Algorithm for answer graph construction for keyword queries on RDF data

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 represented as a graph. Existing approaches for answering such keyword queries, identifies connected trees with minimal cost in the labeled graph as answers. In this paper we present an elegant algorithm for keyword query processing on RDF data that not only identifies trees but also more meaningful graph structures including cycles. The approach adopts a pruned exploration mechanism where closely related nodes are identified, sub-graphs are pruned and joined using suitable hook nodes. The system also exploits Type/SubClassOf relationship during the construction of the answer graph. 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 entity-relationship graphs 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[13] 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.

Conventional query languages like SPARQL through which complex information needs can be expressed, require the knowledge of the schema of underlying semantic data, whereas keyword queries do not require the users to know complex query language or know details regarding the underlying schema. An approach that can leverage the advantages of both query types is to provide a keyword user interface and then translate the queries into formal queries.

Much work has been carried out on keyword search on relational data [4, 12, 3], tree structured data[8] and recently on graph structured data [11, 10, 9]. Current algorithms compute minimal cost connected trees. The trees are identified using an approximation of Steiner Tree problem. But keywords need not map to only nodes but can also map to edges. The answer structure will in general be a graph which includes loops or cycles. There is a need to adopt a different algorithmic approach to address this issue. The graph exploration process also has to exploit semantic relationship(*Type/SubClassOf)* as the keyword might refer to parent type whereas the instance data might refer to a SubClassOf(e.g *Publications/Inproceedings)*

In this paper, we propose a novel algorithm for answer graph construction to keyword queries on RDF data represented as a graph, specifically addressing the above mentioned issues. We have extended our earlier approach presented in [6] by suitably adapting the algorithm to find not only trees but also cycles. For each mapped node or edge for the keyword, the associated concept(type) or set of concepts(types) and neighbouring concepts(types) are first identified to form a graph cluster. Using a pruning strategy, nodes from the clusters which cannot contribute to the overall answer graph for the query, are removed. 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 an approach to ranking,

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

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

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 types), EN-Nodes(representing entities) and D-Nodes(data values) i.e N=C-Nodes ⊎ EN-Nodes ⊎ D-Nodes.* In the RDF fragment shown in Figure 1., *FullProfessor, PhDStudent* are examples of *C-Nodes*, *Rudi Studer, Semantic Web services* are examples of *D-Nodes* and *topic1, proj1* are examples of *EN-Nodes. EN-nodes* are resource identifiers referred by using internal IDs and will not be used for queries as users will not be aware of these IDs .

- 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 *Type/SubClassOf* edges. In *Figure 1 Author, WorksAt are examples of IE-Edges* and *Title, Name are examples of EA-Edges.*

- L is a labeling function which associates a label *l* for an edge. *L = L(IE-Edges) ⊎ L(EA-Edges) ⊎ {Type, SubClassOf} 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* ∈*L(IE-Edges) if and only if* n_1,n_2∈ *EN-Nodes*
  - *l* ∈*L(EA-Edges) if and only if* n_1∈ *EN-Nodes and* n_2∈ *D-Nodes*
  - *l = SubClassOf if and only if* n_1,n_2∈ *C-Nodes*
  - *l = Type if and only if* n_1∈ *EN-Nodes and* n_2∈ *C-Nodes.*

Type and *SubClassOf* are two predefined type of edges which capture 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 either through an EN-Node or through edge with the label SubClassOf. For example the C-Node PhdStudent is a CR-Node for the C-Node Project 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 association of Professors and Students with projects and events, Topics related to the project and publications related to the topics.*

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 a minimal possible subgraph A such that

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

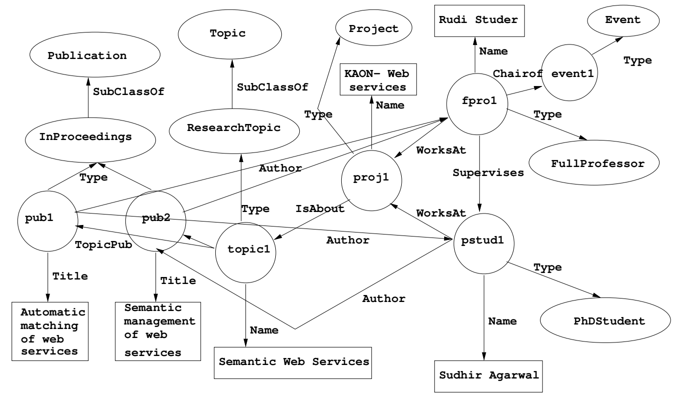

Figure 1: RDF Graph fragment from AIFB Institute Data

- Every keyword is contained in at least one node or edge in G

- The graph consists only of *Nodes* mapped to keywords, *CR-Nodes* connected by inter-entity edges, *Dummy D-Nodes* created during the algorithmic phase and *labels* belonging to *L(IE-Edges) and L(EA-Edges).*

- Answer A is minimal in the sense that no sub-graph of A can be an answer to Q. If a keyword node is removed, then that keyword is not matched. If a non keyword node is removed, the graph becomes disconnected.

The answer graph can be used to generate structured queries using SPARQL query framework.

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 (*SubClassOf* relationship) and detects not only trees but also cycles.

3. ALGORITHM DESCRIPTION

The terms identified by the term mapping step for each keyword k_i are mapped 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. For each node , its category is found. 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 type 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 Type 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. A node is a loosely hanging node, if it is a *CR-node* and if it is not on a path between participating nodes. 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 *SubClassOf* relationship
3. There exists a chain of intermediate *C-Nodes* which connects the two nodes.

We make use of *property 2* for exploiting Type/SubClassOf relationship. 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*. The union of all the similiar nodes found by considering all pairs of component structures, represents the participating node list. The nodes to be pruned is the complement 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. Also we explore whether any similiar graph patterns in terns of nodes and edges exists for merging the clusters. The new list of component clusters obtained will be the input for the *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.

Our earlier algorithmic approach presented in [6] has been adapted suitably to detect not only trees but also graphs by modifying the *pruning* and *hooking steps suitably.* In [6], we have also highlighted the example scenarios where the algorithm fails even in case of answer trees. These scenarios correspond to cases where the keywords are not closer. 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. We have modified the algorithm in [6] *using property 3 of similiar nodes* to remove the distance neighbourhood restriction for exploration and to adapt to scenarios where some keywords will be closer and some keywords are farther apart. This is presented in [7]. A suitable ranking model is also presented in line with our approach. This also demonstrates that our algorithmic framework adapts to different scenarios.

4. ILLUSTRATION OF THE ALGORITHM

Keyword Query: ***titles topic webservice student studer***

- Keyword *topic* will be mapped on to the term *topic*
- Keyword *webservice* will be mapped on to the term *webservice*
- Keyword *titles* will be mapped on to the term *title*
- Keyword *student* will be mapped on to the term *PhdStudent*
- Keyword *Studer* will be mapped on to the term *Rudi Studer*

In the Component_Cluster_Creation step, using the Type nodes and relationship nodes associated with the terms, the component clusters as shown in *Figure 2, Figure 3 and Figure 4* are constructed. For the term *Rudi Studer*, the *C-Node FullProfessor*, *CR-nodes PhDStudent ,Project and Event and edges Supervises ,WorksAt and ChairOf* are used to form the component cluster. For the term *PhdStudent,* which refers to a *C-Node*, the *CR-Nodes FullProfessor, Project, InProceedings along with the respective edges 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 C-Node InProceedings and CR-Nodes FullProfessor, PhdStudent and ResearchTopic are added to form its component cluster as shown in Figure 3.* For the term *webservice* which is mapped on to the *D-Node 'Semantic Web Service',* the *C-Node ResearchTopic along with CR-Node InProceedings are added to form the cluster.* For the term *topic* the *C-Node topic, the CR-Node ResearchTopic and the CR-Node InProceedings is used to form its cluster.*

In pruning step, we take the pairwise intersection of the nodes of the components to prune hanging nodes. For the clusters constructed, the *Node Event along with the associated edge ChairOf* will be pruned. The component cluster of *webservice* is identical to the component cluster of *Topic* except for the additional *D-Node an*d the node *ResearchTopic.* But *ResearchTopic and Topic are similiar nodes(Type / SubClassOf relationship).* Both these clusters are superimposed by pushing the *D-Node* to the smaller sub-cluster and retaining the node *ResearchTopic.*

In the hooking step, the *InProceedings* node of *webservice* cluster is hooked with the *InProceedings* node of *Title* cluster and merged. In the next hooking operation, the *Phdstudent node of previous merged cluster is hooked with PhdStudent node of PhdStudent cluster(Figure 5).* Finally in the last hooking, *FullProfessor node of the previous merged cluster will be hooked with the FullProfessor node of Rudi Studer Cluster.* 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 adopted for keyword queries on graphs and also to the approach adopted in [11], our approach has the following enhancements

- Edge mapping is allowed
- Exploits graph semantics(*Type/SubClassOf relationship*) during the graph exploration phase
- The final answer graph can also have loops or cycles
- Uses graph similiarity pattern for merging similiar graphs.

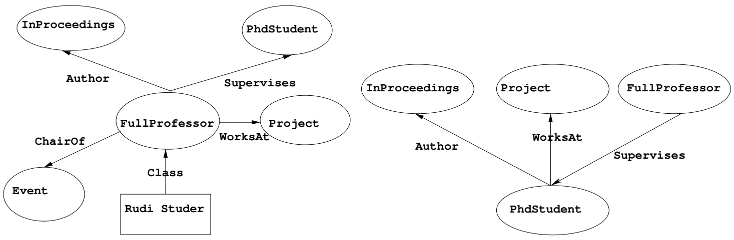

Figure 2: Component Cluster for *Rudi Studer,Phdstudent*

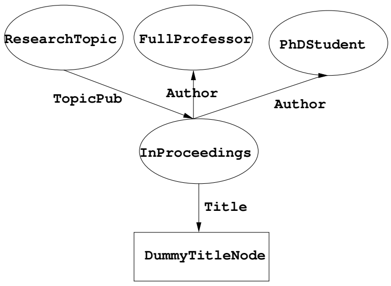

Figure 3: Component Cluster for *title*

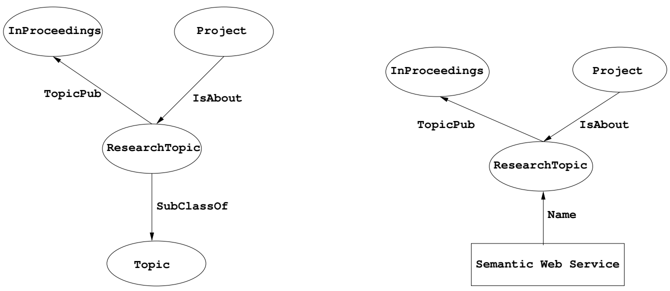

Figure 4: Component clusters for *topic,webservice*

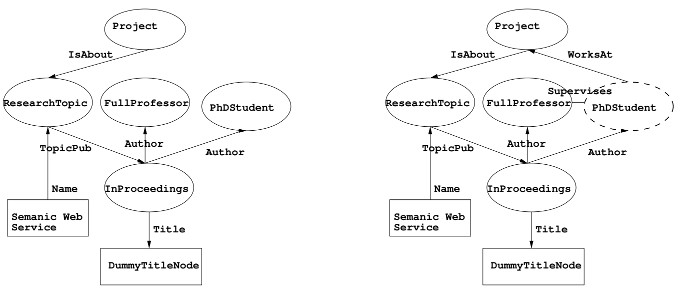

Figure 5: Component clusters formed during hooking}

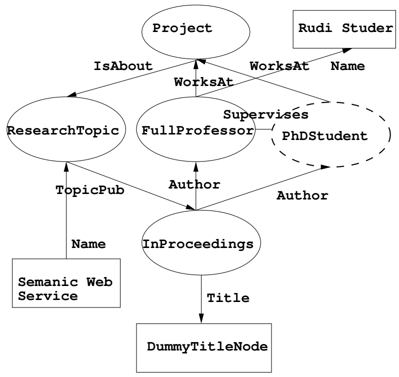

Figure 6: Final answer graph for the query

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. Since we adopt a graph model, ranking from a graph perspective is equally important. In [11], the path length is used for ranking the answers. To the best of our knowledge only [9] provides a comprehensive ranking mechanism for graph data. Our approach is also 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.

Compactness Relevance For the keywords which reflect the information needs of the user, compact answers should be preferred. From a graph perspective, this translates into structural compactness among the elements of the graph. In our approach 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:

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

nodes

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

Term Relevance Since the term mapping step uses IR model, the textual relevancy of the keywords with reference to the nodes mapped is one of the important attribute for ranking. For example, the term *webservice* gets mapped to node with name of the topic having *webservice* or it gets mapped to title of a publication which has the string *webservice* or gets mapped to keyword of a publication which has the string *webservice.* 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

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

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*. The answer graph which has the *C-Node* should be ranked higher since the user intention will also be to get publication information rather than abstract information. *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:

$$\mathrm{RankVal(AG)} = \sum_{1\leq i\leq j\leq n}\mathrm{RankVal}(k_i,k_j|\ \mathrm{AG}) + \mathrm{NR(AG)} + \mathrm{RR(AG)}$$

6. RELATED WORK

Considerable work has been reported in literature on *keyword search* on graph structured data[3, 4, 12]. [3] identifies substructures in the form of trees using an approximation of Steiner tree problem. [12] presents a bi-directional approach to improve the efficiency. [4] proposed a partition based approach to improve the efficiency with a novel indexing scheme. In [11], which is one of the first paper to address keyword based search on entity-relationship graphs in a comprehensive manner, the exploration builds a graph connecting a term element with all its neighbours within a specified range *d*. In [9], the search is modeled as *r-Radius Steiner Graph problem i.e identifying all r-radius Steiner graphs which contain all the keywords within a neighbourhood.* [10] also presents a graph traversal approach with probablistic ranking for keyword queries.

In our approach, which is greatly inspired by that of [11], we create a fragment of closely related concept cluster and then prune unwanted nodes and edges. We also adopt a guided exploration strategy which exploits other knowledge characteristics(Type/SubClassOf relationship) instead of purely relying on assertional knowledge. Our approach not only constructs trees but also graphs. We do not fix any distance metric before-hand. We have also proposed a ranking scheme for the answer graphs in line with our approach.

7. CONCLUSIONS AND FUTURE WORK

We have presented a concrete algorithm for answer graph construction given a set of keywords and a knowledge repository represented as an RDF graph. We have also illustrated the approach on a sample query where the interest is to find out a structural relationships among the keywords which are not only trees but also cycles . The approach seems to be promising and currently it is being implemented for validation on standard data-sets. We are also working to extend the scope 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] K.Parthasarathy, P.SreenivasaKumar, and Damien, D. Ranked answer graph construction for keyword queries on rdf graphs without distance neighbourhood restriction. In *Accepted in PhD Symposium WWW 2011* (Mar 2011), WWW.

[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] 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.

[12] 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.

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