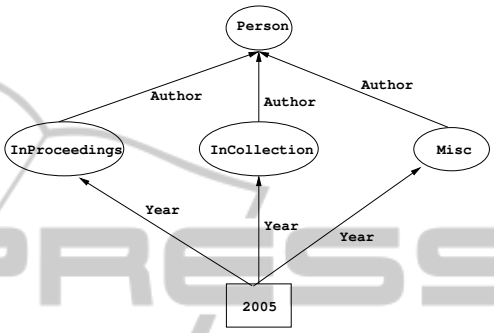


Figure 3: Structural component for *term 2005*.

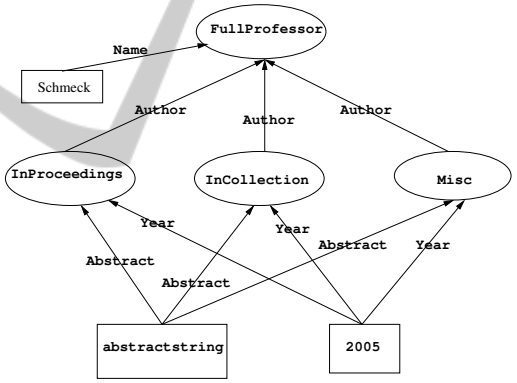


Figure 4: Answer graph for *Example*.

5 EVALUATION

In order to carry out an evaluation of our approach, the knowledge repository of AIFB institute, University of Karlsruhe[2,3] was used. The knowledge represents projects financed by the institute, details about researchers working on those projects and details about the publications of those projects. This knowledge is represented in RDF framework using OWL. For evaluation this framework was first represented as a graph and around 25 different queries

[2]http://km.aifb.uni-karlsruhe.de/ws/eon2006/ontoeval.zip

[3]http://ontoware.org/swrc/swrc/SWRCOWL/swrc_v0.3.owl

were generated. One such query is already illustrated. Some of the other queries were: *""projects X*(Retrieve all projects that X is working), *"author publications data mining"*(publication details related to data mining), *"journal 2006 publications"*(journal publications in 2006) and *"topic webservice publications"*(publications dealing with topic webservice). The approach seems to be promising as the answer graphs constructed for these queries provided the closest assembly of nodes and relationships to the keywords submitted. On the negative side, the algorithm proposed fails for the following keyword query *"AIFB journal"*. In this example which corresponds to an extreme case, *Organisation* and *Project* nodes get added to component structure of *AIFB* and *Article* and *Publication* gets added to component structure of journal. The pruning step does not identify any similiar nodes. The reason for the failure is that the keywords are far apart and hence pruning step fails to identify similiar nodes for hooking to happen. Pruning step will be suitably extended to handle these class of examples.

In our illustrations we have shown one answer graph that gets constructed out of the exploration phase. But given keywords multiple interpretations are possible and this leads to multiple answer graphs. For e.g given the keyword list *{X-Media, Philip, Publications}* the possible interpretations are

- *publications by Philip in Project X-Media*;
- *publications by Philip with X-Media in title*;
- *publications by Philip with X-Media in abstract*.

During the term mapping phase X-Media will be mapped on to name node , title node and abstract node. This leads to three answer graphs for the same set of keywords. There is a need to rank these graphs. We are working on a ranking metric which will be a function of the strength of the individual cluster and also on the strength of the hook between clusters.

6 RELATED WORK

In this paper we have addressed the issue of answer graph construction for keyword queries on graph structured data through a concrete algorithm for graph exploration. We have tried to improve the graph exploration through an alternative approach by adopting pruning and hooking as compared to (Zhou et al., 2007; Tran et al., 2007).

Keyword search on structured data has been extensively investigated in recent years under different contexts. Earlier approaches (Bhalotia et al., 2002; He et al., 2007; Kacholia et al., 2005) tried to address keyword search in the context of relational databases. Exact matches between keywords and labels of data elements were done. Also substructures in the form of trees were constructed and the root element is assumed to be the answer. (Bhalotia et al., 2002) uses backward search algorithm. In order to improve the search by limiting the nodes to be visited, (Kacholia et al., 2005) proposed bi-directional search algorithm where the exploration is through both backward and forward edge. The idea is to reach the root element faster through this approach. (He et al., 2007) also adopts distinct root semantics but improves the efficiency of the search using partitioning, balanced cost strategy and indexing to support forward jumps. These methods however do not exploit the schema knowledge for processing queries.

(Revuri et al., 2006) presents a system for keyword search that fits the query terms in an appropriate way from the ontology graph and derives an enhanced query. This query is given to the basic keyword search engine and results obtained are ranked. The system adopts a template based approach where it fixes the structure and then enhances the terms. Also the keywords are restricted to two terms. (Tran et al., 2007) presents a generic graph based approach to explore the connections between terms mapped to keywords of the query using knowledge available in ontologies. A three step process consisting of term mapping, connection exploration and DL query construction is used. The exploration is restricted to connections where an instance is related to a concept by an *is-a* relation and two instances are related by object and data properties.The exploration builds a graph connecting a term element with all its neighbours within a specified range *d*. 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. (Zhou et al., 2007) also adopts three step process: term mapping, query graph construction and query ranking. For each grouping of terms different query sets are constructed by enumerating all possible combinations from different sense of terms. From each query set a query graph is derived. A probabilistic ranking model is adopted for ranking the query graphs. In this system also the knowledge features and pruning mechanisms are not exploited during the exploration phase.

In our approach we have adopted a different strategy for the exploration phase. Unlike above we are not considering all the nodes for exploration.We create a fragment of closely related concept and relationship cluster and then prune unwanted nodes and edges. We also adopt a guided exploration

strategy which exploits other knowledge characterstics(class/subclass relationship) instead of purely relying on assertional knowledge. This approach can be logically extended if the keywords provided are not within a distance neighbourhood.

7 CONCLUSIONS AND FUTURE WORK

We have presented a concrete algorithm for answer graph construction, given a set of key- words and a knowledge repository represented as an RDF graph. We have also illustrated the approach for different keyword sets and AIFB in- stitute knowledge base.The results obtained on sample queries seem to be promising. This approach needs to be validated using implementation on standard benchmarks.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

REFERENCES

Allemang, D. and Hendler, J. (2008). *Semantic Web for the Working Ontologist Modeling in RDF, RDFS and OWL*. Morgan Kaufmann Publishers, MA, 1st edition.

Bhalotia, G., Hulgeri, A., Nakhe, C., Chakraborti, S., and Sudharshan, S. (2002). Keyword searching and browsing in database using banks. In *ICDE 2002*.

Guo, L., Shao, F., Botev, C., and Shanmugasundaram, J. (2003). Xrank: Ranked keyword search over xml documents. In *SIGMOD Conference 2003*.

He, H., Wang, H., Yang, J., and Yu, P. S. (2007). Blinks: Ranked keyword searches on graphs. In *SIGMOD Conference 2007*.

Kacholia, V., Pandit, S., Chakraborti, S., Sudharshan, S., Desai, R., and Karambelkar, H. (2005). Bidirectional expansion for keyword search on graph databases. In *VLDB 2005*.

Kimfield, B. and Sagir, Y. (2006). Finding and approximating top-k answers in keyword proximity search. In *PODS 2006*.

Lei, Y., Uren, V., and Motta, E. (2006). Semsearch: A search engine for the semantic web. In *15th International Conference on Knowledge Engineering and Knowledge Management(EKAW),(2006)*.

Revuri, S., Upadhyaya, S. R., and Kumar, P. S. (2006). Using domain ontologies for efficient information retrieval. In *International Conference on Management of Data COMAD 2006*.

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

Tran, T., Camiano, P., Rudolph, S., and Studer, R. (2007). Ontology based interpretation of keywords for semantic search. In *ISWC/ASWC,2007*.

W3C (2004). Owl web ontology language reference. In *http://www.w3.org/TR/owl-ref/*.

Zhou, Q., Wang, C., Xiong, M., Wang, H., and Yu, Y. (2007). Spark: Adapting keyword query to semantic search. In *ISWC/ASWC,2007*.