

An Agglomerative Query Model for Discovery in Linked Data: Semantics and Approach

Sidan Gao
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
sgao@ncsu.edu

Haizhou Fu
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
hfu@ncsu.edu

Kemafor Anyanwu
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
kogan@ncsu.edu

ABSTRACT

Data on the Web is increasingly being used for discovery and exploratory tasks. Unlike traditional fact-finding tasks that require only the typical single-query and response paradigm, these tasks involve a *multistage search process* in which bits of information are accumulated over a series of related queries. The ability and effectiveness of users to connect the dots between these pieces of information is crucial to enable discovery. In this paper, we introduce the notion of *agglomerative querying* for supporting “search processes” and present its motivation, formalization and challenges. We focus on a specific class of agglomerative querying called *association agglomerative querying* which is very natural for linked data models such as RDF. We present a preliminary implementation approach for processing such queries and discuss its relationship with SPARQL query processing. Finally, we present empirical results for proving the effectiveness of our approach on the DBLP dataset and future directions.

Categories and Subject Descriptors

H.2.4 [Information Systems]: Systems: Query Processing

General Terms

Algorithms, Measurement, Experimentation

Keywords

Exploratory Query, Agglomerative Query Model, RDF Databases

1. INTRODUCTION

The increasing popularity of metadata standards like RDF [22] and OWL [23] has resulted in heightened focus on storage and querying issues. Most of the existing query languages [3][5] and querying systems [7][8][15][19] for RDF data primarily support a graph pattern matching querying paradigm in which a query (graph pattern) returns a list of matching subgraphs. A characteristic shared by these approaches as well as relational querying techniques is that the query model is a *single query-response paradigm* in which queries are considered independent of each other and the focus is on answering a specific query posed. Such a query model can only be suitable for fact-finding tasks where users specify their needs in a single data pattern description. However, when search tasks are more exploratory in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WebDB '10, June 6, Indianapolis, IN USA

Copyright © 2010 ACM 978-1-4503-0186-2/10/06... \$10.00

nature or discovery-oriented, information need is typically not satisfied by the answer to a single query. Rather, knowledge is acquired in a multistage querying *process* in which pieces of information are gathered and weaved together across a series of questions. [6] characterizes the nature of such searches as “*information-seeking problems that are open-ended, persistent, and multi-faceted*” with processes that are “*opportunistic, iterative, and multi-tactical*”.

Motivating Example As a Ph.D. student, Matt is interested in anything that will help him advance his education and research. He decides to explore the topic “Data Intensive Computing” and begins by submitting a query to retrieve papers on “*data intensive computing*” in the last 3 years. He also queries about conferences that focus on this topic and other related questions. Later, he decides to begin a seemingly unrelated task of registering for courses for the following semester. He queries about the list of courses being offered for the next semester. Matt may choose to make his decision on courses based on information he receives from friends. However, providing Matt with additional information about associations between these queries and his earlier queries could help Matt make a more informed decision. For example, it may be useful to highlight the existence of an association between one of the query answers in query 3 and another answer in query 2 as shown in the lower part of Figure 1. Specifically, (1) *paper “Simultaneous scalability and security for data-intensive web applications” was published in SIGMOD06*; (2) *professor Peter Buneman served as PC chair for an earlier SIGMOD conference (1993)*. Professor Peter Buneman teaches the course “*Applied Databases*”, offered the next semester and therefore should be considered as a potential course to take in that semester.

Awareness about relationships in data is crucial to the weaving process that users confront in exploratory or discovery tasks and supporting their exposition, which could expedite the process of discovery. Further, they could provide clues on other paths that could be pursued. For example, Matt may now consider, Prof. Buneman as a potential dissertation committee advisor and try to schedule a meeting with him. This motivates the need for *agglomerative querying models* that knit together information across multiple queries in order to provide users with useful additional information beyond their specific queries. This paper goes in that direction.

1.1 Challenges and Contributions

There are three main challenges that must be overcome in the development of agglomerative querying models: (1) support for multistage querying processes, caching and management of query and answer histories; (2) the need for efficient computational strategies for association computation; (3) the need for a cost model for estimating the “informativeness” of associations. The

primary focus of this paper is (3) and some components of (1) with a preliminary discussion on (2). Specifically,

- We introduce and formalize the novel problem of *agglomeration query answering over RDF databases* with particular focus on a specific class called *association agglomeration query answering* whose goal is to augment queries on RDF databases with information about query associations.
- Present an information-theoretic cost model for assessing “informativeness” of an association. This leads to the problem of *Optimal Association Agglomeration Query Answering (OAAQA)* problem that ensures that only the most “informative” associations are presented to users.
- A preliminary computational approach for the OAAQA problem is presented based on earlier work on a graph serialization technique for supporting path computation on disk resident databases. We present evaluation of the effectiveness of our approach on a real world dataset, DBLP.

2. RELATED WORK

Top-K Weighted Subgraph Discovery In the context of graph structured data, there have been different research efforts that focused on computing subgraphs connecting graph elements [2][4][7][9][10][19][20]. In keyword search area, existing work [4][7][20] focused on computing spanning or Steiner trees using cost models that prefer smaller trees. For the problem of connection subgraph discovery, current approaches have focused on computing “best” subgraphs connecting query nodes. [2][9] proposed a two phase approach, in which a candidate graph is generated first and then a final smaller “good” display is computed from the candidate graph. Recently, [10] used a decomposition technique by which they partition a large graph into a set of small ones, compute “best” connection within each small graph, and then extend the connection between smaller graphs using an index called community hierarchy tree. Unlike our approach, these approaches use navigational style (e.g., DFS-like) algorithms on memory-resident graphs. However, for disk-based graphs, appropriate storage models will need to be developed to avoid the problem of each step of graph exploration resulting in a disk I/O.

Other Related Efforts Query suggestion systems have been

proposed to facilitate users in their search process by recommending related queries [14]. However, in these systems, recommendations are given based on similar queries issued previously, while our approach does not require the existence of past similar search. There has been similar recent work motivated by the lack of support for discovery tasks that has resulted in the development of Exploratory Systems [12][13][16][18]. However, current efforts largely focus on user interface issues and try to aid and guide the user while browsing a document collection. Also, these systems do not effectively support linking and combining of search results across a long-lived discovery process that involves multiple queries.

3. AGGLOMERATIVE QUERYING: MODEL AND SEMANTICS

An *Agglomerative Query Model* “agglomerates” information in a user’s querying context that typically consists of a series of queries. It is proposed as an extension of the traditional SPARQL pattern matching querying model in which answers of newer SPARQL queries are augmented with useful information based on previous queries. In this section, we present a specific subclass of agglomerative querying which we call *association-based agglomeration*. Its presentation begins with a review of SPARQL query semantics.

3.1 Agglomerative Querying

An *RDF graph* is a collection of subject-predicate-object (s-p-o) triples, in which subject and object are mapped into graph nodes and properties (binary predicates) are mapped into graph edges. Both nodes and edges in an RDF graph are Semantic Web resources identified by URIs but objects may be literals. We refer to URIs and literals as RDF terms.

A *graph pattern Q* is a set of triple patterns which are RDF triples with variables in either of the s-p-o positions. Given a graph pattern query *Q*, a *query answer* to *Q* over an RDF dataset *D*, denoted by $\llbracket Q \rrbracket_D$, is a variable substitution δ , such that $\llbracket Q \rrbracket_D = \{\delta \mid \text{dom}(\delta) = \text{Var}(t) \text{ and } \delta(t) \in D\}$, where *t* is a triple pattern in *Q*, $\text{dom}(\delta)$ is the domain of δ , $\text{Var}(t)$ is the set of variables in *t*, $\delta(t)$ denotes the triple obtained by substituting RDF terms for the variables in *t* in terms of δ . In brief, a query answer is a set of substitutions of variables by RDF terms. In this paper, we focus on a specific subclass of SPARQL queries: those in which variables only occur at subject or object positions.

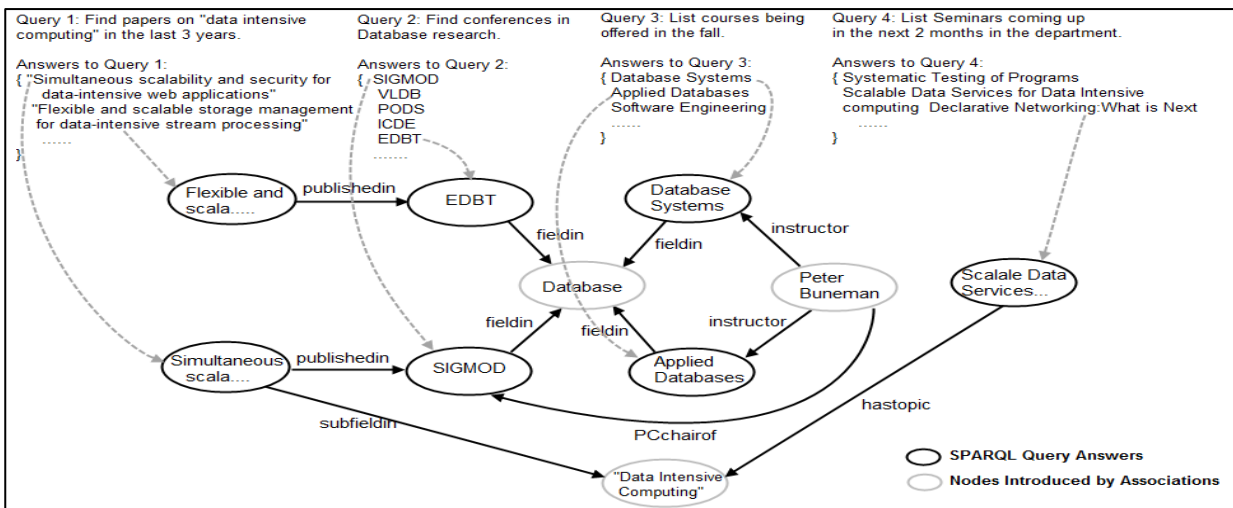


Figure 1 Motivating Example

Definition 1 (Agglomerative Query): Given a query sequence Q_1, Q_2, \dots, Q_i , an *agglomerative query* Q_i^* for Q_i is a function

$$Q_i^* \triangleq \text{Agg}^*(\llbracket Q_i \rrbracket, \{\llbracket Q_j \rrbracket \mid j = 1 \text{ to } i - 1\}),$$

where Q_j is a query that precedes Q_i and Agg^* is a member of a class of *agglomerative functions* i.e. functions of the current query and answers to previous queries. This paper focuses on the Agg^* function that computes associations i.e. connection subgraphs between query answers denoted by Θ and refer to this class of agglomerative queries as *association agglomeration queries*.

Definition 2 (Path Expression): A *Path Expression* (n_x, n_y, PE) is a triple such that n_x and n_y are source and destination nodes respectively, and PE is regular expression over triples/labeled edges whose language contains strings that represent paths between n_x and n_y . For example, the path expression $(n_x, n_y, (a \bullet b) \cup (e \bullet f \bullet g))$ represents two paths $n_x, e, n_1, f, n_2, g, n_y$ and n_x, a, n_1, b, n_y between n_x and n_y .

Definition 3 (Association Agglomeration Query Answer (AAQA)): Given a set of query answers $\{\llbracket Q_j \rrbracket_D \mid 1 \leq j \leq i - 1\}$ and a new SPARQL query Q_i , an *association agglomeration query answer* over D , denoted as $\llbracket Q_i \rrbracket_D^*$, is a tuple:

$$\begin{aligned} \llbracket Q_i \rrbracket_D^* &\triangleq \Theta(\llbracket Q_i \rrbracket, \{\llbracket Q_j \rrbracket \mid j = 1 \text{ to } i - 1\}) \\ &= (\llbracket Q_i \rrbracket_D, \left[(n_x, n_y, PE)_m \right]) \end{aligned}$$

such that the first element is the answer to the current SPARQL query and the second element is a sequence of path expressions:

- (1) $n_x \in \llbracket Q_k \rrbracket_D, n_y \in \llbracket Q_l \rrbracket_D$ for $1 \leq k, l \leq i$. Note that n_x and n_y may belong to the same query answer;
- (2) PE is a regular expression that represents paths from n_x to n_y in D .

Example: Suppose, as shown in Figure 1, the previous queries Q_1 and Q_2 are 1. “Find papers on ‘data intensive computing’ in the last 3 years”, 2. “Find the top conferences in Database research” and the current query Q_3 is “List courses that will be offered in the fall”, then

$$\begin{aligned} \llbracket Q_3 \rrbracket_D^* &= \\ &((\text{Database Systems}, \text{"Software Engineering"}, \text{"Applied Databases"}), \\ &, [(\text{"Database Systems"}, \text{"VLDB"}, \text{fieldin}), \\ &(\text{"Applied Databases"}, \text{"SIGMOD"}, \text{Instructor} \bullet \text{PCchairof})]) \end{aligned}$$

3.2 Problem Statement

Each SPARQL query could have a large number of answer nodes for which we want to compute associations. Further, there could exist multiple associations between each pair of query answers. Providing all answers might lead to information overload. Ideally, we want to compute the most “*informative*” associations for the most important entities in query results. Therefore, it is expected that agglomerative query answering will have a better outcome if only a subset of (top) previous query results are considered. Further, we are only interested in the most important associations amongst the selected subset. We now define the specific problem we address in this paper.

1. PROBLEM (Optimal Association Agglomeration Query Answer (OAAQA)): Given a query Q_i , a subset $\llbracket Q_j \rrbracket$ of query answers $\{\llbracket Q_j \rrbracket \mid j = 1 \text{ to } i - 1\}$, an integer budget b , and

a score function w whose domain is the set of possible path expressions, the problem *optimal AAQA* is to compute $\llbracket Q_i \rrbracket_D^* = (\llbracket Q_i \rrbracket_D, P)$, where

- (a) $P = \left[(n_x, n_y, PE)_m \mid n_x, n_y \in \{\llbracket Q_j \rrbracket\}' \right]$, and
- (b) $w \left((n_x, n_y, PE)_n \right) < w \left((n_x, n_y, PE)_m \right)$ if $n < m$, such that
 - (i) P maximizes $\sum_m w \left((n_x, n_y, PE)_m \right)$, and
 - (ii) the total number of edges contained in all the paths represented in $\left[(n_x, n_y, PE)_m \right]$ being equal to or less than b .

We refer to $P = \left[(n_x, n_y, PE)_m \right]$ as an association set.

Proposition 1: The *Optimal Association Agglomeration Query Answering* problem is NP-hard.

Proof Sketch – Reduction to 0/1 Knapsack problem. We construct an instance of OAAQA by setting the items as paths that connect pairs of entities in query answers, corresponding the weight to the length of path connecting one pair of entities and setting the price as the score of the path between the pair of entities. ■

2. Entropy Based Cost Model

Here, we present an *entropy based cost model* to capture the “*informative content*” of the associations in an association set \mathcal{R}_i . Entropy is a formal representation of the amount of information conveyed by an event, which can be measured by the negative logarithm of the probability of occurrence of the event [1]. Thus, for a random variable or an event, denoted as X , which has n possible outcomes x_1, x_2, \dots, x_n with probabilities $p(x_1), p(x_2), \dots, p(x_n)$, the entropy is given by

$$H(X) = -\sum_{i=1}^n p(x_i) \log(p(x_i)) \quad (1)$$

Intuitively, the entropy of a graph can be defined as the sum of the entropy of its components. In our case, the components of an association set \mathcal{R}_i denote the paths that connect a pair of nodes in the query answers. We begin with defining the entropy $H(\text{PATH}_j(n_x, n_y))$ of a path $\text{PATH}_j(n_x, n_y) = n_x, Pr_1, n_1, Pr_2, n_2, \dots, Pr_m, n_y$ that connects n_x and n_y where $n_i, 1 \leq i \leq m - 1$, are entities and $Pr_j, 1 \leq j \leq m$, are properties. Assume Pr_1, Pr_2, \dots, Pr_m are independent from each other, then, based on (1), we have

$$H(\text{PATH}_j(n_x, n_y)) = -\sum_{k=1}^m p(Pr_k) \log(p(Pr_k)) \quad (2)$$

where $p(Pr_k)$ is the probability of the property Pr_k occurring in an RDF dataset D . Effectively capturing the informativeness of associations must account for both the degree of “relatedness” of nodes (i.e., the number of nodes they connect to) as well as the type of relationships they have i.e. the types of properties on the connections. Consequently, we propose a *hybrid entropy function* consisting of two types of entropy measures: *property-specific entropy* and *property-independent entropy*. For *property-specific entropy*, the probability function used for entropy calculation is defined as:

$$p_{SD}(Pr_k) = \frac{|Pr_k|_D}{|Pr|_D} \quad (3)$$

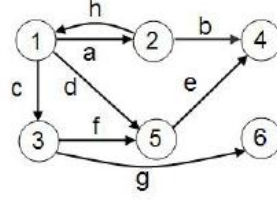
where $|Pr_k|_D$ is the number of instances of the property Pr_k in the dataset D and $|Pr|_D$ is the total number of instances of all properties in D . The probability function used for calculating *property-independent entropy* is defined as:

$$p_{SI}(Pr_k) = \frac{\text{outDeg}(v_k)}{\sum_{v \in V} \text{outDeg}(v)} \quad (4)$$

K-Solve

Given a topological sorted set of source nodes S :

1. for $i = 1$ to l
2. if not_exists S'_{w_i} , then $S'_{w_i} := \{\}$
3. if $v_i \in Q$, then $S'_{w_i} \leftarrow S'_{w_i} + \{v_i\}$
4. if $v_i = w_i$ then
5. foreach $s \in S'_{v_i}$ do
6. $PE(s, v_i) := PE(s, v_i) \bullet PE_i$
7. else
8. foreach $s \in S'_{w_i}$ do
9. $PE(s, w_i) := PE(s, w_i) \cup [PE_i(s, v_i) \bullet PE_i]$



- | | | | |
|------------|--------------|---------------|--------------|
| 1. (1,2,a) | 2. (1,3,c) | 3. (1,5,d) | 4. (2,3,h*c) |
| 5. (2,4,b) | 6. (2,5,h*d) | 7. (3,5,f) | 8. (3,6,g) |
| 9. (5,4,c) | 10. (2,1,h) | 11. (1,1,a*h) | |

Query Nodes (2,3)

Initialize: (2,2)= λ , (3,3)= λ

1. (2,2) = (2,2)U(2,1)*(1,2) = $\lambda U((0 \bullet a)) = \lambda$
2. (2,3) = (2,3)U(2,1)*(1,3) = $0 U((0 \bullet c)) = 0$
3. (2,5) = (2,5)U(2,1)*(1,5) = $0 U((0 \bullet d)) = 0$
4. (2,3) = (2,3)U(2,2)*(2,3) = $0 U((\lambda \bullet (h \bullet c))) = (h \bullet c)$
5. (2,4) = (2,4)U(2,2)*(2,4) = $0 U((\lambda \bullet (b))) = (b)$
6. (2,5) = (2,5)U(2,2)*(2,5) = $0 U((\lambda \bullet (h \bullet d))) = (h \bullet d)$
7. (2,5) = (2,5)U(2,3)*(3,5) = $(h \bullet d) U((h \bullet c) \bullet f) = (h \bullet d) U((h \bullet c) \bullet f)$
- (3,5) = (3,5)U(3,3)*(3,5) = $0 U((\lambda \bullet f)) = (f)$
8. (2,6) = (2,6)U(2,3)*(3,6) = $0 U((h \bullet c) \bullet g) = ((h \bullet c) \bullet g)$
- (3,6) = (3,6)U(3,3)*(3,6) = $0 U((\lambda \bullet g)) = (g)$
-

Figure 3 K-Solve Algorithm and Demonstration

the pair of nodes. The algorithm iteratively computes the maximum entropy for $i+1$ paths by using the maximum entropy for i paths using the function

$$m[i+1][h] = \max(m[i][h], m[i][h-l_i] + e_i)$$

where entry $m[i][h]$ contains the maximum entropy of any subset of paths $\{1, 2, \dots, i\}$ of (combined) length at most h . After we compute all the entries of this array, the array entry $m[M][b']$ contains the maximum entropy of paths whose total length is less than or equal to b' .

• Step 2.2 – Approximate OAAQA generation

Finally, we generate the approximate OAAQA by first sorting the entropy list computed by step 2.1; then, iteratively picking top element from the sorted entropy list and inserting it into the result list if (1) the total length of all paths chosen is less than or equal to global budget b ; and (2) the entropy list is not empty.

5. EVALUATION

Experiment Setup We use SwetoDbIp-Jan2008 [21], which is a real world dataset about computer science publications. The experiments were conducted on a machine with 2.33GHz Intel Xeon and 16GB memory running on Linux, with a 2.6.18 kernel. The detailed experimental results can be found at <http://www4.ancsu.edu/~sgao/>.

5.1 Scalability Evaluation

For the scalability of OAAQA generation evaluation, a subset of SwetoDbIp consisting of 65K nodes and 80K edges was used as evaluation testbed. We ran 12 test cases, partitioned into two groups. One group was based on a query sequence of length 2, the other length 3. The input sizes for each group were 10, 30, 50, 80, 150, and 230. Figure 4 shows the experiment results for the two groups and shows that the OAAQA generation time increases sublinearly with increasing input size. The reason is as follows. For an input Q , K-SOLVE runs in $O(l|Q|)$, where l is the number of pre-computed path expressions. By factoring out the disk I/O for the $|Q|$ nodes, during K-Solve process, we avoid doing potentially $|Q|$ separate disk I/Os for each path expression leading to an improvement in performance.

5.2 Evaluation on ratio of EI to ET

For this experiment, we measured the ratio of Extra Information (EI) to the Extra Time (ET) spent for the computation, where $EI = \sum H(\mathcal{G}_i)$, \mathcal{G}_i is the top-K optimal association and ET is OAAQA generation time. We evaluated different values of budget b against different input sizes selected from two groups of queries (corresponding to different areas in the graph). Results are shown in Figure 5. For both groups, we have the general trend of

increased EI/ET ratio as budget b increases. This is expected because we can add extra information into the budget from the associations computed. We also see a decrease in EI/ET ratio with increasing input sizes for all budget sizes. To understand why, we consider the total number of associations actually found for the different input sizes, 5 associations for $N=10$ and $N=30$, and slight increases thereafter, 9 for $N=50$ and 13 for $N \geq 150$. This means that as the input sizes increases we are doing a lot of unnecessary computation (computing associations originating from query pairs but not terminating at query nodes) that does not contribute to the result. So we have increased amount of time with much less increase in amount of extra information leading to the decreasing ratio. The results also show a spike when going from $N=30$ to 50. This is because four more associations are generated from $N=30$ to 50 with much less increase in extra time used for OAAQA. The spike doesn't increase after $N=50$ because the additional 4 associations require much more time for the association computation part of OAAQA (much of which is not useful) because of the much larger number of input nodes.

5.3 Evaluation on ApproRatio

This evaluation focused on the quality of our OAAQA approximation. It is based on a measure *ApproRatio* defined as a ratio of approximate solution for our approach to exact solution, where the exact solution is the exact optimal associations given global budget b and the approximate solution is the optimal associations generated by our heuristic algorithm. Figure 6 shows the results of *ApproRatio* evaluation for two groups of queries with budget $b=10, 20, 40$ for $N=50, 150, 230$. For both groups, we find the similar trend that the *ApproRatio* increases with the increase of input size. To explain why, we note that there might exist multiple exact solutions for some given global budget and with an increasing of the number of generated associations, the possible number of exact solutions also increases. Consequently, our approach has an increased probability of finding some exact solution. Figure 6A also shows that when budget b increases the *ApproRatio* also increases. Recall that our local budget is some fraction of the global budget b . When b is small, the local budget b' can be very small. Considering a pair of nodes, if there are two associations between them with length of 1 and 2, respectively, our approximate algorithm will choose the association of length 1 as the optimal one even if the other one might be in the exact solution. This means small budget will impact the quality of our approximation results. However, this is not always the case. Figure 6B shows that, for group 2, small budget generates high *ApproRatio*. On close observation, we see that there are a large number of associations that have length 1 with high entropies

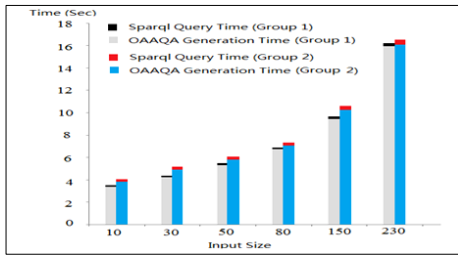


Figure 4 Effect of Input Size on Query Time

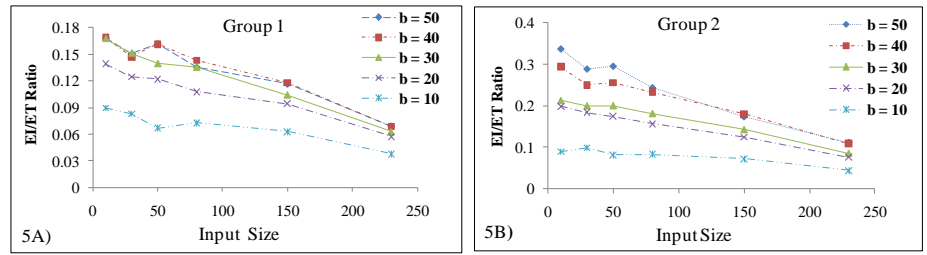


Figure 5 Evaluation on EI/ET

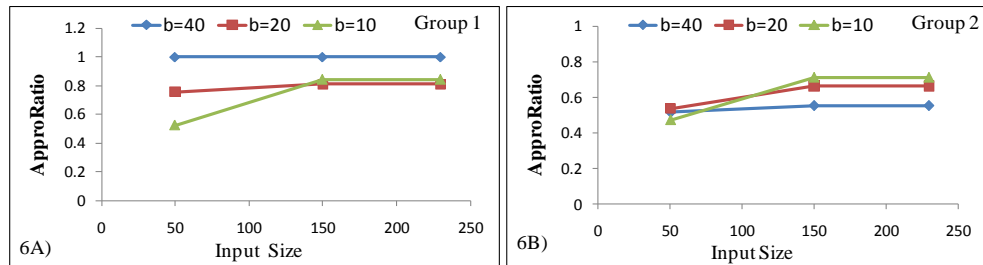


Figure 6 Evaluation on ApproRatio

(around 50%). Thus, the probability of associations of length 1 will be included in some exact solution increases. In this case, even if the local budget is set to one, we are still more likely to pick the association that is in exact solution as our approximate solution.

6. CONCLUSIONS

This paper presents a novel agglomerative querying model for supporting multistage search process. It introduces and formalizes the novel problem of association agglomeration query answer over RDF databases and proposes a simple approach for supporting computation on disk resident databases. In the future, we plan to investigate additional optimization strategies e.g. caching and incremental algorithms to further reduce disk I/O and improve performance.

7. ACKNOWLEDGEMENT

The work presented in this paper is partially funded by NSF grant IIS- 0915865.

8. REFERENCES

- [1] C. E. Shannon. A mathematical theory of communication. *BELL SYST TECH J*, 27:379–423, 1948.
- [2] C. Faloutsos, K. S. McCurley, A. Tomkins. Fast discovery of connection subgraphs. In *KDD 2004*: 118–127.
- [3] E. Prudhommeaux, A. Seaborne. *SPARQL Query Language for RDF*. W3C Recommendation, 2008.
- [4] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In *ICDE 2002*: 431–440.
- [5] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl. *RQL: A Declarative Query Language for RDF*. In *WWW 2002*: 592–603.
- [6] G. Marchionini. Exploratory search: From finding to understanding. *Communications of the ACM*, 49(4): 41–46.
- [7] H. He, H. Wang, J. Yang, P. S. Yu. BLINKS: ranked keyword searches on graphs. In *SIGMOD 2007*.
- [8] H. Wang, K. Zhang, Q. Liu, T. Tran, Y. Yu. Q2Semantic: A Lightweight Keyword Interface to Semantic Search. In *ESWC 2008*:584–598
- [9] H. Tong, C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *KDD 2006*: 404–413
- [10] J. Cheng, Y. Ke, W. Ng, and J. X. Yu. Context-Aware Object Connection Discovery in Large Graphs. In *ICDE 2009*.
- [11] K. Anyanwu, P. Kumar, A. Maduko. Structure Discovery Queries in Disk-Based Semantic Web Databases. In *SKG 2008*.
- [12] M. L. Wilson and M. C. Schraefel. Improving Exploratory Search Interfaces: Adding Value or Information Overload? In *HCI 2008*: 81–84.
- [13] M. Tvarožek, M. Bieliková. Collaborative multi-paradigm exploratory search. In *Hypertext 2008 workshop on CCI*.
- [14] R. A. Baeza-Yates, C. A. Hurtado, M. Mendoza. Query Recommendation Using Query Logs in Search Engines. In *EDBT Workshops 2004*:588–596.
- [15] R. Angles and C. Gutierrez. Querying RDF data from a graph database perspective. In *ESWC 2005*: 346–360.
- [16] R. Capra and G. Marchionini. The Relation Browser Tool for Faceted Exploratory Search. *Bulletin of the IEEE Technical Committee on Digital Libraries* 5(1), Spring 2009.
- [17] R. E. Tarjan. Fast algorithms for solving path problems. *J ACM*, 28(3):594–614, 1981.
- [18] R. W. White and R. A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan and Claypool, 2009.
- [19] T. Tran, H. Wang, S. Rudolph, P. Cimiano. Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. In *ICDE 2009*.
- [20] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. In *VLDB 2005*.
- [21] <http://knoesis.wright.edu/library/ontologies/swetodblp/>
- [22] <http://www.w3.org/RDF/>
- [23] <http://www.w3.org/TR/owl-features/>