Improved Ranking Method for Keyword Queries on Relational Database

Myint Myint Thein
University of Computer Studies, Mandalay
mmyintt@gmail.com

Abstract

Keyword search is an easy and potentially effective way to find information that is stored in relational database for ordinary users or web users. As results needed by user are assembled from joining tuples of multiple relations, ranking keyword queries are needed to retrieve relevant results by a given keyword query. For a given keyword query, we first generate a set of joining tuples, such as candidate networks (CNs). We then model the generated CN as a document. We evaluate the score for each document to estimate its relevance to a given keyword query. Finally, we rank the relevant queries by using each evaluated score as high as possible. In this paper, we propose a new ranking method by adapting existing IR scoring techniques based on the virtual document. We evaluate the proposed ranking method on DBLP dataset. The experimental results are shown by comparison of the proposed ranking method and the previous IR ranking method.

1. Introduction

A significant amount of data such as enterprise data has been stored in relational database. With more and more data being stored in relational database, it has become crucial for users to be able to search and browse the information stored in them. In traditional search model in relational databases, users need to have knowledge of the database schema and to use a structured query language (SQL). Even though relational database management systems (RDBMs) have provided full-text search capabilities, they do not support keyword search model. In contract, information retrieval (IR) techniques allow users to search information using keywords based on scoring and ranking, and do not need users to understand any database schemas. The text database and relational database are different that is challenging task to apply the keyword search techniques in IR to DB. The database research community has been introducing keyword search capabilities into relational database to support keyword queries. The existing methods of keyword search in relational databases can be broadly classified into two categories that are schema based method [1, 2, 10, 12] and graph based method [4 , 5, 6].

In schema based keyword search in relational database, it has a common method that is generating the candidate network (CN) in schema graph transformed from relations. Generating all valid candidate networks that are called connected tuple trees (CTT) by joining tuples from multiple relations. In relational database, data is stored in the form of columns, tables and primary key to foreign key relationships. For a given keyword query, the logical unit of answers needed by users is not limited to an individual column value or ever an individual tuple. It may be multiple tuples joined together. Therefore, the system generates the CNs with multiple tuples from different relations joined by foreign keys. There are many connected tuple trees that can be results for the query. These results are not surely useful to the user. We need to compute a single score for each CN in order to rank the relevant results. So, a ranking method is essential for getting user satisfaction.

There has been many studies dedicated to keyword search in relational database recently

[2, 11, 12]. For the ranking method, some systems considered each text column as a collection and each value in the text column as document by using IR weighting methods. The results are ranked according to a final score that is obtained by dividing the sum of all these scores by the number of tuples in the tuple trees. These methods can help improve the keyword search quality in relational database. Despite the existing studies, there are still several issues with existing ranking methods. Some of existing ranking methods may even lead to search results contradictory to user perception.

In this paper, we propose a new ranking method by adapting the IR ranking methods based on the virtual document. The proposed ranking method can evaluate the accurate scores for relevant results from relational database for the user. We conduct the experimental results on DBLP. The results show that the proposed ranking method support effective keyword-based search on large amounts of relational data.

The rest of the paper is organized as follows: Section 2 discusses the related works. Section 3 presents the preliminaries. Section 4 presents the proposed ranking method. Section 5 shows the system evaluation and Section 6 concludes this paper.

2. Related Work

The main goal of a keyword search system is to find a set of closely inter-connected tuples that collectively match the keywords. One type of methods is based on modeling data as a graph, and the results as subtrees or sub-graphs. Another type of methods is based on relational databases where structured data are stored.

Several researchers have been done on early keyword search systems for relational databases [10, 11]. Yu et al. [6] surveyed the developments on finding structural information among tuples in an RDB using an l-keyword query. They discussed the keyword search systems by comparing between schema-based keyword search and graph-based keyword search in RDB. The former evaluated the sets of answers by defining all minimal total joining networks of tuples between CNs and the latter showed how to answer keyword queries using graph algorithms focused on weighted directed graph.

IR-Style [11] proposed IR-style ranking method in straightforward manner to rank tuple trees. This method had not considered the effectiveness of the query results. Liu et al. [2] described the ranking formula by adapting four normalizations: tuple tree size normalization, document length normalization, document frequency normalization and inter-document weight normalization. This score function is not monotonic due to the four normalizations. SPARK2 [12] modified the IR ranking method based on the virtual document. Their method produced repeated information which concerns overlapping among the top-k join tuples trees. In this paper, we propose a new ranking method to reduce the meaningless results which are disappointed for user.

3. Preliminaries

In this section, we describe some basic concepts such as CN and Connected Tuple Tree (CTT), the generating results for a given keyword query and existing ranking methods.

3.1. Query Representation

A relational database can be viewed as a graph which represents a relational model such as schema graph G_s [1, 6, 7, 8, 13]. A relational database is a collection of relations. Each relation in the database corresponds to a vertex in G_s, denoted as the set of relation schemas {R1,R2,…}. Edges represent the foreign key to primary key relationships between pairs of relation schemas, R_i and R_j, denoted $R_i \rightarrow R_j$.

We use directed schema graph with the relations as its edges and the foreign key to primary key relationships of the relations as its edges that shows in Figure 2, as the schema graph of publication database. For simplicity, we assume all primary key and foreign key attributes are made of same attribute with attribute of related relation. There are no self loops and at most one foreign key to primary key relationship between any two relations.

Person

Pid	Name
P1	Jinlin Chen
P2	Peter P.Chen
P3	David W.Chen
P4	Yui-liang Chen

Publisher

Uid	Name
U1	Springer
U2	IEEE
U3	ELsevier
U4	ACM

Relation-Person -Inproceeding

Pid	IPid
P1	I1
P2	I3
P3	I4
P1	I2

Series

Sid	Title
S1	IFIP Series
S2	Lecture Notes in Computer
S3	Advances in Data Base
S4	Lecture Notes in Control

Inproceeding

IPid	Title	Pages	Rid
I1	An Adaptive Web Content Delivery System	284-288	R1
I2	Visual Based Content Understanding towards Web Adaptation	164-173	R3
I3	ER Model, XML and the Web	538	R2
I4	Abstract Machines, Control, and Sequents	123-136	R4

Proceeding

Rid	Title	Uid	Sid	Year	ISBN
R1	Adaptive Hypemedia	U1	S2	2000	3-540-67910-3
R2	Conceptual Modeling	U1	S2	1999	3-540-66686-9
R3	Adaptive Web-Based	U1	S2	2002	3-540-43737-1
R4	Applied Semantics	U1	S2	2000	3-540-44044-5

Figure 1. Publication Database Example

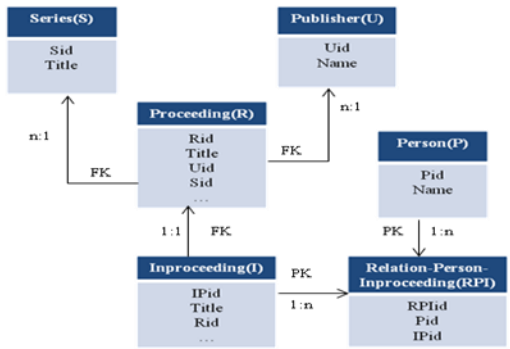

Figure 2. Publication Database Schema Graph

A keyword query (Q) consists of a list of keywords $\{k_1,k_{2,...},k_q\}$, and searches interconnected tuples that contain the given keywords. For a given query Q, a result is the set of all possible joining networks of tuples. A joining network of tuple is a connected tuple tree (T). Each node t_i is a tuple in the database, and each pair of adjacent tuples in T is connected via a foreign key to primary key relationship. Suppose (R_i,R_j) is an edge in the schema graph. Let $t_i \in R_i$, $t_j \in R_j$, and (t_i join t_j) $\in$ (R_i join R_j). Then (t_i,t_j) is an edge in the connected tuple tree T. The size of a connected tuple tree is the number of tuples involved. Note that a single tuple is the simplest tuple tree with size 1. Each connected tuple tree is the sets consisting of relational names that produced by a relational algebra expression, if each tuple in one relation contains a term of the keywords. For a given keyword query Q, the query tuple set R^N is a set of all tuples which belong to relation R that contain at least one keyword of the query Q. We denote R^F the free tuple set which is the set of all tuples in relation R and we use R^Q to denote a tuple set, which can be either a non-free tuple set or a free tuple set.

A candidate network is a tree of tuple sets R^Nor R^F with the restriction that every node must be a query tuple set. Every edge (R_i^Q,R_j^Q) in a CN corresponds to an edge (R_i,R_j) in the schema graph G_s. A CN can be easily transformed into its equivalent SQL statement that joins a sequence of relations with selections of tuples for

keywords over the relations involved. The size of a CN is the number of its tuple sets. For simplicity, we use I, R, U, P, S and RPI to denote the relations Inproceeding, Proceeding, Publisher, Person, Series and Relation-Person-Inproceeding respectively.

3.2. Candidate Network Generation

In this section, we describe generating the connected tuple trees as result. Given a keyword query Q, the system first receives all the query tuple set R^Q for all relations R as input. Then it focus on generating all the valid CNs which are joined expressions to be used to create connected trees of tuples that will be considered as potential results to the query.

For a given query, if CN is a result then each node belongs to the non-free query tuple set R^N and the free query tuple set R^F of each relation R. Note that the free query tuple set in CN cannot contain the query keyword, but they support to the non-free query tuple set as primary-foreign keys relationship. We generate CNs with the previous proposed CN generation algorithm [9] for this purpose. The generated CNs is only data bounded by the query and database. And it produces connected tuple trees as results by evaluating the corresponding joined expressions. In Figure 3, we present Connected Tuple Tree 1 and Connected Tuple Tree 2 that generate CN1 and CN2 are shown as example with data that contains in Figure 1.

Query 1:	"Peter XML Springer"
CTT 1:	P2→ I3→R2→U1
CN1:	$P^N \bowtie RPI^F \bowtie I^N \bowtie R^F \bowtie U^N$
Query 2:	"David Control Springer"
CTT 2:	P3→ I4→R4→U1
CN2:	$P^N \bowtie RPI^F \bowtie I^N \bowtie R^F \bowtie U^N$

Figure 3. Queries, Connected Tuples Trees and Candidate Networks

Finally, the every connected tuple tree as a result to a keyword query has its relevance score which indicates how relevant the connected tuple tree is to the query. Conceptually, all connected tuples trees of a keyword query will be sorted according to the descending order of their scores and only those with the highest scores will be returned.

3.3. Problems of Existing Ranking Methods

To rank documents, IR systems assign a score for each document as an estimation of the document relevance to the given query. In IR, a document is a basic information unit stored in a text database. It is also the basic unit of answers needed by users. A similarity value between a given query and a document is computed to rank documents.

In relational keyword search, the basic text information unit stored in a relational database is a text column value [3]. The basic unit of answers needed by users is a connected tuple tree which is assembled by joining multiple tuples, each of which may contain zero, one or multiple text column values. A similarity value between a given query and a connected tuple tree needs to be computed to rank connected tuple trees. Equation (1) shows the pivoted normalization scoring method by modifying existing IR ranking score, which is one of the widely used scoring methods in IR.

$$score(k, D) = \frac{ntf}{ndl} * \ln(idf) \quad (1)$$

$$ndl = \sum_{k \in D} (1 - s) + s * \frac{dl_{CN}}{avgdl(CN)} \quad (2)$$

$$ntf = \sum_{k \in D} 1 + \ln(tf_k(CN)) \quad (3)$$

$$tf_k(CN) = \sum_{k \in D} \frac{kf_i}{\max\{kf_1, \ldots, kf_i\}} \quad (4)$$

$$idf = \sum_{k \in D} \frac{N(CN) + 1}{df_k(CN)} \quad (5)$$

, where ntf indicates the normalized trem frequency, ndl is the normalized document length, idf is the inverse document frequency and tf_k (CN) denotes the number of occurrences of the CN in a document.

In general, retrieval effectiveness is vital to keyword search on relational database due to the fuzzy nature of keyword queries. Existing ranking systems [2, 3, 11, 15] have considered the size of an answer as a ranking factor to compute the relevance. The basic idea of the ranking method is: (i) assign to each tuple in the JTT a score by using a standard IR-ranking formula and (ii) combine the individual scores together by using an aggregation function, such as SUM, to obtain the final score [13]. In this method, the ranking results contain a large amount of one keyword query over results that contain all or most keyword queries but only once. This method is contradicted to user perception by ranking results. To solve this problem, we propose a new ranking method by adapting the IR ranking methods based on the virtual document and present a size completeness factor to retrieve the relevant ranking results by supporting modified IR ranking score.

4. Proposed Ranking Method

In this section, we propose a solution based on the idea of modeling a connected tuple tree as a virtual document. Consequently, the entire results produced by a CN will be modeled as a document collection. For example, a connected tuple tree: P1→IP1 for query "chen web" by modeling SQL queries that is shown in Figure 4.

```
SELECT *
FROM Person P, Inproceeding IP, Relation-
Person-Inproceeding RPI
WHERE P.Pid == RPI.Pid
AND RPI.IPid ==IP.IPid
AND P.Name LIKE '%chen%'
OR    P.Name LIKE '% Web %'
AND IP.Title LIKE '%chen%'
OR    IP.Title LIKE '%Web%'
```

Figure 4. SQL Statement Example

By adopting such a model, we assign an IR ranking score, such as $score_a$, to a connected tuple tree by using Equation (1) in section 3.3. Then, Equation (2) used to compute value of document length normalization and Equation (5) computes the inverse document frequency for each modeling connected tuple tree. Equation (3) evaluates the normalized trem frequency, whereas Equation (4) used to compute the number of occurrences of the CN which belongs to the connected tuple tree such as document.

After computing the modified IR scoring method, we then evaluate a score value for the size of CN and the size of the given query, especially for a keyword query whose relevant results are connected tuple tree involving multiple tuples, each of which contains a subset of the keywords query. We believe that the users usually prefer documents matching many keywords query to those matching only few keywords. To approximately the user perception, we define the size completeness factor for a query that is as follow:

$$score_b(k, D) = \frac{\ln(size(Q))}{\ln(size(CN))} \quad (6)$$

Finally, the final score of a connected tuple tree to a keyword query is the product of all the two scores:

$$score(T,Q) = score_a(k,D) * score_b(k,D) \quad (7)$$

We can get the significant score for the highest relevant keyword query after computing the final score(T,Q) for each connected tuple tree. We evaluate each CTT with the proposed ranking method by using two relations: person and inproceeding of DBLP dataset.

For example, we compute each score value with "chen web content" query step by step. For this query, some examples of the connected tuples trees include: P1→I2, P1→I1, P2→I3 and P3→I4. Note that P3→I4 is not a valid result tree to the query, as the leaf node I4 does not contribute to a match to the query. A possible results for this query may be: P1→I2, P1→I1, and P2→I3 whereas nodes P1 and P2 contain the

keyword “chen”, and nodes I1 and I2 contain two keywords “web” and “content”, I3 contains the keyword “web”. Then, we model a document for each CTT that is shown in Figure 5.

CTT	Document
P1→I2	Jinlin Chen→An Adaptive Web Content Delivery System
P1→I1	Jinlin Chen→Visual Based Content Understanding towards Web Adaptation
P2→I3	Peter P.Chen→ER Model, XML and the Web

Figure 5. Related Virtual Document for a CTT

For the modeling query of each CTT, we calculate each score value by using $score_a$ that is shown in Table 1.

Table 1. Evaluating Different Scores for Query “chen web content”

CTT	t∈CTT	tf_{chen}	tf_{web}	$tf_{content}$	$Score_a$
P1→I2	P1	1	0	0	2.66
	I2	0	1	1	
P1→I1	P1	1	0	0	2.67
	I1	0	1	1	
P2→I3	P2	1	0	0	2.17
	I3	0	1	0	

Table 2. shows the relevant results for query “chen web content” with connected tuple tree and its final score according to multiply each score value of $score_a$ and $score_b$. In order to this table, we can see that score value of P1→I2 is increased with the highest relevant score value.

Table 2. Relevant Keyword Queries for Query “chen web content”

CTT	Score
Jinlin Chen→An Adaptive Web Content Delivery System	1.41
Jinlin Chen→Visual Based Content Understanding towards Web Adaptation	1.33
Peter P.Chen→ER Model, XML and the Web	1.08

5. System Evaluation

We evaluate our proposed method on DBLP dataset. All queries generating algorithm was implemented in Java, and JDBC was used to connect to the database. For evaluation, we focus on 10 queries with query length ranging from 2 to 4.

To measure the effectiveness, we adopt two metrics used in previous studies [2, 13]: (i) number of top-1 results that are relevant (#Rel), and (ii) reciprocal rank (R-Rank), for a given query. The reciprocal rank is 1 divided by the rank at which the first correct answer is returned or 0 if no correct results are returned. In order to find the relevant results, we used the ranking strategies: IR-style and our ranking method for the same query. Then, we manually evaluated the results and selected the relevant result for each query.

5.1. DBLP Dataset

We use the Original Digital Bibliography and Library Project (DBLP) dataset [14] in our evaluation. It consists of a set of XML entries with each entry representing a single publication. We decomposed into relations from a downloaded XML file according to the schema that is shown in Figure 1. The size of the XML file is 173MB. Table 3. shows the statistics after the decomposition.

Table 3. Statistic of DBLP Dataset

Relation Schema	#Tuples
Person(Pid,Name)	174,709
Inproceeding(IPid,Title,Pages,Rid)	212,273
Proceeding(Rid,Title,Uid,Sid,…)	3,007
Publisher(Uid,Name)	86
Series(Sid,Title)	24
Relation-Person-Inproceeding (RPIid,Pid, IPid)	491,777

5.2. Evaluation Results

We compare the evaluation results of IR-style and the proposed method by using the same DBLP dataset. The manually evaluated relevant results are based on the AND semantics for keyword queries. Figure 6. shows the #Rel value of previous method and proposed method on the same queries. In this figure, there are no significantly different #Rel score values such as query (Q1). But the proposed method is higher than IR-style queries such as queries: Q2 and Q7 that contain subset of keyword query in the contents of each tuple.

Query	IR-Style	Proposed method
	#Rel	#Rel
Q1. chen web	12	15
Q2. chen web content	1	2
Q3. chen web springer	4	7
Q4. web content	2	13
Q5. content springer by chen	3	4
Q6.chen web springer 2000	3	5
Q7. david compiler generator	1	3
Q8. compiler springer by david	12	19
Q9. compiler generator	9	15
Q10. david compiler generator springer	1	3

Figure 6. #Rel Comparison of IR-Style and Proposed Method

In Figure 7, it shows the comparison of IR-style and the proposed method by using R_Rank metric in order to queries from Figure 6. Although Q1 has no significant on #Rel, there has been difference on R-Rank. Then, we can see that the proposed method gets the highest relevant results at Q2, Q6, Q7 and Q10. We observe that proposed method achieve more relevant results than the existing ranking method.

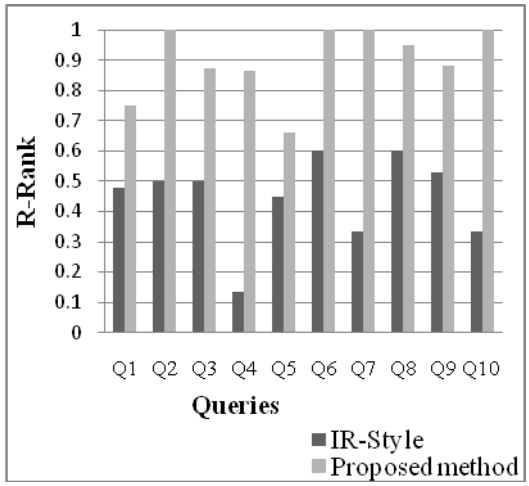

Figure 7. R-Rank Comparison of IR-Style and Proposed Method

6. Conclusion

Keyword search allows ordinary users to find text information in relational databases with much higher flexibility. In this paper, we present the keyword-based search to retrieve relevant queries in relational database by free-style keyword query. A keyword query in the system is a list of keywords and does not need to specify any relation or attributes names. We proposed a new ranking method by adapting the IR ranking techniques based on the virtual document to rank the connected tuple trees, which potentially include tuples from multiple relations by joining tuples in database. The proposed ranking method can reduce the meaningless results which are disappointed for user with the ranking methods in previous works. The experimental results on DBLP show that the

proposed ranking method retrieves the relevant results approximately for the user desired query. We intend to investigate the top-k algorithm to improve query processing as future work.

References

[1] A.Baid, I.Rae, J.Li, A.Doan, J.Naughton, “Toward Scalable Keyword Search over Relational Data,” Proc. VLDB End, Vol. 3, 2010.

[2] F.Liu, C.Yu, W.Meng, “Effective Keyword Search in Relational Databases,” Proc. 2006 ACM SIGMOD Int. conference on Management of data, 2006, pp. 563-574.

[3] G.Li, J.Feng, l.Zhou, “RETUNE: Retrieving and Materializing Tuple Units for Effective Keyword Search over Relational Databases,” Springer, ER, 2008, pp. 469-483.

[4] G.Li, S.Ji, C.Li, J.Feng, “Efficient Type-Ahead Search on Relational Data: a TASTIER Approach,” Proc. 2009 ACM SIGMOD Int. conference on Management of data, 2009, pp. 695-706.

[5] G.Li, X.Zhou, J.Feng, J. Wang, “Progressive Keyword Search in Relational Databases,” IEEE Data Engineering Bulletin, Vol. 5071, 2009, pp. 1183-1186.

[6] J.X.YU, L.Qin, L.Chang, “Keyword Search in Relational Databases: A Survey,” IEEE Data Engineering Bulletin, Vol. 33, 2010, pp. 67-78.

[7] K.Stefanidis, M.Drosou, E.Pitoura, "PerK: Personalized Keyword Search in Relational Databases through Preferences," Proc. 13th Int. Conference on Extending Database Technology, EDBT, 2010, pp. 585-596.

[8] L.Qin, J.X.Yu, L.Chang, “Keyword Search in Databases: The Power of RDBMs,” Proc. 35th SIGMOD Int. Conference on Management of data, 2009, pp. 681-694.

[9] M.M.Thein, “Querying Connected Tuple Trees for Relational Keyword Search,” Proc. Int. Conference on Information Retrieval and Knowledge Management, 2012.

[10]V.Hristidis, Y.Papakonstaninou, “DISCOVER: Keyword Search in Relational Databases,” Proc. 28th Int. Conference on Very Large Data Bases, 2002, pp. 670-681.

[11] V.Hristidis, L.Gravano, Y.Papakonstantinou, "Efficient IR-Style Keyword Search over Relational Databases," Proc. 29th Int. Conference on Very Large Data Bases, 2003, pp. 850-861.

[12] Y.Luo, W.Wang, X.Lin, X.Zhou, “SPARK2: Top-k Keyword Query in Relational Databases,” TKDE Special Issue: Keyword Search on Structured Data, 2011.

[13] Y.Xu, Y.Ishikawa, J.Guan, "Effective Top-k Keyword Search in Relational Databases Considering Query Semantics,” APWeb/WAIM Int. Workshops, 2009, pp. 172–184.

[14] http://www.dblp.uni.trier.de.

[15] S.Wang, J.Zhang, Z.Peng, J.Zhan, Q.Wang, “Study on Efficiency and Effectiveness of KSORD,” APWeb/WAIM International Workshops, 2007, pp. 6-17.