Semantic index for keyword search over tagged data

Ying Lou¹,a, Feng Zhong¹,b, Jinxiang Zhang¹,c, Yubo Peng²,*

¹School of Science and Technology, Zhejiang International Studies University, Hangzhou 310012, China
²Zhejiang Province E-commerce Promotion Centre, Hangzhou 311100, China

*Corresponding author e-mail: pengyubo2007@126.com, a louying2013@126.com, b zhong@zisu.edu.cn, czjx@zisu.edu.cn

Abstract. The index is crucial for information retrieval efficiency. Different with text data, tagged data contained rich semantics, which is useful to promote the quality of search results. It is observed that most existing indexes for keyword search do not consider semantics of tags. After an analysis of tagged data, we proposed the concept of result entity basing on the theory of relational database. We present a formula to quantify semantics of tags and then introduce a novel semantic index for keyword search. Experimental results demonstrated that our approach can help to reduce the size of the keyword inverted list in tagged document dramatically and improve the retrieval quality.

1. Introduction

Because of the friendliness, keyword search has become an important means to obtain information, and it is widely used in text documents and web documents. As more and more tagged data are generated in web and database, how to effectively obtain the information has both research value and practical significance [1]. Unlike text documents, the tagged data has contained a lot of semantics, which should be utilized to improve the search quality.

Traditional text data and web data do not contain any semantics. Structural and semi-structural data contain a lot of semantics, which presented in the tags. The friendliness of keyword search makes it more suitable for commercial applications, which can bring more value to common users for the tagged data. At present, most methods of keyword retrieval based on tagged data are based on structural information. The notion of SLCA (Smallest Lowest Common Ancestor) is adopted widely in XML keyword search [2]. Firstly, they located the LCA node that include all the keyword, then, the subtrees rooted LCA node should be returned as final results, the notion of sub-graph is adopted in keyword search on relational database [3]. Obviously, these researches focused on the structure information of tagged data, which payed Insufficient attention to semantics.

The index has great influence on retrieval efficiency and quality. The inverted index of keyword has obtained popular success in traditional information retrieval. The researches of keyword search on tagged data inherited the inverted index. Dewey code has adopted by XML keyword search, which can reflect the hierarchical information of tree. The following research continuously improve the dewey code to heighten the search results. In this paper, we focus on the semantics of tagged data. First, we propose the notion of results entity. Then we analyze the semantics of tagged data and introduce a
novel semantic index called TSI for keyword search. Final experimental results show that TSI improves keyword search quality beyond that of existing approaches.

The rest of this paper is organised as follows: Related work are presented in Section 2; Section 3 presents the concept of result entity; Section 4 describes our semantic index, we presented the experimental study is presented in Section 5; and we conclude this paper in Section 6.

2. Related works

Existing keyword researches on XML data use inverted index to improve retrieval efficiency [4], which use the Dewey code to write down the location information of keywords. The dewey code also can help to calculate the LCA results. Structure summary are used in the search of XML structure language. D(k)-index [5] is a dynamic structure index based on similarity relation, which can adjust the structure of index dynamically according to the load of current query statement. On the basis of a (k)-index, it absorbs the advantages of dynamic reflection of query distribution in apex and combines them. Its core idea is to dynamically adjust the local similarity of each node in the index structure.

There are also many researches on how to rank the search results. [6] proposed to rank query results according to the distance between different keyword matches in a document. XRank [4] extends Google’s page-rank hyperlink metric to XML elements to rank returned LCA results. XSearch [1] combines a simple TF (Term Frequency) / IDF (Inverse Document Frequency) IR ranking with tree size to rank results. XReal [7] exploits the statistics of under-lying XML data to infer users’ search intention and rank LCAs. Based on the concepts of XML TF and XML DF (document frequency), it proposes an XML TF / IDF similarity ranking scheme.

3. Result entity

In text retrieval and web retrieval, retrieval results are usually based on documents, which does not apply to tagged data. The single tagged document commonly contains a lot of information. Returning all the single document should make users search information again. The search results of tagged should be the fragment of the document. What kind of fragments should be returned is the issue that we need to study.

In keyword search, the keywords are actually the known information of the users, so it is meaningless to only return the information of the keyword itself. The information contained around the keyword needs to be returned. The LCA method only obtains the information fragments containing keywords from the structure, which will lead to the situation of too little or too much information. We can conclude that the search results should include enough information, simultaneously, it can't contain too much information to make a second search. We use the conception of entity in relational database to define the scope of search result. Entity refers to the objective objects or things that exist in the real world. It can be distinguished from each other. As far as database is concerned, entity often refers to the collection of some kind of things. So we define the Entity Fragment in tagged data.

| ID | Title       | author | year | price |
|----|-------------|--------|------|-------|
| No1| keyword search | Tom   | 2020 | 49    |

Fig 1 A tagged document fragment
3.1. Definition 1 (Entity Fragment).
Entity Fragment is the smallest fragment in tagged that there are at least two attributes describe the entity. The tags that directly describes the value is attribution.

According to the definition 1, the subtree in figure 1 rooted “book” is an entity fragment. The subtree root “id” is not an entity fragment. Although fragments with multiple attributes may still belong to one attribute, they can be eliminated through manual judgment.

3.2. Definition 2 (Identical Semantic).
Give two Entity fragment E₁ and E₂, if D(E₁)=D(E₂), E₁ and E₂ are identical semantic. The D(E₁) is the description of tag EF(entity fragment).

According to the definition 2, there are two identical semantic EFs “book”. Because tagged data does not require a strict schema definition like a relational database, entity fragments that have the IS (identical semantic) have not the exactly the same attribute, but still belong to the same entity description.

3.3. Definition 3 (Result Entity).
Processing a keyword query Q on tagged data D returns a set of Result Entities. Each result entity is the fragment that is composed at least one entity fragment and contains all keywords in Q.

The search result should be Result entity, which can the completeness of search results. As shown in fig.1, consider the query (Tom, keyword, search) on bookstore record. The user intends to find the book about “keyword search” that authored by “Tom”. The tagged data (a) could be indicated as (b) and (c). “Tom” is tagged by author, which indicated that “Tom” is author’s name. “49” is tagged by price, which noted that the price of the book is 49. Despite with the semantics of “author” and “price”, we would lose the tagged data information.

We propose the concept of result entity to define the search result. In our approach, all return results should be the result entities, which contain complete information. We should discuss the index based on the result entities.

4. Semantic index
In order to efficiently identify the keyword search answers, we propose the new index called SIT (semantic index of tag) in this section.

We consider the index from two aspects, one is the size of the index, the other is the semantics of the index. Existing indexes usually only consider the information of structure, just as Dewey coding in XML data considers tree structure and graph structure in relational database. According to the limitation of entity results, we can ignore the details of the structure. The size of the index will be reduced, which can improve the efficiency of retrieval. For example, the keyword “Tom” in fig 1, its index should be ignored the structure of “author” which is not an entity result.

However, the semantic of author should be lost because it also contains semantics. This paper does not study whether the semantics match the query through artificial intelligence. We use the concept of entity in relation theory to judge the importance of the attributes of keywords. In the entity, the primary key is necessary and unique. We count the importance of the attribute by its distribution in the data.

**Inference 1** If an attribute always appears in the entity fragment, this attribute is necessary for the entity fragment.

**Inference 2** If an attribute always appears once in the entity fragment, the attribute is important for the entity fragment.

According the inferences, for example, the **title** is necessary for the book and the author is also necessary. But **title** is more important for **author** just because that title is more unique than **author**. the attribute containing the keyword determines the importance of the keyword, so the keyword weight is
calculated by calculating the weight of the attribute. These values can be calculated offline as the semantics of the query results before the query. We calculate the semantic weight through the following formula:

$$SW(a) = \ln\left(\frac{C(a)}{C(EF_a)} + \frac{C(EF_a)}{C(EF)} + 1\right)$$  \hspace{1cm} (1)$$

$C(a)$ is the count of the attribute $a$ in the certain kind of entity fragment. $C(EF_a)$ is the count of the entity fragments that contain attribute $a$. $C(EF)$ is the sum of the certain kind of entity fragment. So the formula (1) can reflect the importance and necessary of the attribute $a$.

Before executing query, semantic index is constructed by scanning the target data. For each keyword, there is an inverted list corresponding to it. Each item in the inverted list contains the entity fragment and weight value of the keyword. Our method is not to discard all structural information. For example, in XML data, we will record structural information in entity fragments, but we do not record structural information in detail in entity fragments. Only the structure information is converted to weight information. Dewey cipher is used to represent the structural information of keywords in XML data. This paper inherits this idea. In other tagged data, we also can utilize the existing method to represent the structure information.

Compared with the existing index, SI has two advantages. First, it effectively reduces the length of the keyword inverted list, which has a great impact on query efficiency. Second, it can calculate the importance of keywords in advance. There is no need to decorrelate the query results after they are finished, which will also improve the relevance of the query. In this paper, we mainly use the importance to express the degree of correlation without further research.

5. Experiments

On the basis of the SI, we implement the keyword query algorithm named SIsearch. In the pre-process step, we parse the tagged data calculate its importance, record its entity fragment and construct inverted lists. Each distinct keyword $k$ has an inverted index with entries that correspond to the fragments and its importance.

We record the SI index size of three data (DBLP Xmark sigmodrecord) sets in Table 1. From the results, we can see that the SI index is relatively large with small dataset. The reason is that in the small data, the number of key words is not reduced and the record of importance information is increased. When the raw data is large, the SI can reduce the structural information of key words, so the increment is small.

| Dataset        | Rawdata | Semantic index |
|----------------|---------|----------------|
| DBLP           | 127M    | 156M           |
| Xmark          | 78M     | 82M            |
| SigmodRecord   | 3M      | 6M             |
We test the query accuracy on DBLP and XMark datasets. Comparing with the Xrank [4] algorithm, we can see that in most cases, the query accuracy of SIsearch is higher than that of XRank. XRank query is based on PageRank. It can be seen that in most cases, the query results of SIsearch are also more precise than XRank. Although XRank utilized the sorted method, it computed the search results by structural information, which will cause the irrelevant results should be returned. The entity results is more efficient.
We measure efficiency as CPU time consumed by running the algorithms. We compare the performance of SIsearch with that of SLCA [2] algorithm. The SIGMOD record is a small data set. The execution efficiency of SIsearch is not higher than that of SLCA, and it is lower in some queries. However, on the large data set of DBLP, the advantages of SIsearch are shown. Because in large data sets, entity fragments can contain more repeated keywords, which can be indexed once time. the query process will save time.

From the index size and query efficiency, we can see that SIsearch has obvious advantages when the dataset is large. In practical application, when the amount of data is large, the response time is longer, so SIsearch is more practical.

6. Conclusion
After analyzing the characteristics of tagged data, this paper proposed the semantic index. SI is based on the theory of relational database. It can record the semantic importance of keywords. We implement the keyword query algorithm on the basis of semantic index. Our experiments show that SIsearch outperforms existing approaches with respect to the efficiency and accuracy of keyword query.

The following research will focus on the relevance judgment based on semantic index. This paper mainly considers the use of database technology to complete the query. In the future, it will consider combining information retrieval technology to help query tagged data. In reality, there are a lot of data without explicit tags, but in fact, tags can still be used for semantic annotation. For some data of different industries, non-computer professionals still need to get professional information by keyword query, which will also be our research direction.

Acknowledgements
This research was financially supported by the Zhejiang Basic Public Welfare Research Project (No: LGF19G020002).

References
[1] S. Cohen, J. Mamou, Y. Kanza, Y. Sagiv, Xsearch: a semantic search engine for xml, in: Proceedings of 29th VLDB2003, 2003, pp. 45–56.
[2] Y. Xu, Y. Papakonstantinou, Efficient keyword search for smallest lcas in xml databases, in: Proceedings of the ACM SIGMOD, 2005, pp. 537–538.
[3] G. Li, B.C. Ooi, J. Feng, J. Wang, L. Zhou, Ease: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data, in: Proceedings of the ACM 2008, pp. 903–914.
[4] L. Guo, F. Shao, C. Botev, J. Shanmugasundaram, Xrank: ranked keyword search over xml documents, in: Proceedings of the 2003 ACM SIGMOD, 2003, pp. 16–27.
[5] Chen Qun ,Lim A ,Ong K W .D(k)-index :an adaptive structural summary for graph-structured data[ C] // Proceeding s of the 2003 ACM SIGM OD .New York :ACM ,2003 :134-144
[6] V. Hristidis, Y. Papakonstantinou, A. Balmin, Keyword proximity search on xml graphs, in: Proceedings of the 19th ICDE2003, pp. 367–378
[7] Z. Bao, T.W. Ling, B. Chen, J. Lu, Effective xml keyword search with relevance oriented ranking, in: Proceedings of the 25th ICDE2009, pp. 517–528.