International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-2, Issue-4,April 2016
ISSN: 2395-3470
www.ijseas.com

Implementation of Keyword Search Mechanism on RDF Graph Model

Manisha Bhaik [1], Shyam Gadekar [2], Nikhil Gumaste [3], Laxmikant Suryawanshi [4]
[1234] JSPM's Imperial College of Engineering & Research
Wagholi, Pune-412207, India.

Abstract

We present the implementation of a keyword based querying system operating on RDF databases. As in various search technique keyword is used which provides a simple but user friendly interface to retrieve information from complicated data structure. Most of these knowledge bases adopt the Semantic-Web data model RDF as a representation model. Querying these information bases is classically done using structured queries utilizing graph-pattern languages like as SPARQL. However, queries require some expertise from users which limits the accessibility to such data sources. To overcome this drawback, keyword search will be supported. This paper used indexing, pruning and refinement phases. This method provides efficient result for searching keyword on graph. Approximate mining algorithms can be used to form sub graph from RDF graph data based on scores at the level of keywords, data elements, element sets, and sub graphs that join these elements. To retrieve the well-organized keyword from sub graph keyword matching algorithm can be used for graph data. The purpose of this technique is to reduce the high cost of processing keyword search queries on graph information and get better performance of keyword search, without compromising its result quality. Also, it reduces processing time for keyword search in RDF graph data.

Keywords: *Data Mining, RDF Graph, Semantic Web, SPARQL, Keyword Search.*

1. Introduction

In various real world applications, RDF (Resource Description Framework) has been widely used as a W3C standard to describe data in the Semantic Web. RDF data may often suffer from the unreliability of their data sources, and exhibit irregularity or errors. In this paper, we model such unreliable RDF data by probabilistic RDF graphs [1] and learn a vital problem; keyword search query over probabilistic RDF graphs (i.e. the pg-KWS query). To retrieve meaningful keyword search results, we implement the score rankings for sub-graph answers specific for RDF data.

Now a day, keyword search is the leading technique of searching a data source such as the Web. Using solely keywords, i.e., a small number of highly perceptive terms the consumer anticipates that she will identify the web pages most relevant to her information needs. Keyword search offers a straightforward, intuitive, and yet flexible method of retrieving information. The success of keyword search in the field of Information Retrieval (IR) and the World Wide Web (WWW or just Web) has generated awareness in keyword search interacts to relational databases and similar structured and semi structured data sources. This is the way how keyword search has evolved over time and it has been adopted by different fields in computer science, such as IR, databases, and semantic web, and surveys the state of the art in keyword search for fields managing structured and semi structured data. Beyond that, it presents and extensively evaluates the design and implementation of a system working on RDF data, accommodating keyword queries with temporal constraints. At last, it presents the state of the art in assessment of keyword search system working on structured and semi structured data.

Query processing over graph data has attracted considerable attention recently as an increasing amount of data which is available on the web, XML data sources and relational sources can be modelled in the form of graphs. RDF as a framework for web resource description appears to have gained a larger impetus on the web and an increasing collection of repositories of data are modelled using RDF framework.

Notable examples are Biological Databases, Personal Information Systems where e-mails, papers and images are merged into a graph and Enterprise Information Management (EIM) systems like launch vehicle blueprint information where details of vehicle parameters and stage sequence events is modelled as graph data. The large size and complication of data sets in these domains makes their querying a difficult job.

The keyword search over RDF graph is useful in applications in Semantic Web. RDF graph consist of RDF resources, RDF schema, and their vertices related to information of keyword. SPARQL is standard language of RDF graph. In semantic web, during data extraction/integration, data contains errors or a problem of data inconsistency because of data contains irregular format or unstructured texts. Also, there are various types of information extraction methods.

Because of unreliability of data, we integrate with RDF graphs and keyword search becomes efficient. Therefore we call the RDF graph as also probabilistic graph.

Example: YAGO data set [20] which contains probabilistic RDF data integrated from WordNet and Wikipedia. RDF triples which are (*subject, predict, object*) or (S.P.O.).

2. System Description

2.1 Related Work

RDF has been used in the data mining for accurate keyword search with the help of the different data schema of RDF data, and improves the performance throughout the process in the lifecycle.

SPARQL query is a standard SQL-like query language for RDF data. For knowing the SPARQL query, have to know schema of RDF data, including the subject, predicate, or object. It has used several data models, like as triple store [4], [17], [23], Column-store [2], [18], [19], property tables, [24], [25], and graphs [3], [21].

Works held previously have the problems of efficiency and processing on the data. Already, there existing works on searching methods like as r-Radius Graph from EASE [12] keyword Search Method for all type of data. From the works IR-based ranking score functions [21], [1] like as matching and popularity score.

Probabilistic RDF databases works done by graph representation relation by Fukushige[7] in Bayesian network. The work done in the SPARQL [9] queries as an alternative of keyword search which will provide high flexibility and performance in Lian and Chen [15] has the efficient query answering with RDF Schema.

Several probabilistic queries for unstructured data have projected, as a probabilistic range query (PRQ) [6], nearest neighbor (PNN) [5], and reverse nearest neighbor (PRNN) [14].

Top-k queries [13] gives retrieve tuples with variety of probabilities and score. Keyword Search with Graph gives more ease with searching throughout the databases. Existing works provides us significant query keyword, and different levels of abstraction from graphs. Tree with root r [8], [11] and other leaf node contains the query keywords. Ranking Score has been calculated with path length. BANKS [10] has backward search method, which will traverse thoroughly with link with predecessor.

Above, all works states that there have keyword search over graphs.

2.2 Existing System

In existing system , Clustering Large Probabilistic Graphs: Problem of clustering probabilistic graphs is identical to the problem of clustering standard graphs, probabilistic graph clustering has numerous applications, like finding complexes in probabilistic protein-protein interaction networks and discovering groups of users in affiliation networks. The edit-distance based definition of graph clustering to probabilistic graphs. Establish a connection between objective function and correlation clustering to propose practical approximation algorithms for problem.

A benefit of approach is that objective function is parameter-free. Therefore, the number of clusters is part of the output. It also develops methods for testing the statistical significance of the output clustering and study the case of noisy clustering.

Using a real protein-protein interaction network and ground-truth data, methods discover the correct number of clusters and identify established protein relationships.

Finally, the practicality techniques using a large social network of Yahoo! users consisting of one billion edges.

2.3 Propose System

We propose powerful pruning techniques (by means of disconnected from the net pre-processed score limits and probabilistic edge) to rapidly sift through false cautions. Broad trials have been led to check the viability and productivity of our proposed approaches. It proposed to answer watchword look questions on specific charts. We can change probabilistic RDF diagram with indeterminate vertex/edge watchwords to the one with dubious catchphrases in vertices just, to which we apply our proposed approaches.

We will use this entropy idea to propose a metric that can demonstrate our inclination to RDF watchword query items in probabilistic RDF charts. We will propose two pruning techniques, **score bound pruning** and **probabilistic pruning**, which use score limits or probabilistic edge, separately, to empower the pruning. Our proposed pruning strategies by means of score limits.

We propose **a heuristic based calculation**, which gets ***w*-PWG**s of a probabilistic sub *graph g* with ease. We report the exploratory consequences of our proposed approaches for noting pg-KWS questions on both genuine and engineered information.

The proposed procedures investigated pruning techniques with chart structures and coordinating probabilities. different probabilistic inquiries over dubious information have been proposed, including *probabilistic range query (PRQ) , nearest neighbor (PNN (PNN), reverse nearest neighbor (PRNN) , skyline (PSQ) , reverse skyline (PRSQ) , and similarity join (PSJ).*The bidirectional inquiry, which utilizes the pre-figured separation in the middle of catchphrases and hubs, and gets the limits of positioning scores to empower quick pruning and recovery. Watchword seek has been finished up to recover helpful information from database. *Catchphrase* look has significant advantage i.e. it is anything but difficult to work. Like In informal organizations, every connection between any two persons is regularly joined with a likelihood that speaks to the vulnerability of the connection or the quality of impact a man has over someone else in viral advertising.

XML information having tree or chart structure, vulnerabilities are coordinated in XML records known as probabilistic XML report. The Catchphrase seeking in RDF information, interpersonal organizations and XML information have numerous profound applications. For information with XML and relational construction, particular inquiry dialects, for example, SQL and XQuery have been developed for data recovery. Keeping in mind the end goal to inquiry such information, the client must ace an intricate question dialect and comprehend the hidden information mapping.

Both XML databases and relational databases can be seen as graphs. In particular, XML datasets can be viewed as charts when IDREF/ID connections are thought about, and a social database can be viewed as an information diagram that has *triplet* and *watchwords* as ***node*s**.

3. System Architecture, Algorithms, Modules, Mathematical Model, Results and System Performance

3.1 System Architecture

The following diagram *Fig.1* shows the system architecture of the RDF Framework. It shows the processing models of all the state of input and output throughout the whole execution process. The system architecture shown below,

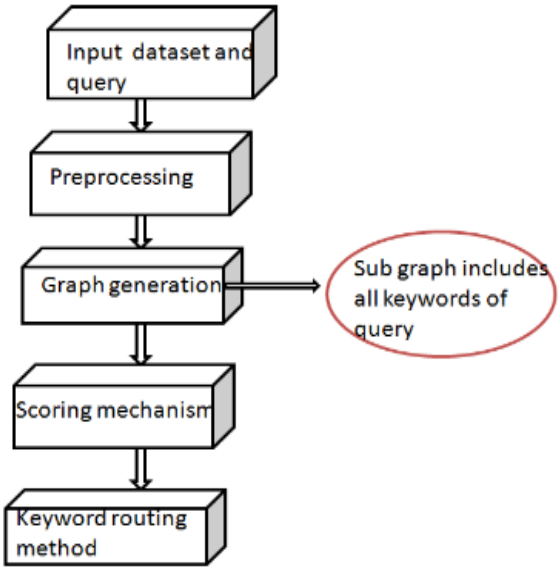

Fig.1 System Architecture

In the framework consists of three phases, *indexing*, *pruning*, and *refinement* phases.

1. Indexing Phase: We will offline extract probabilistic *r-radius* graphs from the probabilistic RDF graph, and precompute data for each graph. Then, we construct an index over these precomputed data for probabilistic *r-radius* graphs, which will be used later for online pruning and pgKWS query answering.

2. Pruning Phase: Given any pg-KWS query, the second pruning phase traverses the index, and meanwhile applies pruning methods to quickly rule out false errors (i.e. those sub graphs that cannot be pg-KWS query answers). In particular, we will intend two pruning strategies, ***Score bound pruning*** and ***Probabilistic pruning***, which utilize score bounds or probabilistic threshold, respectively, to enable the pruning. After the index traversal, we can obtain a candidate set S_{cand}.

3. Refinement Phase: Finally, in the refinement phase, we refine candidates in S_{cand} by checking the condition and return the actual pg-KWS answers.

a. We will use following methods for pg-KWS Query Processing,
b. Index Construction with Index Nodes Procedure
c. Pruning
d. Query

3.2 Algorithm

Inspired by the complexity of finding a good PWG partitioning, we propose *a heuristic based algorithm*, which obtains *w-PWGs* of a probabilistic *subgraph g* with low cost. We execute the randomized algorithm for iter rounds, and select the one with the lowest cost as the output of function *F(•)*.

Our PWG partitioning algorithm will iteratively obtain PWG partitioning strategy *PS(i; j)*,as well as its corresponding cost *PS(i; j)*:cost. Specifically, the algorithm maintains a cost matrix, cost matrix, in which each element cost matrix*[i][j] = PS(i; j):cost for i ≤ j.*

3.3 Modules

1. *Keyword searching based Social Authentication Module:* The system prepares keyword searching for a user registration in this phase. Specifically, this is first authenticated with her main authenticator (i.e., password) and then a few friends, who also have accounts in the system, are selected by either herself

or the service provider from registration friend list and are appointed as Keyword searching.

Admin Phase: The Admin is using for files upload for system. Specifically, this is which file is more than seeing is ease to find out.

2. Search Module:

We use a measure, which is the (summed) difference between upper and lower score bounds for keyword pairs (or keywords).Different from leaf nodes, aggregates in non-leaf nodes summarize bounds or keyword existence information for all possible worlds of graphs (rather than w PWGs). Thus, we do not need to record existence.

Intuitively, probabilistic *r-radius* graphs with similar keywords tend to have similar score bounds. This way, tree nodes on different levels can be iteratively built until a final root is obtained we randomly generate m possible keywords (each keyword is hashed to an integer within range [1; 100]), as well as conditional probability tables (CPTs) that depend on vertex keywords from its incoming edges, where m is randomly picked up within [1; 5] by default.

We tested two types of distributions for uncertain vertex keywords, Uniform and Skew and denote their corresponding data sets as Uniform and Skew, respectively.

3. Graph Module:

We first generate a schema graph, by producing vertices that represent RDF entities, and randomly connecting vertices via directed edges, where the averages of vertex out-degrees and in-degrees are set to by default.

3.4 Mathematical Model

Assumptions:

S: System; A system is defined as a set such that:

S = {I, P, O}.

Where,

U: Set of users
= {UR: Set of Registered Users,
UN: Set of Un-Registered Users}

I: Set of Input.
O: Set of output.
P: Set of Processes.

Input Set Details:

1. *PHASE 1: InputModule.*
   Ir= { username: i1,
   Dataset: i2}

Data Size in Record Count	Proposed System time	Existing system time
10000	2.3	5.3
20000	3.3	6.84
30000	4.6	8.7
40000	5.71	9.3
50000	7.2	10.6

2. PHASE 2: PAGE MGMT
   Iv= { *username*: i1,
   messages: i2,
   partition: i3,
   }

Process Set Details:

1. *PHASE 1: InputModule.*
   P1={ User registration: p11}

2. PHASE 2: QUERT PROCESSING
   P2={ Data computation: p21,
   Data processing:p22,
   graph partition,: p23}

3. PHASE 3: Result
   P3={ SR_ partition: p31,
   SR_computation: p32,
   SR_ processing: p33}

Output Set Details:

1. *PHASE 1: InputModule.*
   O1={userid: o11,
   Password: o12}

2. PHASE 2: QUERT PROCESSING
   O2={dataClassification: O21}

3. PHASE 3: Result
   O3={DR_Statistic : o31,
   DR_Result : o32}

3.5 Results

Diverse characteristics, (*for example, the dataset count, CPU time, Performance or Precision*) which may be favored by clients who need to consider their connections. From the semantics of our seek

answer defination, *Graph g*. Execution Time *versus* Genuine/Synthetic Data Sets.

We demonstrate the pg-KWS Query efficiency on genuine/engineered information in *Fig. 2*, with default settings. Our analyses demonstrate that our inquiry noting methodology can accomplish low CPU time, which shows the viability of our score bound and probabilistic pruning. Beneath, we will utilize manufactured information to test the heartiness of our pg-KWS approach by shifting distinctive parameter values.

Table.1 Data Set Values

Above table shows the data size in record count used for testing system results. Following bar graph shows the proposed system time versus existing system time.

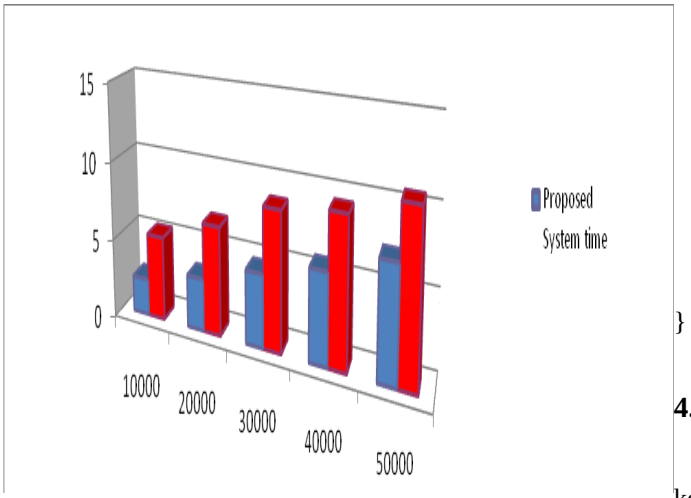

Fig.2 Resultant Proposed Output

3.6 System Performance

We execute the randomized algorithm for iter rounds, and select the one with the lowest cost as the output of function F(•). Our PWG partitioning algorithm will iteratively obtain PWG partitioning strategy *PS(i; j)*, as well as its corresponding cost *PS(i; j):cost*.

Specifically, the algorithm maintains a cost matrix, cost matrix, in which each element cost matrix*[i][j] = PS(i; j):cost for i ≤ j*.

Procedure:

Procedure pg_KWS_Processing{
Input: a probabilistic RDF graph database *G*, an index *I* over *G*, a set of *q* query keywords k_1, k_2,..., k_q, and a probabilistic threshold α
Output: subgraphs *g* from *G* that contain keywords

(1) initialize a min-heap $\mathcal{H}$ accepting entries in the form (*e*,*key*)
(2) S_{cand}=(/);
(3) Insert(*root(I)*,*0*) into heap $\mathcal{H}$
(4) while $\mathcal{H}$ is not empty
(5) (*e*,*key*)=de-heap $\mathcal{H}$
(6) if *e* cannot be pruned by score
(7) if *e* is a probabilistic *r*-radius graph *g*
(8) if PWGs of g can not be printed by probabilistic pruning
(9) add *g* to S_{cand}
(10) if *e* is a leaf node
(11) for each probabilistic *r*-radius graph *g* < *e*
(12) if *I* can not be pruned b score bound/probabilistic pruning
(13) insert(*g*,*key(g)*)into heap $\mathcal{H}$
(14) if *e* is non-leaf node
(15) For each entry ε_x< e
(16) if e_x cannot be pruned by score bound/probabilistic pruning
(17) insert(e_x,*key*(e_x)) into heap $\mathcal{H}$
(18) refine candidates in S_{cand} and report final pg-KWS query answers

}

4. Conclusions

We invent and undertake an important problem of keyword search over probabilistic RDF graphs, called pg-KWS queries. We will propose efficient pruning methods via offline pre-computed score bounds and probabilistic threshold to quickly filter out false errors. Furthermore, we construct a directory for the pre-computed data for RDF and present an efficient query resulting approach. General research has been conducted to verify the effectiveness and efficiency of our proposed approaches.

Acknowledgments

The authors wishes to thank Prof. Rashmi Sonawane (Guide), Prof. Darshika Lothe(PG–Coordinator), Prof. S.R.Todmal (HOD) and Dr. Sachin Admane (Principal) for valuable guidance and encouragement.

References

[1] Xiang Lian, Lei Chen, Member, IEEE, and Zi Huang, Member, IEEE "Keyword Search Over Probabilistic RDF Graphs" in Proc.IEEE Transactions on Knowledge and Data Engineering, Vol. 27, No. 5, May 2015, pp.1246-1260.

[2] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach, "Scalable semantic web data management using vertical partitioning," in Proc. 33rd Int. Conf. Very Large Data Bases, 2007, pp. 411–422.

[3] R. Angles and C. Gutierrez, "Querying RDF data from a graph database perspective," in Proc. 2nd Eur. Conf. Semantic Web: Res. Appl., 2005, pp. 346–360.

[4] M. Atre, V. Chaoji, M. J. Zaki, and J. A. Hendler, "Matrix "bit" loaded: A scalable lightweight join query processor for RDF data," in Proc. 19th Int. Conf. World Wide Web, 2010, pp. 41–50.

[5] G. Beskales, M. Soliman, and I. F. Ilyas, "Efficient search for the top-k probable nearest neighbors in uncertain databases," in Proc. 34th Int. Conf. Very Large Data Bases, 2008, pp. 326–339.

[6] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2003, pp. 551–562.

[7] Y. Fukushige, "Representing probabilistic relations in RDF," ISWC-URSW, pp. 106–107, 2005.

[8] H. He, H. Wang, J. Yang, and P. S. Yu, "BLINKS: Ranked keyword searches on graphs," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2007, pp. 305–316.

[9] H. Huang and C. Liu, "Query evaluation on probabilistic RDF databases," in Proc. 10th Int. Conf. Web Inform. Syst. Eng., 2009, pp. 307–320.

[10] A. Hulgeri and C. Nakhe, "Keyword searching and browsing in databases using BANKS," in Proc. 18th Int. Conf. Data Eng., 2002, pp. 431–440.

[11] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, "Bidirectional expansion for keyword search on graph databases," in Proc. 31st Int. Conf. Very Large Data Bases, 2005, pp. 505–516.

[12] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou, "EASE: An effective 3-in-1 keyword search method for unstructured, semi-structured and structured data," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2008, pp. 903–914.

[13] J. Li, B. Saha, and A. Deshpande, "A unified approach to ranking in probabilistic databases," Proc. VLDB Endowment, vol. 2, no. 1, pp. 502–513, 2009.

[14] X. Lian and L. Chen, "Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data," The VLDB J., vol. 18, pp. 787–808, 2009.

[15] X. Lian and L. Chen, "Efficient query answering in probabilistic RDF graphs," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2011, pp. 157–168.

[16] Y. Luo, X. Lin, W. Wang, and X. Zhou, "Spark: Top-k keyword query in relational databases," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2007, pp. 115–126.

[17] T. Neumann and G. Weikum, "RDF-3X: A risc-style engine for RDF," Proc. VLDB Endowment, vol. 1, no. 1, pp. 647–659, 2008.

[18] L. Sidirourgos, R. Goncalves, M. Kersten, N. Nes, and S. Manegold, "Column-store support for RDF data management: Not all swans are white," Proc. VLDB Endowment, vol. 1, no. 2, 2008, pp. 1553–1563.

[19] M. Stonebraker, D. J. Abadi, A. Batkin, X.Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, and S. Zdonik, "C-store: A column-oriented dbms," in Proc. 31st Int. Conf. Very Large Data Bases, 2005, pp. 553–564.

[20] F. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: A large ontology from wikipedia and wordnet," Web Semantic, vol. 6, no. 3, pp. 203–217, 2008.

[21] T. Tran, H. Wang, S. Rudolph, and P. Cimiano, "Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data," in Proc. IEEE Int. Conf. Data Eng., 2009, pp. 405–416.

[22] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. Hellerstein, "Bayestore: Managing large, uncertain data repositories with probabilistic graphical models," in Proc. Int. Conf. Very Large Data Bases, 2008, pp. 340–351.

[23] C. Weiss, P. Karras, and A. Bernstein, “Hexastore: Sextuple indexing for semantic web data management,” Proc. VLDB Endowment, vol. 1, no. 1, pp. 1008–1019, 2008.

[24] K. Wilkinson, “Jena property table implementation,” in Proc. Scalable Semantic Web Knowl. Base Syst., 2006, pp. 35–46.

[25] K. Wilkinson, C. Sayers, H. Kuno, D. Reynolds, and J. Database, “Efficient RDF storage and retrieval in Jena2,” in Proc. Eur. Semantic Web Databases workshop, 2003, pp. 131–150.