International Journal on Applications in Information and Communication Engineering
Volume 1: Issue 8: August 2015, pp 33-36. www.aetsjournal.com ISSN (Online) : 2394-6237

Advanced Calculation of Complex Keyword Queries Above Databases

C.Princy Diana , K.Ramiah Ilango

Abstract**—** The aim of the project is we analyze and study about the efficient query retrieval in the database. The queries on the databases provide easy access to data to be searched. But we suffer with a problem called low ranking quality which is been shown in the recent benchmarks. We have to work in the ranking quality so that it provides a good user satisfaction. We are going to analyze about hard queries also we are working over the building a framework to measure the difficulty over databases. We are organizing the UN structured data and the content of the database. We evaluate the query predication model so that it provides an efficient access towards data. Our result is to predict a hard query with accurate output. We also present a suite of optimizations to minimize the incurred time that has been taken while searching.

Index Terms - Query performance, query effectiveness, keyword query, robustness, Ranking, databases.

I. INTRODUCTION

Keyword Query Interfaces (*KQIs*) for databases have attracted much attention in the last decade due to their flexibility and ease of use in searching and exploring the data [1]–[5]. Since any entity in a data set that contains the query keywords is a potential answer, keyword queries typically have many possible answers. KQIs must identify the information needs behind keyword queries and rank the answers so that the desired answers appear at the top of the list [1], [6]. Unless otherwise noted, we refer to *keyword query* as *query* in the remainder of this paper.

Databases contain entities, and entities contain attributes that take attribute values. Some of the difficulties of answering a query are as follows: First, unlike queries in languages like SQL, users do not normally specify the desired schema element(s) for each query term. For instance, query *Q*1:

We introduce the problem of predicting the degree of the difficulty for queries over databases. We also analyze the reasons that make a query difficult to answer by KQIs.

We propose the *Structured Robustness (SR)* score, which measures the difficulty of a query based on the differences between the rankings of the same query over the original and noisy (corrupted) versions of the same database, where the noise spans on both the content and the structure of the result entities. We present an algorithm to compute the SR score, and parameters to tune its performance.

We introduce efficient approximate algorithms to estimate the SR score, given that such a measure is only useful when it can be computed with a small time overhead compared to the query execution time.

C.Princy Diana, PG Scholar, SBM College of Engineering. INDIA.
K.Ramiah Ilango. Assistant Professor., SBM College of Engineering.INDIA

The performance of an information retrieval system on a query can be accurately predicted, an informed decision can be made as to whether the query should be expanded, reformulated, biased toward a particular intent or altered in some other way. Increasing evidence points to the fact that valuable clues to a query's ambiguity and quality of corresponding results can be gleaned from query pre-retrieval features, and post-retrieval properties of the query's result set [9]. For example, the query clarity score [7] measures the divergence of a language model over the top-ranked pages from the generic language model of the collection. A separate but related performance prediction problem is to assess the likely effect of query expansion for a given query. Because query expansion is both inherently risky and adds further computational expense, methods for predicting the likely success of expansion and correctly scaling back expansion when it is unlikely to be effective are both valuable.

DATABASES play an important role in today's IT-based economy. Many industries and systems depend on the accuracy of databases to carry out operations. Therefore, the quality of the information (or the lack thereof) stored in the databases can have significant cost implications to a system that relies on information to function and conduct business.

In an error-free system with perfectly clean data, the construction of a comprehensive view of the data consists of linking in relational terms, joining two or more tables on their key fields. Unfortunately, data often lack a unique, global identifier that would permit such an operation. Furthermore, the data are neither carefully controlled for quality nor defined in a consistent way across different data sources. Thus, data quality is often compromised by many factors, including data entry errors (e.g., Microsft instead of Microsoft), missing integrity constraints and multiple conventions for recording information (e.g., 44 W. 4th St. versus 44 West Fourth Street). To make things worse, in independently managed databases, not only the values, but also the structure, semantics, and underlying assumptions about the data may differ as well.

II. RANKING ROBUSTNESS PRINCIPLE FOR STRUCTURED DATA

In this section we present the *Ranking Robustness Principle*, which argues that there is a (negative) correlation between the difficulty of a query and its ranking robustness in the presence of noise in the data. Section 4.1 discusses how this principle has been applied to unstructured text data. Section 4.2 presents the factors that make a keyword query on structured data difficult, which explain why we cannot apply the techniques developed for unstructured data. The latter observation is also supported by our experiments in Section 8.2 on the *Unstructured Robustness Method* [13], which is a direct adaptation of the Ranking Robustness Principle for unstructured data.

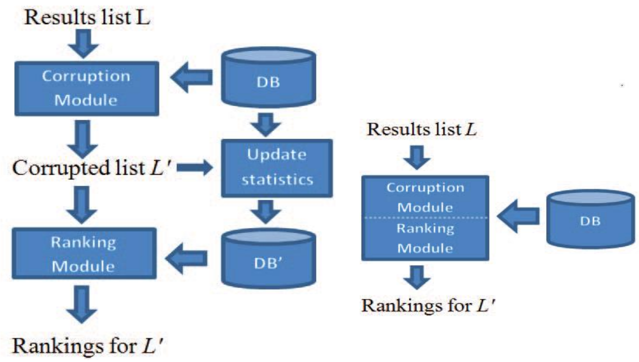
In comparison with hard clustering methods, in which a pattern belongs to a single cluster, fuzzy clustering algorithms allow patterns to belong to all clusters with differing degrees of membership. This is important in domains such as sentence clustering, since a sentence is likely to be related to more than one theme or topic present within a document or set of documents. To enhance the searching techniques we are going for various methods as mentioned. This literature presents an efficient structured robustness algorithm that operates on relational input data. In this system we are also using the ranking technique to get the optimized results.

We propose the *Structured Robustness (SR)* score, which measures the difficulty of a query based on the differences between the rankings of the same query over the original and noisy (corrupted) versions of the same database, where the noise spans on both the content and the structure of the result entities we present the *Ranking Robustness Principle*, which argues that there is a (negative) correlation between the difficulty of a query and its ranking robustness in the presence of noise in the data.

The first challenge in using the Ranking Robustness Principle for databases is to define data corruption for structured data. For that, we model a database DB using a generative probabilistic model based on its building blocks, which are terms, attribute values, attributes, and entity sets. A corrupted version of DB can be seen as a random sample of such a probabilistic model. Given a query Q and a retrieval function g, we rank the candidate answers in DB and its corrupted versions $DB\_,DB\_\_, \dots$ to get ranked lists L and $L\_, L\_\_, \dots$, respectively.

The less similar L is to $L\_, L\_\_, \dots$, the more difficult Q will be. According to the definitions in Section 3, we model database DB as a triplet (S, T, A), where S, T, and A denote respectively. $|A|, |T|, |S|$ denote the number of attribute values, attributes, and entity sets in the database, respectively. Let V be the number of distinct terms in database DB. Each attribute value $Aa \in A$, $1 \le a \le |A|$, can be modeled using a V-dimensional multivariate distribution $Xa = (Xa,1, \dots ,Xa,V)$, where $Xa,j \in Xa$ is a random variable that represents the frequency of term wj in Aa. The probability mass function of Xa is:

$fXa\ (\_xa) = Pr(Xa,1 = xa,1, \dots ,Xa,V = xa,V),$ (1) where $\_xa = xa,1, \dots, xa,V$ and $xa,j \in \_xa$ are non-negative integers.



(b)

Fig. 1. Execution flows of SR Algorithm and SGS-Approx: (a) SR

Corruption of structured data. The first challenge in using the Ranking Robustness Principle for databases is to define data corruption for structured data. For that, we model a database DB using a generative probabilistic model based on its building blocks, which are terms, attribute values, attributes, and entity sets. A corrupted version of DB can be seen as a random sample of such a probabilistic model.Given a query Q and a retrieval function g, we rank the candidate answers in DB and its corrupted versions $DB\_,DB\_\_, \dots$ to get ranked lists L and $L\_, L\_\_, \dots$, respectively.The less similar L is to $L\_, L\_\_, \dots$, the more difficult Q will be. According to the definitions in Section 3, we model

database DB as a triplet (S, T, A), where S, T, and A denote the sets of entity sets, attributes, and attribute values in DB, respectively. $|A|, |T|, |S|$ denote the number of attribute values, attributes, and entity sets in the database, respectively. Let V be the number of distinct terms in database DB. Each attribute value $Aa \in A$, $1 \le a \le |A|$, can be modeled using a V-dimensional multivariate distribution $Xa = (Xa,1, \dots ,Xa,V)$, where $Xa,j \in Xa$ is a random variable that represents the frequency of term wj in Aa.

The probability mass function of Xa is:

$fXa\ (\_xa) = Pr(Xa,1 = xa,1, \dots ,Xa,V = xa,V),$ (1) where $\_xa = xa,1, \dots, xa,V$ and $xa,j \in \_xa$ are non-negative integers. Random variable $XA = (X1, \dots ,X|A|)$ models attribute value set A, where $Xa \in XA$ is a vector of size V that denotes the frequencies of terms in Aa. Hence, XA is a $|A| \times V$ matrix. The probability mass function for XA is:

$fXA(x\_) = fXA(x\_1, \dots, x|\_A|) = Pr(X1 = x\_1, \dots ,X|A| = x|\_A|),$ (2)

where $\_\ xa \in \_x$ are vectors of size V that contain non-negative integers. The domain of $\_x$ is all $|A| \times V$ matrices that contain non-negative integers, i.e. $M(|A| \times V)$. We can similarly define XT and XS that model the set of attributes T and the set of entity sets S, respectively. The random variable $XDB = (XA,XT ,XS)$ models corrupted versions of database DB. In this paper, we focus only on the noise introduced in the content (values) of the database. In other words, we do not consider other types of noise such as changing the attribute or entity set of an attribute value in the database. Since the membership of attribute values to their attributes and entity sets remains the same across the original and the corrupted versions of the database, we can derive *XT* and *XS* from *XA*. Thus, a corrupted version of the database will be a sample from *XA*; note that the attributes and entity sets play a key role in the computation of *XA* as we discuss in Section 5.2. Therefore, we use only *XA* to generate the noisy versions of *DB*, i.e. we assume that *XDB* = *XA*. In Section 5.2 we present in detail how *XDB* is computed.

Structured Robustness calculation. We compute the similarity of the answer lists using Spearman rank correlation It ranges between 1 and −1, where 1, −1, and 0 indicate perfect positive correlation, perfect negative correlation,

and almost no correlation, respectively query *Q* over database *DB* given retrieval function *g*:

SR(Q, g,DB,XDB)

= E{*Sim(L(Q, g,DB), L(Q, g,XDB))*}

= *Sim(L(Q, g,DB), L(Q, g, _x))fXDB (_x),* (3)

where *_x* ∈ *M(|A|× V)* and *Sim* denotes the Spearman rank correlation between the ranked answer lists.

III. EFFICIENT STRUCTURED ROBUSTNESS ALGORITHMS

Computes the exact SR score based on the top *K* result entities. Each ranking algorithm uses some statistics about query terms or attributes values over the whole content of DB. SR Algorithm generates the noise in the DB on-the-fly during query processing. Since it corrupts only the top *K* entities, which are anyways returned by the ranking module, it does not perform any extra I/O access to the DB, except to lookup some statistics. Moreover, it uses the information which is already computed and stored in inverted indexes and does not require any extra index.

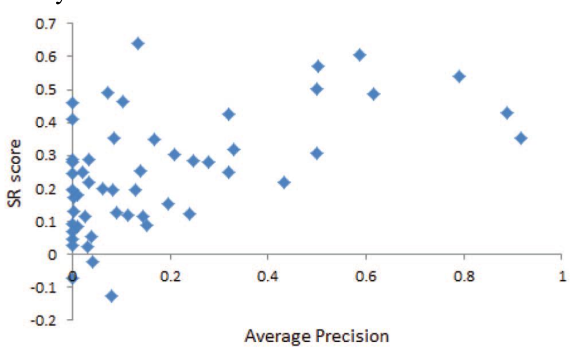


Fig. 2. Average precision versus SR score using IR-Style over SemSearch with K = 20.

Algorithm:

Input: Documents to be structured and ranked.
Output: Ranked and structured documents.
For (i=1 to n) Do
Compute the results by searching the documents
Calculate and cache them
For (j= 1 to n)
Calculate the rank for the query that is given as input
Update the rank and structured data.

IV. ELIMINATING NOISE GENERATION

When we search the data in the large database we can get duplicate results so that we are framing a noise reduction algorithm in this module. In this we are creating an noise generation model to overcome the noises. Displaying the exact query results.

Require: String A of length m, string B of length n

Ensure: normalized Levenshtein Edit-distance between A and B

Step 1: Create 2D array d0..m,0..n

// for all i and j, di,j holds the Levenshtein distance between the first i characters

of A and the first j characters of B; note that d has (m + 1) × (n + 1) values

Step 2: for i = 0 to m do

Step 3: di,0 ← i // distance of null substring of B from A1...i

Step 4: end for

Step 5: for j = 0 to n do

Step 6: d0,j ← j // distance of null substring of A from B1...j

Step 7: end for

Step 8: for j = 1 to n do

Step 9: for i = 1 to m do

Step 10: if Ai == Bj then

Step 11: di,j ← di−1,j−1 // no editing required

Step 12: else

Step 13: di,j ← min(di−1,j , di,j−1, di−1,j−1) + 1 // deletion, insertion, substitution

Step 14: end if

Step 15: end for

Step 16: end for

Step 17: NED(A, B) = dm,n

max(|A|,|B|)

STEP 18: RETURN NED(A, B) // NORMALIZED EDIT DISTANCE

V. CONCLUSION

The proposed strategies and techniques that are being suggested for efficient query retrieval has been satisfied the user needs. The solution is based on the structured framework that is being implemented by structured robustness algorithm. The structured model gives an efficient output when compared to old models. The result we got has efficient ranking display and with an optimized output.

REFERENCES

[1] V. Hristidis, L. Gravano, and Y.Papakonstantinou, "Efficient IRstyle keyword search over relational databases," in *Proc. 29th VLDB Conf.*, Berlin, Germany, 2003, pp. 850–861.

[2] Y. Luo, X. Lin, W. Wang, and X. Zhou, "SPARK: Top-k keyword query in relational databases," in *Proc. 2007 ACM SIGMOD*, Beijing, China, pp. 115–126.

[3] V. Ganti, Y. He, and D. Xin, "Keyword++: A framework to improve keyword search over entity databases," in *Proc. VLDB Endowment*, Singapore, Sept. 2010, vol. 3, no. 1–2, pp. 711–722.

[4] J. Kim, X. Xue, and B. Croft, "A probabilistic retrieval model for semistructured data," in *Proc. ECIR*, Tolouse, France, 2009, pp. 228–239.

[5] N. Sarkas, S. Paparizos, and P. Tsaparas, "Structured annotations of web queries," in *Proc. 2010 ACM SIGMOD Int. Conf. Manage. Data*, Indianapolis, IN, USA, pp. 771–782.

[6] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in databases using BANKS," in *Proc. 18th ICDE*, San Jose, CA, USA, 2002, pp. 431–440.

[7] C. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*. New York, NY: Cambridge University Press, 2008.

[8] A. Trotman and Q. Wang, "Overview of the INEX 2010 data centric track," in *9th Int. Workshop INEX 2010*, Vugh, The Netherlands, pp. 1–32,

[9] T. Tran, P. Mika, H. Wang, and M. Grobelnik, "Semsearch ´S10," in *Proc. 3rd Int. WWW Conf.*, Raleigh, NC, USA, 2010.

[10] S. C. Townsend, Y. Zhou, and B. Croft, "Predicting query performance," in *Proc. SIGIR '02*, Tampere, Finland, pp. 299–306.

[11] A. Nandi and H. V. Jagadish, "Assisted querying using instantresponse interfaces," in *Proc. SIGMOD 07*, Beijing, China, pp. 1156–1158.

[12] E. Demidova, P. Fankhauser, X. Zhou, and W. Nejdl, "DivQ: Diversification for keyword search over structured databases," in *Proc. SIGIR' 10*, Geneva, Switzerland, pp. 331–338.

[13] Y. Zhou and B. Croft, "Ranking robustness: A novel framework to predict query performance," in *Proc. 15th ACM Int. CIKM*, Geneva, Switzerland, 2006, pp. 567–574.

[14] B. He and I. Ounis, "Query performance prediction," *Inf. Syst.*, vol. 31, no. 7, pp. 585–594, Nov. 2006.

[15] K. Collins-Thompson and P. N. Bennett, "Predicting query performance via classification," in *Proc. 32nd ECIR*, Milton Keynes, U.K., 2010, pp. 140–152.

[16] A. Shtok, O. Kurland, and D. Carmel, "Predicting query performance by query-drift estimation," in *Proc. 2nd ICTIR*, Heidelberg, Germany, 2009, pp. 305–312.