Keyword Queries by Matching Synonyms in Relational Databases

Dingfang Huang[a], Dong Xie[b], Heyun Liu[c*]

Information School, Hunan University of Humanities, Science and Technology, Loudi, China

[a]rcyzdingding@qq.com, [b]dong.xie@hotmail.com, [c]changsha7298@126.com

[*]corresponding author

Keywords: Relational database; Keyword query; Query expansion

Abstract: In recent years, the database technology has been widely used in a few fields. The keyword search technology based on relational databases does not require users to master any knowledge of SQL grammar and database schema. Users only need to input keywords, and conveniently search information via keyword like internet search engines. As results, users are pleased to use the technology. In this paper, we employee a query expansion plan based on the query sentence, and propose a query expansion method based on thesaurus. The method matches key words with the thesaurus. If the key word matches a word in the thesaurus, and the words in the same row are used as a synonym for the key word. Test results show that the recall and precision ratio of the system are satisfied with needs of applications.

Introduction

With the development of information technology, increasing data in the information world are mixed with all sorts of data in various areas of industry. It also requires continuous development and improvement of search technologies. Users search contents not only in the keyword itself, and more contact with the key words for a variety of information.

Most relational database systems based on keyword searches only use simple matches without semantic expansions of keywords. For example, a user enters a keyword, system need to determine what field to search for this keyword belongs to. Retrieval systems are lack of semantic matching of keywords, and it is easy to lead to problems such as similar or same terminologies in different fields. It is difficult to accurately search the users' desired results. Especially, in the agriculture field. For example, when the user enters 'Apple', there will be Apple's cell phone and other electronic products. Fruit apples can also occur, it is difficult to select for users. Users enters 'potato', results often only contain 'potato' information. Results of potato information was deleted in the database. Therefore, the semantic and syntactic analysis of keywords are the focus of search technologies, which are extended to search by semantic matching, This will not only improves accuracies of queries, and also makes results returned more to meet the needs of users.

Matching tactics

Synonymicon

Dictionary of the design process, synonymous with all in one row, separated by a space between each word, and this set of synonyms most commonly the first word in each line. Dictionary of this design approach will increase the redundancy, but the structure of the dictionary is full, not only can very quickly find synonyms, dictionary and the space is usually very small. When we use a thesaurus to find synonyms of keywords, simply to match keywords with the word in the dictionary if the keyword matching on a word in the dictionary, this word at all of the words are synonyms of the keywords. Keywords analysis and extended in the process can use a synonym dictionary.

Table 1 Synonym dictionaries

agriculture noun	another name
tudou	malingshu yangshanyu…
hongshu	digua fanqie…
…	…

Synonym matching strategies based on thesaurus

When a user in a given keyword, a keyword exists more than one synonym, given the existence of these synonyms, leading to incomplete search results for that keyword, omits many keyword-related results.

Synonym matching ideas are entered by the user keyword and synonym dictionary to match if the match is successful, the synonyms found in the dictionary, as a synonym for the keyword, so as to achieve the keyword expansion.

Keyword search system

Definition

Definitions 3.1: Data charts [1]: data graph *G* is a directed graph consists of two sets of, respectively of the two sets is the set of nodes *V(G)* and the set of nodes that form the edge *E(G)*. Which direction is determined by the primary foreign keys that point to the direction.

Data side of a graph *g* is a weight, the weight function assigned to each edge of *G*. *e* is W_G a positive weight $W_G(e)$. Data graph *G* weight:

$$W(G) = \sum_{e \in e(g)} W_G(E) \quad (1)$$

In a directed graph *G*, *u*, and *v* are two nodes in the graph, which assigns a weight of two nodes, as shown in the following formula: $W_e((u,v)) = 1$.

$$We((u,v))=\log 2(1+Nin(v)) \quad (2)$$

Among them, *We*((*u*,*v*)) indicated to the edge before, namely primary foreign key relationships between the nodes *u* and *v*, *We*((*v*,*u*)) back to the edge, node master a foreign key relationship between *v* and *u*, *Nin*(*v*) represents a reference number of nodes *v*. This right of every edge in the graph the data value to 1.

Steiner tree

Relational database query system based on data graphs. Steiner tree can be used to represent the results of keyword search.

Steiner tree [2]: In a graph *G*(*V*, *E*), there is a $v \subseteq v'$. If *t* is a connected subtree of a graph *G* and *t* contains all nodes in *V* ′, then called *t*-*v* in a graph *G* ' a Steiner tree.

Minimum Steiner tree [3]. Suppose *t* is a Steiner tree on *v*' in a graph *G*, $c(T) = \sum_{e \in E(T)} W_G(e)$ is *T* weights, *E*(*T*) represents a collection of edges in the tree *T*, $W_G(e)$ for the weights of the edges *E*. If at all in *v*′of Steiner trees of a graph *G*, *c*(*T*) minimum, *t* is said to be a minimal Steiner tree.

Top-k Steiner tree [4]. Assuming a key phrase query {*K1*, *K2*, *K3*,..., *Kn*}, which is the minimum Steiner tree to {*T1*, *T2*, *T3*, ..., *Tn*}, these minimum Steiner tree according to the size of the sum of edge weights to be ordered, mean $S(T_1) \le S(T_2) \le ... \le S(T_n)$.

Because Top-k Steiner tree is sorted according to the side of the right size, similar to the keyword search results are sorted.

Top-k keyword search: suppose we want to query keyword combination for *K*={*K1*,*K2*, *K3*,..., *Kn*}, we can graph *G*(*v*, *e*) found in Top-k Steiner tree for {*T1*,*T2*,*T3*,...,*Tn*}, scores for each tree, tree scores denoted $c(T_i)$,and $c(T_i) = \sum_{e \in E(T_i)} w_G(e)(i = 1, 2, ..., k)$, $c(T_1) \le c(T_2) \le ... \le c(T_k)$.

Keyword search method based on Steiner tree idea is to find in the data graph containing Steiner tree node associated with the keywords, and keywords in the query results are sorted according to a certain rule is returned to the user.

Classify Steiner tree

In the data graph, the number of connections can determine order of keyword search results. If the result tree is the number of keywords in the same phenomenon is less results in the number of connections the connection number ahead of the result tree. Because of a smaller number of tuples in the resulting tree more compact, illustrate the relevance of query results is good. However, this method is used when the query results are sorted, and often many connections the same number will appear in the result tree, so error will occur during the sorting process.

Assume a keyword collection for K={*K1, K2, K3, ... Kn*}, number of nodes in the data graph based on the keywords are 2n-1 types. Assuming entered 3 keywords in the keyword group are the *K1*,*K2* and *K3*, then there will be three scenarios: the first case only have 1 keyword in the Meta Group; in the second group there are 2 keywords; third level 3 keywords in a tuple. Steiner tree according to the three categories. Classification rules are as follows:

（1）{*K1*, *K2*, *K3*}
（2）{*K1*, *K2*}
（3）{*K2*, *K3*}
（4）{*K1*, *K3*}
（5）{*K1*}
（6）{*K2*}
（7）{*K3*}

Third contains three keywords into the first Steiner tree, which consists of only one node in the Steiner tree, not connected;

Second contains two key words into type Steiner tree, it needs to attach a two-element group set up to find all keywords, which means that we need to connect two nodes;

First contains a keyword into a third type of Steiner trees, it will connect three a set to find all keywords, and the corresponding need to connect three nodes.

Derived, category I need I to the Steiner tree is connected to the set of tuples can contain all of the keywords, Steiner tree I class (I) said.

Sorting results

Connection may decide on the number of keyword search results sorting order. 1st class contains three keywords in the Steiner tree, does not need to be connected with the other set of tuples, so the result is the best, followed by the 2nd, then 3rd class. We can come to a conclusion, Steiner(I) structure to Steiner(i+1) as well. So for the collation of the result according to the following principles:

(1) Steiner(I) ranks higher than the results of Steiner(i+1).

(2) In the same class in the Steiner tree, if the keywords appear more frequently, then the ordering of the results appearing first.

Algorithms

Searches for the purpose of the algorithm is to find qualified Steiner tree and return it. Using Top-k Steiner tree query results, find k minimum Steiner tree in front.

First is find a 1st class Steiner tree, because Steiner tree is the best query result, and we don't need to compute the tuple connection. 1st class so priority queries all Steiner tree and outputs, if find a k a 1st class Steiner tree, end of the query, returns the result in k; if the 1st class Steiner tree did not meet the number of k-, Steiner tree queries 2nd class. Steiner tree for 2nd class, it needs to connect 2-tuple, we discuss below the 2nd class of the Steiner tree calculation.

Assuming that input contain 3 keywords in the keyword group *k*={*K1,K2,K3*}, when Steiner tree in the calculation of the 2nd class, need to find the two-element group results. 4.5.3 the situation referred to in (2) and (3), (2) and (4), (3) and (4), (2) and (7), (3) and (5) and (4) and (6) the connection results. Assumptions taken (2) Steiner as a 2nd class a node in the tree, then starting from this node using the bidirectional search method for searching, and if it finds a node makes it and (2) nodes after doing and contains all of the keywords, then add this node to the Steiner tree, and (2) nodes a 2nd class Steiner tree. Cycle this process until we find a Steiner tree all 2nd class.

For category *I* Steiner tree, we need to be found to connect the *I*th tuple.

Search algorithm shows as follows:

Input: *k*={*K1, K2,..., Kn*} keyword group, *k*

Output: ex-k query results

```
1:    Find the set of tuples containing the keyword s；
2:    Result, Count=0；
3:    FOR i=1 to s DO
4:    Remove a tuple    //Calculate 1st Steiner tree
5:        IF    Number of keywords in the Meta Group==n
6:            Count= Count+1;
7:            RETURN ;
8:            IF K== Count
9:            BREAK;
10:       ELSE      //Calculation of class I Steiner tree
11:           Add the node s Steiner tree
12:           Starting with the s node bidirectional search containing Keywords node v
13:       IF s&&v If we find a stage and s nodes and operations contains all keywords
14:           Add a node v to Steiner tree, and s Constitution class I Steiner tree
15:           RETURN ;
16:           Count= Count+1;
17:           IF K== Count
18:           BREAK;
19:   END
```

The function of the algorithm is a search result and returns a result. Users enter keywords to find all nodes that contain keywords set. First search the 1st Steiner tree, such optimal results, if the search to such a Steiner tree output it, and the results were counted. Next to the search category *I* Steiner trees, Steiner tree output in line with conditions. Final judgments on whether the number of query results is *k*, if reached, *k* results returned.

Converted SQL statement

When the search algorithm to find all Steiner tree, the next step is to turn the Steiner tree into an SQL statement to find related records in a database.

Suppose we want to query "Gong Caichun published information on garlic", which contains 2 keywords, respectively, "Gong Caichun" and "garlic", need to find 2 Steiner trees during a search.

(1)1st class Steiner tree, only one node in the tree, no connection, the 1st Steiner tree into SQL statements for:

```
select *
from Information
where InformationId=I1
```

(2)For the second type of Steiner trees, we need to connect 2-tuples, Steiner trees into SQL statements for this class:

```
select *
from User, Information, Release
where    User. UserId= Release. UserId
      and Information. InformationId= Release. InformationId
```


and *User. UserId =U1*
and *Information. InformationId =I1*

The time complexity of algorithms

Set in the data graph containing the number of keywords to n'. First working group are the nodes, because the need to traverse all nodes during algorithm execution, set the time complexity of the algorithm is o(n). For each type of Steiner trees. Assume that enter keywords included in the Group c, the algorithm needs to cycle c, on each iteration process, need to find I collection and connect them. Suppose we want to connect *C1*, and *C1*<<min(*2c*,*k*),*K* number of keyword search results. A collection of average number of connections is required is *C*/2. Average portfolio size in the collection is $n/2^c$-2.The total cost of computing Steiner tree is $O(c \times c_1 \times \frac{n^{'}}{2^c-1}^{\frac{c}{2}})$.Therefore, the time complexity of algorithms is $\max(O(n^{'}), O(C_0 \times \frac{n^{'}}{2^c-1}^{\frac{c}{2}}))$. Because the values of *c* and *k* are usually smaller, and the *n*' value is decided by the user-entered keywords *c*, often will not be too great.

Experiment results

In this paper, SEEKE[5] of precision Precision and recall to measure Recall of keyword search results. The formula looks like this: $Precision = \frac{|R_Q|}{|A|}$ and $Recall = \frac{|R_Q|}{|R|}$. Among them, the RQ returns the query result set and the number of keywords related to the result tree, a number representing the number of query results in the results set, r represents the database and the number of keywords related to the result tree. Precision and recall of the keyword query as shown in following figures:

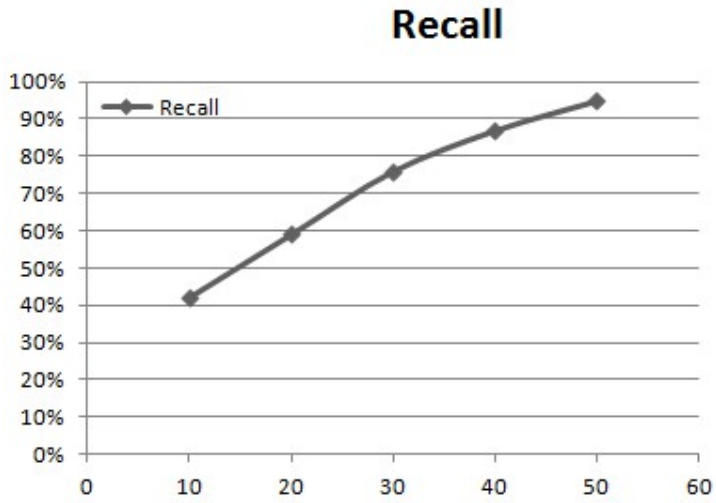

Fig. 1 Average Precision

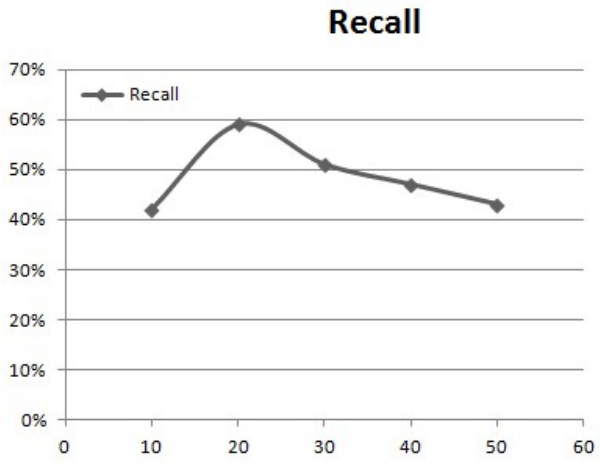

Fig. 2 Average Recall

Experimental results show that the synonym matching strategies to expand the keywords used for this article is valid.

Conclusion

In this paper, we are based on the query expansion strategies, and raise a query expansion method based on dictionary of synonyms, semantic expansion, so as to improve the precision and recall of the keyword.

Based on a specific keyword, the search technology will be in the area of future research, keyword search technology and many features in the field of combined research and development applied to different areas of search systems, applied to various fields, so as to solve practical problems. With rapid development of semantic technologies and ontologies, ontology-based database keyword search technology has become the focus of the relationship. Using Ontology to describe relational database semantics, semantic search technology to develop a relational database, we can further improve on the results of keyword search recall and precision.

References

[1] D. Wang. Query expansion based on keyword research and implementation of text retrieval technology [J]. 2014:9-15 Anhui University of technology.

[2] G. L. Li, Zhou XF, Feng JH, Wang JY. Progressive keyword search in relational databases. Los Almaitos, CA: IEEE Computer Society Press, 2009. 1183 -1186.

[3] V. V. Vazirani. The Steiner Tree Problem and Its Generalizations. In proceedings of APPROX,Aalborg,Denmark,1998:33-38.

[4] M. Tang. Research on algorithm of keyword search in relational databases [J], Heilongjiang University, 2013,37-48.

[5] J. Wen, S. Wang. SEEKER: relational database information retrieval based on keywords [J]. Software journal, 2005, 16 (7): 1270-1281.

