

CoSi: Context-Sensitive Keyword Query Interpretation on RDF Databases

Haizhou Fu
North Carolina State
University, Raleigh, NC
hfu@ncsu.edu

Sidan Gao
North Carolina State
University, Raleigh, NC
sgao@ncsu.edu

Kemafor Anyanwu
North Carolina State
University, Raleigh, NC
kogan@ncsu.edu

ABSTRACT

The demo will present CoSi, a system that enables context-sensitive interpretation of keyword queries on RDF databases. The techniques for representing, managing and exploiting query history are central to achieving this objective. The demonstration will show the effectiveness of our approach for capturing a user's querying context from their query history. Further, it will show how context is utilized to influence the interpretation of a new query. The demonstration is based on DBpedia, the RDF representation of Wikipedia.

Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Search Process; H.2.4 [Database Manager]: Query Processing

General Terms

Algorithms, Measurement, Performance, Experimentation

Keywords

Query History, Keyword Query Interpretation

1. INTRODUCTION

Keyword queries are an easy-to-use interface for (semi-)structured databases. However, the task of interpreting keyword queries can present various challenges due to their ambiguous nature. Labeled graph models such as RDF databases permit labels on data and metadata and support simultaneous querying of data and metadata. Consequently, in order to process keyword queries on RDF databases effectively, effort must be made to distinguish the roles of the keywords in a query. This demands a more sophisticated approach than the *meaning-as-match* interpretation paradigm used in IR and Web Search where the meaning of a query is considered as a set of objects containing “*matches*” to keywords. As an example, consider a query containing the keyword “*River*”. This keyword could be part of a label of an attribute (e.g., a street address “*765 River Bank Rd*”) for a specific location entity or the label of a particular river entity “*Mississippi River*”. On the other hand, it could also be *interpreted* as a concept or class indicating “*a large body of water*”, which will need to be interpreted as its set of instances, many of which do not necessarily contain matches

to the keywords, e.g., “*the Amazon*” or “*the Nile*”. Therefore, interpreting keyword queries in this context requires disambiguating the meaning of each keyword by identifying its specific role in the query. Correct identification of roles for keywords will enable the appropriate return variables and conditional expressions to be constructed as part of the structured query generation process preceding query processing.

There have been some recent approaches [2, 4, 5] focused on this problem of keyword query interpretation that utilize ranking schemes to rank the most likely intended interpretation higher in the list of interpretations presented to users. However, their ranking techniques are based on properties of the database which do not capture user characteristics and context. As a result, these approaches provide the same interpretation for the same set of keywords regardless of user intended meaning. One direction that remains unexplored is the use of a user's query history to ascertain querying context and hence guide the interpretation process. Techniques that use query logs [3] have been widely adopted in the area of IR. However, to the best of our knowledge, the idea of storing and managing history of structured queries has not been considered previously.

Motivating Example: We exploit the observation that users tend to issue conceptually related queries in close sequence. Therefore, exploiting query history can identify the “*most likely intended*” interpretations for a new query. For example, in Figure 1, given a keyword query “*Mississippi River Bank*”, where “*Mississippi*” has three occurrences, “*River*” has four and “*Bank*” has five, existing techniques will always select the interpretation “*part of the name of a bank*” as the most likely intended ones. The reason is that they are in the same database term and has the smallest connection subgraph. However, if a user had previously queried about “*Mortgage Loan*”, then it is more reasonable to select the interpretation of the current query as being that of the financial institution “*Mississippi River Bank*”. On the other hand, if a user's previous query was “*Fishing Techniques*” it may make more sense to interpret the current query as referring to the “*bank of the Mississippi River*”.

Contributions: We present a system CoSi that implements *context-sensitive keyword query interpretation* on RDF databases. Techniques (including a sophisticated index strategy, a dynamic context representation model and a context sensitive query generation algorithm) are employed to endow CoSi with the ability to “*sense*” the querying context and therefore, to increase the likelihood of generating the most likely intended interpretation.

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.

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

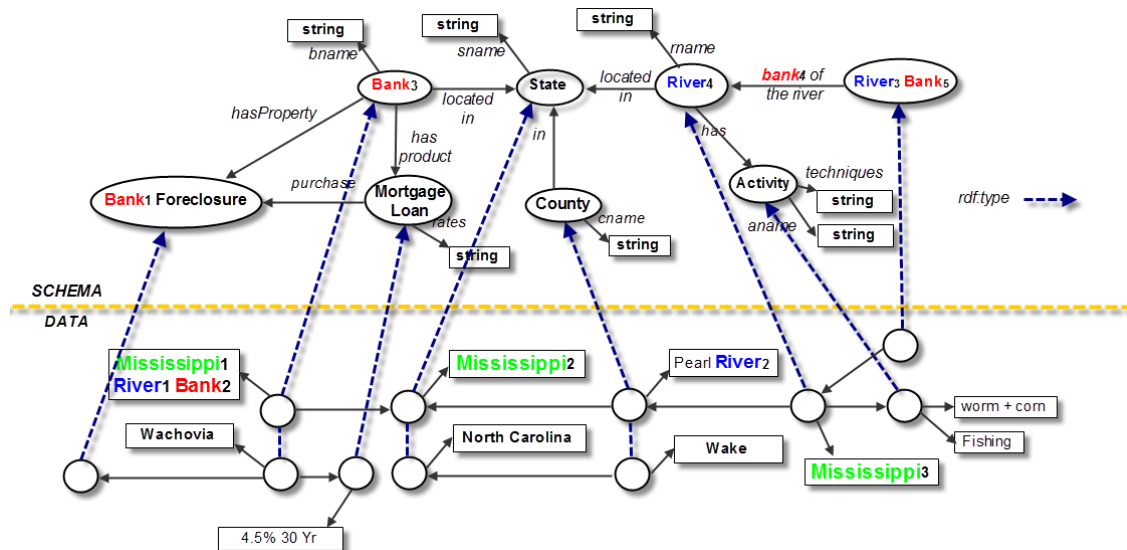


Figure 1: RDF Schema and Data Graph (the numbers after those keywords are serial numbers showing different matches of those keywords)

2. SYSTEM ARCHITECTURE

The *CoSi* system architecture consists of three core components: an *indexer*, a *context-sensitive cost model* and a *query interpreter*.

2.1 Indexer

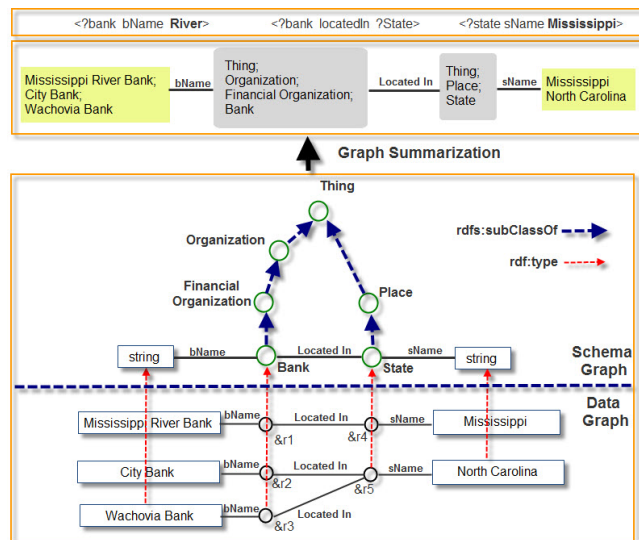


Figure 2: Graph Summarization

There are two types of index that are maintained by the Indexer. The *Summary Graph Index* is a *graph summarization* of a RDF data graph and its schema graph, called *summary graph*. Figure 2 shows an example of a summary graph. In the summary graph, each internal node is a hyper-node that contains the labels of a leaf schema class and all labels of its super-classes following the class hierarchy tree. Every literal node is also a hyper-node containing all literal values of a particular attribute. As is shown in Figure 2, one

literal node in the summary graph includes all literal values for attribute “*bName*”, which means it contains the names of all banks. The advantage of using summary graph is that it is a much smaller graph than a data graph, and candidate query interpretations can be generated as subgraphs of the summary graph. The pattern shown at the top of Figure 2 is an example of graph pattern query that is equivalent to this summary graph for a keyword query “*Mississippi River Bank*”.

The other type of index, i.e., *SL-Trie* index is an advanced inverted list structure for fast matching from keywords to summary graph elements(nodes/edges). More importantly, the *SL-Trie* offers an efficient way to identify the most relevant (most related to the context) matches of a particular keyword. Since such a relevance based ranking of keyword matches is dynamic due to the constantly evolving context, *SL-Trie* provides an effective mechanism to maintain this dynamic information. The cost model to evaluate the relevance between matches and the context will be discussed in the next sub-section. The discussion of the algorithm for keeping the dynamic index updated will not be elaborated in this proposal.

2.2 Cost Model

The goal of the cost model is to assign weights to the summary graph such that the weights reflect the impact of the context on the summary graph. The degree of relevance between the context (i.e., the past query interpretations) and the summary graph is determined by two factors: *chronological age* and *conceptual scope*. Intuitively, *summary graph elements*(nodes/edges) hit by more recent queries (with smaller chronological age) should have higher weights. Further, we prefer to give higher weights to the summary graph elements in the close proximity of a query in the querying context (within the conceptual scope). To achieve this overall effect, our cost model records the impact of a previous keyword query K_t (issued at time t) on its directly hits and closely related summary graph elements. Further, the

strength of this impact must be maximal at time t but must decay as the K_t ages out of the query history.

Consequently, our cost model consists of two factors: the *historical impact factor* $hif(t)$ and the *region factor* $rf(d_t)$. The region factor is a decreasing function of distance d_t between a summary graph element and a previous query; The historical factor is designed as a decreasing function of chronological age, and the older a query is, the lower the impact it exerts.

2.3 Query Interpreter

The query interpreter is responsible for generating the context-sensitive top-K structured query interpretations for a given keyword query K_t , at time t . Given a weighted summary graph SG_t , where the weights have been dynamically updated to reflect the user’s current query context, this context information can then be used to bias the graph exploration procedure. We propose a **CO**ntext **A**ware **B**idirectional **E**xpansion algorithm, called *CoaBe*, that identifies an near-optimal Minimum Steiner tree (MST). The *CoaBe* algorithm tries to construct a sub-tree of the summary graph that a) connects every hit of the keywords in the given keyword query; b) the total cost of the sub-tree is the minimum one only among all the other trees that satisfy condition a). Further there is a significant difference between *CoaBe* and traditional graph exploration algorithms such as [4]. For a given keyword and its two matches, *CoaBe* gives the one with relatively higher degree of relevance to the context a higher velocity during the graph expansion phase. This special feature enables a preference for keyword matches that are more related to the querying context.

2.4 Workflow

Generally, as shown in Figure 3, the system works in an iterative way. Given an RDF data/schema graph, a *summary graph* is the first to be built by the indexer. As is shown in Figure 3, the weights are assigned/re-assigned to nodes/edges in the summary graph by using the cost model: $SG_{t+1} = w(SG_t, Q_t)$, where Q_t is the last query interpretation in the query history. Given a keyword query K_t and the *weighted summary graph* SG_t , the *Query Interpreter* is executed to generate new candidate top-k queries. After that, the top-1 query interpretation (or user specified interpretation) is passed to the indexer and the cost model will update the weighted summary graph for next search itera-

tion (i.e., user issues a new keyword query). At the initial stage, if there is no queries in the query history, each graph element has equal weight. A user can choose to select the best interpretation from those candidates.

3. DEMONSTRATION

We will demonstrate that the CoSi system can improve the quality of the keyword query interpretation task by using query history as context information. The system will be demonstrated using a very large real-world dataset DBpedia [1]. CoSi is a desktop application that will be run locally using a laptop or desktop PC. The end users can interact with several features of the system.

3.1 System Interface

Figure 4 shows the graphical user interface. By default, CoSi enables a context-sensitive interpretation mode. Users can also manually change the mode by clicking the “*Search Mode*” button. The users can start a new session by choosing “*New Query Session*” on the menu. By clicking the “*New Search*” button, user can start issuing keyword queries and browse the results. After user click the “*Interpret*” button, the system will generate a ranked list of up to five candidate interpretations. By selecting one of the candidate interpretations, the corresponding SPARQL query will be shown in the text box under the “*Interpret*” button. Users can also view any interpretation from the list and a sub-graph that is equivalent to the interpretation of the keyword query will be shown on a panel in the main interface. Any information related to issued queries such as keywords, candidate interpretations (subgraphs and SPARQL queries) will be recorded. User can select to view the information of any query in the query history by clicking that query in the list shown on the top-right carousel panel control. The entire history of query interpretations can be saved by using the “*Save Query Session*” function on the main menu. CoSi also provides users a “*weights viewer*” to see the subgraph of the weighted summary graph. The size of the red circle in the node reflects the value of the weight. The subgraph contains nodes and edges related to all the queries in the query history. The users can observe the changes of subgraphs caused by the evolving context. Finally, the end users can choose to investigate the results of using context-agnostic mode, where the traditional approach is applied.

3.2 Demonstration Scenarios

This demonstration will show you some interesting features of the CoSi system. We will demonstrate how CoSi can provide effective solutions to compensate for limitations of traditional interpretation system. Three interesting user cases are given:

Scenario A: The end users can choose to issue a sequence of related keyword queries under different search modes (context-sensitive and context-agnostic). In this way, they can observe how queries in recent query history influence the interpretation of newer queries.

Scenario B: Scenario *B* shown in Figure 5 illustrates a more interesting feature of CoSi. Given a keyword query “*Apple Manhattan*”, assume that two possible interpretations are $I - A$ and $I - B$. Given two different query histories (The end users can load existing query logs by clicking “*open query session*” from the main menu), CoSi ranks

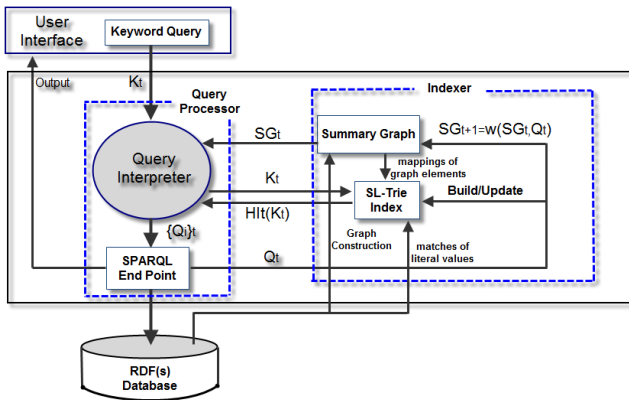


Figure 3: System architecture and Workflow

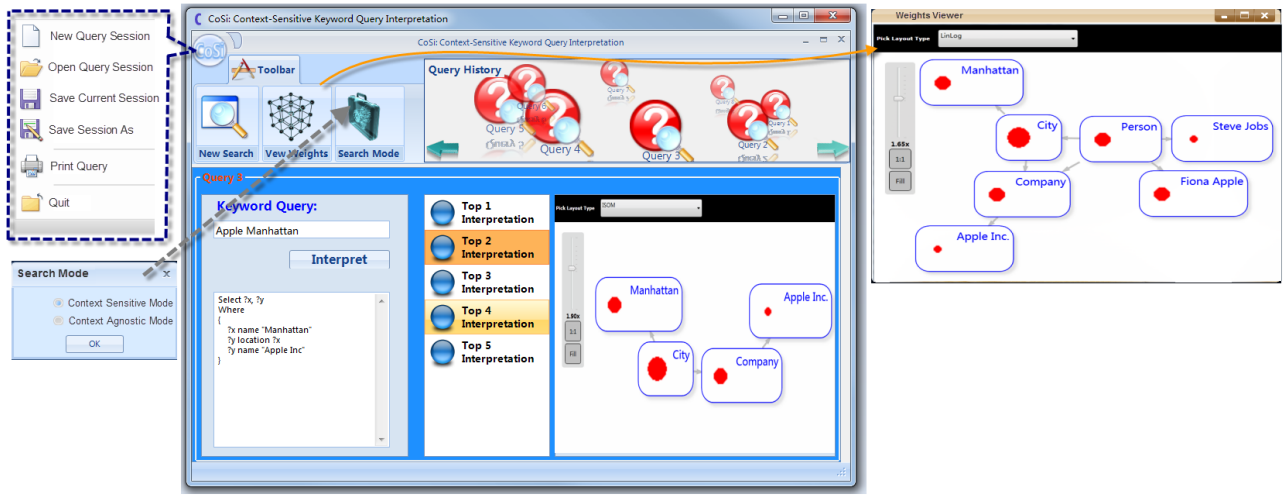


Figure 4: Graphical User Interface

	Query History 1	Query History 2
Query 1:	Tigal	Mac OS
Query 2:	When the Pawn	Iphone App Store
Conceptual Scope of the Context:	Artist, Album	Company, Products
Query 3 (Current Query)	Apple Manhattan	
Candidate Interpretations		
I-A:	Singer named Fiona Apple who was born in Manhattan	
I-B:	Headquarters of Apple Inc. in Manhattan	

Figure 5: Scenario B: Different contexts different rankings

	Query History 1	Query History 2	Query History 3
Query 1:	Tool	Tool	Tool
Query 2:		Model Database	Model Database
Query 3:			Entity Relationship
Most Recent Query:		Rose	
Ranking of Intended Interpretation	>5	2	1
Intended Interpretation:	A software Rational Rose for ER diagramming		

Figure 6: Scenario C: The impact of length of query history on interpretation

I – A higher by loading “Query History 1” because the conceptual scope of this search session is about “artist” and “album”. Alike, CoSi ranks *I – B* higher if “Query History 2” is loaded because the context is all about “company” and “products”. Therefore, CoSi can detect the context which reflects the users’ focus and interests during the specific query session, while traditional methods (using context-agnostic mode) will always return the same ranking regardless of the context.

Scenario C: Another important capability of CoSi is demonstrated by scenario *C* shown in Figure 6. The CoSi system is capable of learning the knowledge from query logs. The longer the query history the more information can be digested and utilized by the CoSi system, and therefore, the disambiguation process will be more accurate. In this scenario, three query logs are given, and as is shown in Figure 6,

each log contains one more keyword query than the previous one. By looking at the target query “Rose”, there is no idea what the users want to ask, which is a usual case that the users themselves sometimes do not know how to issue a query for what they really want. Therefore, they may issue a sequence of queries for a clue. But traditional interpretation systems are not helpful because they are not context-sensitive. Users still need to sift through the search results. But with CoSi, when people search in an exploratory way by issuing serial queries, CoSi will learn what they are really asking for and rank the intended interpretation higher such that the end users can find them more easily.

4. ACKNOWLEDGEMENT

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

5. REFERENCES

- [1] AUER, S., BIZER, C., KOBILAROV, G., LEHMANN, J., CYGANIAK, R., AND IVES, Z. G. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC* (2007), pp. 722–735.
- [2] KASNECI, G., RAMANATH, M., SOZIO, M., SUCHANEK, F. M., AND WEIKUM, G. Star: Steiner-tree approximation in relationship graphs. In *ICDE* (2009), pp. 868–879.
- [3] SHI, X., AND YANG, C. C. Mining related queries from search engine query logs. In *WWW* (2006), pp. 943–944.
- [4] TRAN, T., WANG, H., RUDOLPH, S., AND CIMIANO, P. Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *ICDE* (2009), pp. 405–416.
- [5] WANG, H., ZHANG, K., LIU, Q., TRAN, T., AND YU, Y. Q2semantic: A lightweight keyword interface to semantic search. In *ESWC* (2008), pp. 584–598.