Performance Enhancement of Search System

Ms. Uma P Nalawade¹ Mr. Ganesh V Kadam²

Computer Engineering, JSPM’s NTC, Pune
Computer Engineering, JSPM’s NTC, Pune.

Abstract: In this computerized world every element have to compete with the other one and the maximum stress is given on the fastest innovations and for that speed in information retrieval is a focused factor. As it is seen in studies of data or information retrieval there are so many proven theories, on which already proven theorems are present and the other important factor is everyone says that their theorems have more usable. Although this many things are present still there is lack of benchmarking about a particular searching technique. In this paper the study and comparison of the new searching technique with the help of keyword searching technique and keyword re-ranking technique. This paper tells about how a keyword searching can be enhanced with the merging of keyword re-ranking technique in it. The algorithm used in it will try to find out specific information by taking keywords given in the user entered queries. This technique can be more useful to fetch the information with the help of keyword re-ranking technique. According to observations, while accessing information various kinds of algorithms are applied on different types of databases, again the data which is stored is in different formats, as well as of different sizes. The very well-known type of data is structured and the semi-structured data which uses keyword search techniques which include query processing, ranking functions, result generation, search quality results. Searching of information from the database is an indispensable component for which algorithms like greedy algorithm, again networks generator algorithm are used for giving better performance. Day-by-day the size of database keeps increasing that time the networks generator algorithm is very much useful to form a network and access the node. By providing users to access databases with the help of simple keywords can relieve the users from complex structured query language. Techniques which enable users to seamlessly access vast collection of heterogeneous data sources are in great demand.

Keywords: keyword search; structured data; semi-structured data, ranking; query processing; information retrieval.

I. INTRODUCTION

The ubiquitous search text box has transformed the way people interact with information. Nearly half of all Internet users use a search engine daily, performing in excess of 4 billion searches. The success of keyword search stems from what it does not require namely, a specialized query language or knowledge of the underlying structure of the data. Internet users increasingly demand keyword search interfaces for accessing information, and it is natural to extend this paradigm to relational data. Keyword search on semi-structured data (e.g., XML) and relational data differs considerably from traditional IR.1 A discrepancy exists between the data’s physical storage and a logical view of the information. Relational databases are normalized to eliminate redundancy, and foreign keys identify related information. Search queries frequently cross these relationships (i.e., a subset of search terms is present in one tuple and the remaining terms are found in related tuples), which forces relational keyword search systems to recover a logical view of the information. The implicit assumption of keyword searches that is, the search terms are related complicates the search process because typically there are many possible relationships between two search terms. To improve search quality and users search experience, various techniques have been provided. Unlike traditional database applications where query results are fully specified by structured queries, the first task in keyword search is to define query result which automatically gathers relevant information that is generally fragmented and scattered. This paper is all about the information retrieval based on keyword search techniques, for that the candidate networks generator algorithm is been used and the greedy algorithm is also used. With the help of these techniques the new system (i.e. a web application) will be generated which will be containing the various search techniques like query based searching, graph based searching, keyword based searching and the keyword re-ranking. The keyword re-ranking will be the count of the keyword that is for how many times that particular keyword is entered, according to that the count will be stored.

To be effective, the engine must have three fundamental properties. It should not miss relevant answers, has to be efficient and must generate the answers in an order that is highly correlated with the desired ranking. It is shown that none of the existing systems has implemented an engine that has all of these properties. In this paper the three algorithms are applied on the DISCOVER database, first is the candidate networks generator algorithm, this algorithm is used for evaluation of the candidate networks, this candidate networks can share the join expressions. Ultimately it results in giving the opportunity to build a set of intermediate results and it can be further used for computation of multiple candidate networks. The plan generator produces an execution plan that calculates intermediate results. The important thing is the candidate networks may
have a number of joins that is only bound by the dataset. In this case by formalizing keyword search on relational databases and produce intuitive semantics. This algorithm ultimately results into a complete and non-redundant set of candidate network. Here the word “complete” means it produces the minimal joining networks of tuples and the word “non-redundant” means if any candidate network of set is excluded then there are minimal joining networks of tuples that are not discovered. The analysis has specified that when a maximum size of the candidate networks is bound by the size of database schema [1, 2, 3]. The next part is the study of a cost model to minimize the total cost of the evaluation of all candidate networks. Then we move towards the greedy algorithm that discovers near-optimal plans, with respect to the actual cost model, for set of candidate networks by choosing in each step of join between two tuple sets.

II. OVERVIEW

Users perform searches to satisfy data wants. A keyword question is associate degree expression of such associate degree data want, and it's the task of the retrieval system to come back data things that area unit relevant thereto want. For unstructured text, the data things area unit distinct documents. For relative knowledge, however, the data things area unit (possibly joined) tuples. The relative search system thus has the extra responsibility of deciding the candidate tuple joins. In addition, the keyword question contains no schema data, in order that every keyword probably should be matched against every field of the joined tuple. In a very structured source language like SQL, there's just one correct answer set. In distinction, there square measure several plausible similarity metrics, every with its own manner of inferring a user’s data want from a question , and of hard the query’s similarity to data things, to get a ranking of answers. The effectiveness of a response to a keyword question, and therefore of the similarity metric, isn't one thing that may be formally proved; rather, it's determined by the user World Health Organization complete the data want, developed the question, and perused the response. This effectiveness should be by trial and error assessed.

2.1. Keyword search for Structured and Semi-structured data:

Web search engines are widely used for searching textual documents, images, and videos. There are also vast collections of structured and semi-structured data both on the Web and in enterprises, such as relational databases, XML, data extracted from text documents, workflows, etc. Traditionally to fetch such kind of data recourses, all the users have to go for learning structured query languages like XQuery, SQL etc. the traditional process of accessing information was very complex and lengthy, also speed was lesser so the keyword search has become empowerment for the users to efficiently access structured data using keyword queries.

2.2. Keyword search for Graph databases:

The different databases such as XML, HTML and relational as a graphs with entities as nodes and relationships as edges. The textual information is available with nodes and possibly edges. To efficiently extract from the data graph, the best answer is trees. In text-driven queries the backward expanding searches, starting at nodes matching keywords and working up in the direction of roots. This is a very commonly used method. Instead of this we can also apply Bidirectional search, which improves backward expanding search by allowing forward search from potential roots towards leaves. Keyword search over graph-structured textual data has become more popular in database community. Systems for “schema-agnostic” keyword search on databases such as DBXplorer [2], BANKS [3], DISCOVER [1], model a response as a tree connecting nodes (tuples) which contains different keywords. The experimental results which are found after implementation of Bidirectional search algorithm which is a part of BANKS system, the performance is compared with the backward expanding search.

III. PROBLEM DEFINITION

All the searching techniques which are been used traditionally in the user required to learn the structured query languages like SQL, XQuery etc. According to the analysis there is huge amount of data present and that should be handled in very efficient manner. There are various methods to search the exact required methods like some methods are graph based, some are query based, classification according to the types of data for example images, audio, video, textual information, digital signaling etc. the database keep growing so to efficiently search we can apply keyword search and keyword re-ranking with the current used algorithm can improve the overall performance of the web application.
IV. METHODOLOGY

4.1. Candidate network generation algorithm:

The candidate network generator inputs the set of keywords \( k_1, \ldots, k_m \), the non-empty sets \( R_a \) and the maximum candidate networks size \( T \) and outputs a complete and non-redundant set of candidate networks [1, 2]. The algorithm has following major steps:

1. We create a tuple set graph \( G_{TS} \).
2. A node is created for each non-empty tuple set \( R_{ik} \), it includes free tuple set.
3. An edge \( R_i \rightarrow R_j \) is added if the schema graph \( G \) has an edge \( R_i \rightarrow R_j \).
4. The algorithm is based on breadth-first traversal of \( G_{TS} \). We keep a queue of active joining networks of tuple sets.
5. We start traversal from all tuple sets that contain the randomly selected keywords.

In addition, the implementation never places in \( Q \) a joining network of tuples sets \( J \) that has more than \( m \) leaves, where \( m \) is the number of keywords in the query. For example, if the keywords are two then only joining sequences are placed in \( Q \).

Indeed, even if this rule were excluded the output of the algorithm would be the same, since such a network \( J \) can neither meet the acceptance conditions listed next nor be expanded into a network \( J \) that meets the acceptance conditions. Nevertheless, the rule leads to cleaner traces and better running time. The algorithm outputs a joining network of tuple sets \( J \) if it satisfies the following acceptance conditions:

1. The tuple sets of \( J \) contain all keywords, i.e. \( \text{Keywords} (J) = \{k_1, \ldots, k_m\} \).
2. \( J \) does not contain any free tuple sets as leaves.

An important property of the algorithm is that it outputs the candidate networks with increasing size. That is, the smaller candidate networks, which are the better solutions to the keyword search problem, are output first.

4.2. Cost Model:

The theoretical study of the complexity of selecting the optimal execution plan is based on a simple cost model of execution plans:

1. We assign a cost of 1 to each join.
2. We use this theoretical cost model in proving that the selection of the optimal execution plan is NP-complete.
3. It is easy to see that the problem is also NP-hard for the actual cost model of DISCOVER.

The actual cost model of DISCOVER exploits the fact that we can get the sizes of the non-free tuple sets from the master index. We also assume that we know the selectivity of the primary to foreign key joins, which can be calculated from the sizes of the relations. The actual cost model defines the cost of a join to be the size of its result in number of tuples. (The cost model can easily be modified to work for the size in bytes instead of the number of tuples.) The cost of the execution plan is the sum of the costs of its joins. Notice that since there are indices on the primary and foreign keys, the cost of a join is proportional to the size of its result, since the joins will typically be index-based joins [1, 2, 3].

4.3. Greedy Algorithm:

Theorem given below depicts that a greedy algorithm that produces a near optimal execution plan, with respect to the actual cost model defined above, for a set of candidate networks by choosing in each step the join \( m \) between two tuple sets or intermediate results that maximizes the quantity. Where \( \text{frequency} \) is the number of occurrences of \( m \) in the candidate networks, \( \text{size} \) is the estimated number of tuples of \( m \) and \( a, b \) are constants. The \( \text{frequency} \) a term of the quantity maximizes the reusability of the intermediate results, while the \( \log_b (\text{size}) \) term minimizes the size of the intermediate results that are computed first. We have experimented with multiple combinations of values for \( a \) and \( b \) and found that the optimal solution is closer approximated. The greedy algorithm may output a non-optimal list of intermediate results. However, in special cases the greedy is guaranteed to produce the optimal plan. One such case is described by the theorem.

4.3.1. Algorithm Selects the list if common results

Input: set \( S \) of all candidate networks of size \( \leq T \)
Output: list of \( L \) i.e. the common networks to build
{  
While not all candidate networks in S have been added to L  
Do  
{  
Let Z be the set of all small join sub expression of l join contained in at list one candidate network in S;  
Add the intermediate result m with the  
Maximum[Frequency/log²(size)] value in Z to L;  
Rewrite all candidate networks in S to use m where possible;  
}  
}  

4.4 Experiments
The paper evaluates the algorithms of DISCOVER with detailed performance evaluation on a TPC-H database. First it measures the pruning efficiency of the candidate network generator. In particular, we measured how many joining networks of tuple sets are ruled out based on the pruning conditions of the candidate network generator. Then we compare the plans produced by the greedy to the ones produced by the optimal, where the optimal execution plans are computed using an exhaustive algorithm [1, 3]. We also compare the speedup in runtime performance for generating and executing the execution plan using the greedy and the optimal algorithm compared to the naive method, where no intermediate results are built. Finally, we compare the overall execution times of DISCOVER for some typical keyword queries to the naive method and to the optimal method.

The system is nothing but the combination of these different searching techniques and addition to that a keyword searches is applied with it. Due to the application of the keyword search technique the overall performance is been increased. As we have shown in above sections about the different algorithms which are doing different tasks for the as the paper tells about the candidate networks generator algorithm which is used to obtain the non-empty tuple sets and second thing is to avoid production of any redundant candidate networks. We model the database as a weighted directed graph in that all the nodes will be the representation of the tuples or rows in that particular database, or an XML element [2]. In the graph model which is generally used in several systems such as DBXplorer, BANKS, DISCOVER and Object Rank for each row r in a database that we required to represent, the graph that has a corresponding node[1].

V. RESULT

5.1. Generated Graphs: 

![Performance Graph](image)

*Figure 1. Example of Performance Graph after Search*
VI. CONCLUSION

The paper has taken reference of DISCOVER database, a system that performs keyword search in relational databases. It proceeds in three steps. First it generates the smallest set of candidate networks, and then the greedy algorithm creates a near-optimal execution plan to evaluate the set of candidate networks [1]. Finally, the execution plan is executed by the DBMS. In this work, we defined two keywords to be associated if they are contained in two tuples connected through primary to foreign key relationships. This is an intuitive and challenging association criterion as we have shown. In the future, we plan to extend DISCOVER to handle more association criteria such as: First, the keywords [1,4].

REFERENCES