EFFICIENT KEYWORD SEARCH ON LARGE RDF DATA USING OPTIMIZATION TECHNIQUE

Leya Zacharias[1], Neema George[2]

Department of Computer Science and Engineering, Mahatma Gandhi University, Mangalam College of Engineering,Kerala

Email id: leyaaza@yahoo.com

9496736469

Abstract- Now a day's keyword search in data mining is very emerging topic. Latest keyword search techniques on Semantic Web are moving away from shallow, information retrieval-style approaches that merely find "keyword matches" towards more interpretive approaches that attempt to induce structure from keyword queries. Exploiting identity links among RDF resources allows applications to efficiently integrate data. Keys can be very useful to discover these identity links. A set of properties is considered as a key when its values uniquely identify resources. However, these keys are usually not available. The approaches that attempt to automatically discover keys can easily be overwhelmed by the size of the data and require clean data. By using summarization algorithm the RDF data can be summarized. Here searching is done with optimization technique, so the result is accurate and efficient and also time complexity can be reduced. Unlike other techniques, our search algorithms always return correct results. For making the searching more efficient and get the accurate result within the time bound genetic algorithm is used.

Keywords— Keyword search, RDF data, Genetic algorithm.

INTRODUCTION

Data mining becomes an essential research area in the field of computer science, as it helps much more in keyword searching .In recent day's more and more data is provided in RDF format. Storing large amounts of RDF data and efficiently processing queries on these types of data is becoming very crucial. .But it is not always necessary that they will provide the appropriate result that they are actually looking for. Now a day's more and more data is provided in RDF format, storing large amounts of RDF data and efficiently processing queries on such data is becoming crucial. .but it is not always necessary that they get the appropriate result that they searching for.

The amount of data published on the Semantic Web has grown at increasing rates in the recent years. This is mainly happened due to the activities of the Linked Data community and the adoption of RDF by major web publishers. The number of data to be managed is stretching the scalability limitations of triple stores that are conventionally used to manage Semantic Web data. In the same time, the Semantic Web is increasingly reaching end users who need efficient and effective access to large subsets of this data. These type of end users prefer simple, but ambiguous natural language queries over highly selective, formal graph queries in SPARQL, the query language of triple stores. In a web search scenario, formulating SPARQL queries may not be feasible altogether due to the heterogeneity of data.

The RDF (Resource Description Framework) is the de-facto standard for data representation on the Web. It is no surprise that we are inundated with large amounts of rapidly growing RDF data from disparate domains. For example, the Linked Open Data (LOD) initiative integrates billions of entities from hundreds of sources. Just one of these sources, the DBpedia dataset, explains more than 3:64 million things using more than 1 billion RDF triples; and it contains numerous keywords. The Resource Description Framework (RDF) may be understood as a common purpose, schema-flexible model for explaining meta data and graph-shaped information. RDF represents information in the form of statements (triples or quads). Every triple connotes an edge between two nodes in a graph. The quad position can be used to give statements identity or to place statements within a named graph. RDF gives some basic concepts used to model information - statements are composed of a subject (a URI or a Blank Node), a predicate (always a URI), an object (a URI, Blank Node, or Literal value), and a context (a URI or a Blank Node). URIs are used to identity a particular resource, whereas Literal values describe constants such as character strings and may carry either a language code or data.

Keyword-based queries over semi-structured data are an important function of modern information management. When this information is available as Linked Data, it can be abstracted as a graph. Generally speaking, the end results of queries over this type of data are sub-structures of the graph that contain the keywords searched in the nodes. The task of indexing and finding the individual nodes containing those keywords is relatively inexpensive and has well-established tools available for it.

Ascertaining the connections between those selected nodes is a decomposable problem. This involves expensive and time-consuming graph explorations. More than that it must be solved on-the-fly at query processing time. The scale at which those interrogations must be solved grows as more and more data is made accessible to the Web of Data. Over and above that, to address the possibility of merging databases in the future, and to execute queries that include databases that had originally different schemes. Those queries must trust only on the most basic format of the Linked Data model, the subject-predicate-object format.

First and most important objective is to evidence the related work done in keyword search. This is worth noting that it does not aim at surveying this field. As an alternative, it aims at giving the full picture of the evolution of keyword searching, initiated by the field of IR, and then adopted by the fields of web and databases. This is how each field has contributed to each other and in keyword searching severally. This way, the demands and trends of each research period can be identified and assessed, together with the research problems that each area has faced. The very next objective of the dissertation is to advance the state-of-the-art research in keyword search over RDF data. To this goal, the contributions of this dissertation lay on the design, implementation, and evaluation of a system supporting keyword searching over RDF data. The third and endmost objective is to shed light on the evaluation of systems and techniques targeting at keyword search over structured and semi structured data. Keyword search provide only an approximate description of the information items to be retrieved. Hence, the correctness of the retrieval cannot be formally verified, as it is the case with query languages, such as SQL. Alternatively, retrieval effectiveness is measured by user perception and experience.

The efforts of this work are twofold:

1. Provide a mechanism to store sizable RDF graphs in a distributed way. we developed a mechanism that partition and summarize (shards) the RDF graph and persist data in separate, distributed data stores.

2. Provide a measurable keyword-based search mechanism for RDF graphs. We use optimization processing mechanism to provide a scalable result to the task of building keyword indexes for RDF datasets.

The rest of the paper is organized as follows: in Section 1,we describe the related works. In Section 2, we present the search using genetic algorithm. In Section 3, reports extensive experimental results to support the proposed searching using genetic algorithm . Finally, in Section 4, we summarize the present study and draw some conclusions.

I. RELATED WORK

Conventional keyword search engines are limited to a specific information model and cannot easily get used to unstructured, semi-structured or structured data. This paper projected an effective and merging keyword search method, called EASE[6], for indexing and querying large collections of heterogeneous data. To fulfill high efficiency in processing keyword queries, we first model unstructured, semi-structured and structured data as graphs. After that summarize the graphs and create graph indices as an alternative of using traditional inverted indices. This proposed an extensive inverted index to facilitate keyword-based search. After all this present a novel ranking mechanism for enhancing search effectiveness. It have conducted an extensive experimental study using real datasets. The final results show that EASE achieves both high search efficiency and high perfection, and outperforms the existing approaches drastically.

Query dispensation over graph-structured information is enjoying a growing number of applications. A top-k keyword search query on a graph ands the top k answers according to some ranking criteria [2]. Here each answer is a substructure of the graph containing all query key-words. Present techniques for supporting such queries on general graphs suffer from several drawbacks. For example poor worst-case performance, not taking full benefit of indexes, and high memory requirements. To deal with these problems, it proposed BLINKS, a bi-level indexing and query processing scheme for top-k keyword search on graphs. BLINKS are follows a search strategy with provable performance bounds, while additionally exploiting a bi-level index for pruning and accelerating the search. To minimize the index space, BLINKS partitions a data graph into blocks: The bi level index stores summary information at the block level to initiate and guide search among blocks, and more detailed information for each block to accelerate search within blocks.

Applications in which bare text coexists with structured information are enveloping. Commercial relational database management systems (RDBMSs) generally provide querying capabilities for text attributes that incorporate state-of-the-art information retrieval (IR) relevance ranking strategies. But this search functionality requires that queries specify the exact column or columns against which a given list of keywords is to be matched [5]. This necessity can be cumbersome and unbendable from a user point of view. Perfect answers to a keyword query might require to be "assembled" –in perhaps unpredicted ways– by joining tuples from multiple relations. This inspection has stimulated recent research on free-form keyword search over RDBMSs. This paper adapts IR-style document-relevance ranking strategies to the problem of processing free-form keyword queries over RDBMSs. This query model can handle problems with both AND and OR semantics. The same exploits the sophisticated single-column text-search functionality often available in commercial RDBMSs. It develops query-processing measures that are constructed on a crucial characteristic of IR-style keyword search: only the few most relevant matches –according to some definition of "relevance"– are generally of interest. As a result, rather than computing all matches for a keyword query, which leads to inefficient executions, this techniques focus on the top-k matches for the query, for moderate values of k.

With the mounting volume of text information stored in relational databases, there is a huge demand for RDBMS to support keyword queries over text data [4]. As a search result this often assembled from multiple relational tables, traditional IR-style ranking and query evaluation methods cannot be applied directly. In this paper we will study the effectiveness and the efficiency issues of answering top-k keyword query in relational database systems. This will propose a latest ranking formula by adapting existing IR techniques based on a natural notion of virtual document. Compared with prior approaches, this new ranking method is very simple but effective. And all these agree with human perceptions. It studied effective query processing mechanism for the new ranking method, and propose algorithms that have minimal accesses to the database. It carried out extensive experiments on large-scale real data-bases using two popular RDBMSs. These results demonstrate strategic progress to the alternative approaches in terms of retrieval effectiveness and efficiency.

Design a scalable and exact solution that handles practical RDF data sets with tens of millions of triples. To address the scalability issues, our solution builds a new, succinct and efficient summary from the underlying RDF graph based on its types. Given a keyword search query, we use the outline [1] to prune the search space, leading to much better efficiency compared to a baseline solution. To sum up, our contributions we identify and address limitations in the existing, state-of-the-art methods for keyword search in RDF data. This shows that these limitations could lead to incomplete and incorrect answers in real RDF data sets. But we propose a new, correct baseline solution based on the backward search idea. Develop efficient algorithms to summarize the structure of RDF data, based on the types in RDF graphs, and use it to speed up the search is more scalable and lends significant pruning power without sacrificing the soundness of the result. The conclusion is light-weight and updatable. Also by applying optimization algorithm to the search result we will gain most appropriate and efficient result. Optimization here is check with genetic algorithm

Proposed algorithm

To find the keyword on large RDF first convert the RDF data set to RDF graph. Apply partitioning algorithm [1] and Summerization algorithm [1] to the RDF graph. By this result searching is done here searching is done with Genetic algorithm. By applying GA, result will be more perfect and also time complexity can be reduced.

Genetic algorithm is based on principles of development and natural selection. It is a search technique used in computing to find true or fairly accurate solutions to optimization and search problems. Genetic Algorithms are guided by analogy of natural behavior. They make use with a population of "individuals", each representing a possible solution. The majority conferred individuals are given opportunities to "reproduce", by "cross breeding" with other individuals in the population. The new generation thus produced contains mellower proportion of the description owned by the good members of former generations. The well projected GA will converge to optimal solution of the problem.

Parameters in Genetic Algorithm

There are many parameters to consider for the application of GA. Given below are certain parameters and methodology that are discussing

Representation of a Chromosome

A chromosome represent a path which includes the given search keyword.

Evaluation of fitness function

The fitness functions is the shortest path problem to find the weight of the path and thus find the shortest path.

Initialization

Initialization is the phase of genetic algorithm in which individual genomes are chosen from a large scale group for later breeding.

Crossover

The stage at which the chromosome is broken depends on the randomly selected crossover point. It is based on exchange between two fixed length chromosomes.

Mutation

When selection gets over and crossover, a new population full of individuals will emerge. This operator randomly alters genes to partially shift the search to new locations. Mutation also randomly selected operators are taken.

Selection

A practical variant of the genetic process of constructing a new population is to allow the best organism from the present generation to carry over the next, unaltered. This will make sure the quality of the genetic algorithm will not decrease from one generation to next. Individual solutions are selected through a *fitness-based* process. Most functions are stochastic and designed so that a small proportion of less weight solutions are selected.

Reproduction

The very next step is to generate a second generation population of solutions from those selected by way of genetic operators - crossover or mutation.

II. PSEUDO CODE

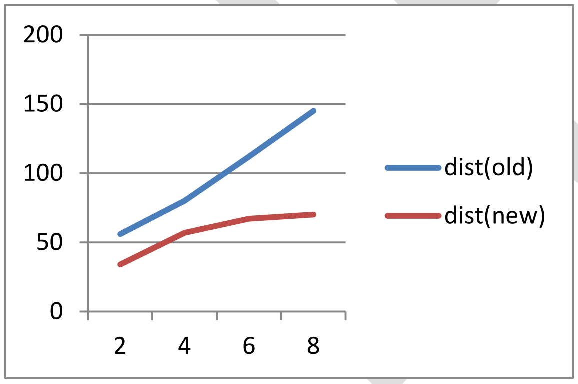
ALGORITHM

1. Initialization - Create an initial population. This population is usually randomly created and can be any desired size, from only a few individuals to thousands. Here the initial population is the number of nodes that matching the keyword and foaming a path with the keyword. All should be fully connected from that minimum path should considered as the best solution. Here fully connected best m solution is taken as the initial population.
2. Evaluation – Each and every member of the population is then evaluated and we calculate a 'fitness' for that individual. The fitness value is computed by how well it fits with our desired requirements. Here best is chosen with the best path that have the lesser weight , weight is calculated by best=7*(m-(number of selected nodes))/m, were m be the total number of solution
3. Selection - We want to be constantly improving our populations overall fitness. Selection helps us to do this by discarding the bad designs and only keeping the best individuals in the population. There are a few different selection process but the basic idea is the same, make it more likely that fitter individuals will be selected for our next generation. in the selection process 2m solution are generated ,first m solution is copied from the initialization process, "for i=1 to m choose randomly selected two solution and find best in the two "these should be repeated for m times and thus 2m solution should be generated.
4. Crossover - During crossover we generate new individuals by combining aspects of our selected individuals. The hope is that by combining certain traits from two or more individuals we will create an even 'fitter' offspring which will inherit the best traits .In the crossover section first best m solution is chosen from the selection process, next m solution is generated by " for i=1 to m "choose randomly selected two solution apply a randomly selected crossover operation to the two solution and generate a new solution""
5. Mutation - We need to add a little bit randomness into our populations' genetics else every combination of solutions we can create would be in our initial population. Mutation typically works by making very small changes at random to an individual. In the mutation section first best m solution is chosen from the crossover process, next m solution is generated by " for i=1 to m "choose randomly selected two solution apply a randomly selected mutation operation to the two solution and generate a new solution""
6. And repeat! - Now we have our next generation we can start again from step two until we reach a best solution.

III. Simulation Results

The section assess the performance of the proposed Searching in the RDF data using Genetic algorithm. Experiments are conducted on a desktop computer with Intel Core2 Duo Processor with Windows8 operating system along with 2GB RAM. The algorithms were introduced in java. MySQL is used as a backend. Any real life RDF datasets can be used for the experimental evaluations. The search result using genetic algorithm is compared with the old search algorithm result it shows that searching using genetic algorithm gives the most accurate result. Also compared with time complexity it also shows that the new one is taking lesser time with best results.

The figure shows the new one give the most perfect result compared to the old one, and also second figure shows time complexity regarding the new and old one. This shows the better performance.



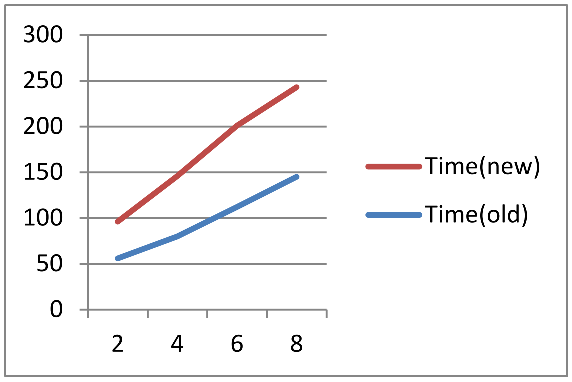


Fig : Performance Evaluation

IV. CONCLUSION

The most important research areas in data mining are searching in RDF data set which is useful in web search areas, which is important when considering to the semantic web concept.While using genetic algorithm for searching improves the searching quality and also better performance regarding to the old one. The main benefit and advantage of using genetic algorithm is it provides better accuracy and time complexity can be reduced compared to the old one. Keyword search now days we know the most important research area. The end results are analyzed based on the algorithm and dataset selected.

V. ACKNOWLEDGEMENT

The first author would like to thanks all those people, who guided and supported. Without their valuable guidance and support, this task was not possible and also likes to thank colleagues for their discussions and suggestions.

REFERENCES:.

[1] Wangchao Le, Feifei Li, AnastasiosKementsietsidis, and SongyunDuan, "Scalable Keyword Search on Large RDF Data" ieee transactions on knowledge and data engineering, VOL. 26, NO. 11, NOVEMBER 2014

[2] H. He, H. Wang, J. Yang, and P. S. Yu, "BLINKS: Ranked Keyword Searches on Graphs," in Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '07, June2007, pp. 305–316.

[3] Y. Chen, W. Wang, Z. Liu, and X. Lin, "Keyword Search on Structured and Semi-Structured Data," Proc. ACM Int'l Conf. Management of Data (SIGMOD), 2009.

[4] Y. Luo, W. Wang, and X. Lin, "SPARK: A Keyword Search Engine on Relational Databases," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE), 2008.

[5] V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IRStyle Keyword Search Over Relational Databases," Proc. 29th Int'l Conf. Very Large Data Bases (VLDB), 2003.

[6] G. Li, B.C. Ooi, J. Feng, J. Wang, and L. Zhou, "Ease: Efficient and Adaptive keyword Search on Unstructured, Semi-structured and Structured Data," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2008.

[7] Y. Chen, W. Wang, and Z. Liu, "Keyword-Based Search and Exploration on Databases," Proc. 27th Int'l Conf. Data Eng. (ICDE),2011.

[8] A. Aggarwal and J.S. Vitter, "The Input/output Complexity of Sorting and Related Problems," Comm. ACM, vol. 31, no. 9,pp. 1116-1127, 1988.

[9] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S.Sudarshan, "Keyword Searching and Browsing in DatabasesUsing Banks," Proc. 18th Int'l Conf. Data Eng. (ICDE), 2002.

[10] J. Broekstra et al., "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," Proc. First Int'l Semantic Web Conf.The Semantic Web (ISWC), 2002.

[11] H. Fu and K. Anyanwu, "Effectively Interpreting Keyword Queries on RDF Databases with a Rear View," Proc. 10th Int'l Conf. Semantic Web (ISWC), 2011.

[12] Y. Guo, Z. Pan, and J. Heflin, "LUBM: A benchmark for OWL Knowledge Base Systems," J. Web Semantics, vol. 3, pp. 158-182,2005.

[13] C. Weiss, P. Karras, and A. Bernstein, "Hexastore: Sextuple Indexing for Semantic Web Data Management," Proc. VLDB Endowment, vol. 1, pp. 1008-1019, 2008.

[14] B.B. Dalvi, M. Kshirsagar, and S. Sudarshan, "Keyword Search on External Memory Data Graphs," Proc. VLDB Endowment, vol. 1,pp. 1189-1204, 2008