Efficient Keyword Search Across Relational Databases

Gayatri Priya Madhira
Post-Graduate Student
Department of Computer Science and Engineering
JNTU College of Engineering (Autonomous)
Anantapur-515002, AP, India
madhiragayatri@gmail.com

Dr. K.F. Bharati
Assistant Professor
Department of Computer Science and Engineering,
JNTU College of Engineering (Autonomous),
Anantapur-515002, AP, India
kfbharati@gmail.com

Abstract— In Traditional Keyword search system over relational data bases, a keyword query is submitted by the user to the system and the relevant results are retrieved, where the user has restricted knowledge about the data. While issuing queries the user often feels "Left in the dark" and for finding the information the user has to use try-and see approach. Here we guild the information based on the user needs for constructing a keyword query that provides easy access to data from the database. Low ranking quality is often observed even though easy access to data is provided by the keyword queries on databases. For such hard queries, the user may advise to the user alternative queries. We propose a novel framework to predict hard query using algorithms and ranking methods and provide relevant results according to the user needs with high accuracy.

Keywords—Keyword search; Relational Data bases; Structured Robustness; Query Effectiveness

I. INTRODUCTION

Data mining is the process of sifting through very large amounts of data for useful information. Now-a-days much attention is observed in the keyword query interfaces (KQI) for relational databases for searching and exploring the data [1], [2]. Keyword queries often suffer from ambiguity known as hard queries. User information needs must be identified by the KQIs behind keyword queries and rank the answers to be displayed on the top of the list [1], [3].

The degree of difficulty for queries over databases is predicted and characteristics of those hard queries are analyzed. The Structured Robustness (SR) score is used to measure the query difficulty based on the rankings of original and corrupted versions of the data base. An algorithm is used to compute the SR score to enhance the performance. SR score is estimated with efficient approximation algorithms that can be computed with small time overhead. . INEX [4] and SemSearch are the two standard benchmarks that have been provided for keyword search methods over databases. The ranking quality of the ranking algorithms is effectively predicted by the SR score. The SR score computation time is negligible compared to the query execution time.

II. RELATED WORK

To predict hard queries over unstructured text documents two methods have been proposed.

Pre-retrieval methods [6], [9] without computing the results the difficulty of a query is predicted. Specificity, ambiguity or term relatedness is measured to predict the query difficulty using statistical properties. The statistical characteristics are like the number of attributes that are relevant to the query terms or the number of entities that are related to the query terms [10]. If the query terms are more distinct, then the query will be very easier. But have low performance.

Post-retrieval methods: The results of a query are utilized for predicting the query difficulty. These methods have better prediction accuracy and are categorized as below.

Clarity score based: This method assumes that the user is interested in very few topics. So the query is said to be easy if its results belongs to very few topics [5], [6], [7], [11]. However, domain knowledge is required about the datasets. So, the idea of clarity score for queries over databases is extend.

Ranking-score-based: The similarity of the query and the document returned by the retrieval systems is estimated using ranking score. Based on the score distribution of results the degree of difficulty of a hard query is measured [8].

Robustness-based: This method contends the results of an easy query are comparatively stable against the disruption of queries [12], documents or ranking algorithms. The proposed system falls in this category.

III. DATA AND QUERY MODELS

A database is modeled as a set of entity sets where each entity set S comprises a collection of entities E. For instance teams and players are two entity sets in cricket database. Fig 1 depicts a portion of the data set where each sub tree root is named as team and represents an entity. Each entity has set of attribute values A_i, $1 \leq I \leq |E|$. Each attribute value is a collection of terms. Based on current un-structured and semi-structured retrieval methods, we ignore stop words that appear in the attribute values. The attribute T is belonged to every attribute value A and is written as A Є T. For example, Virat Kohli, Anil and Shane Watson are three attribute values in the team entity shown in the cricket team entity shown in the figure. Node 2 depicts the attribute of *Indian team*, which is *coach*.

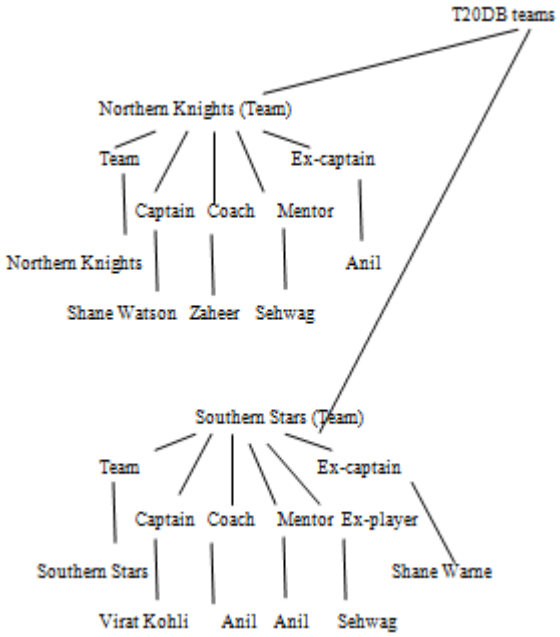


Figure 1. Cricket database Fragment

The above models an abstract data model. In this paper, we ignore the physical representation of data. That is, an entity could be stored as a set of normalized related tables or in an XML file. This model has been used to a great extent in works *data-centric XML retrieval* [4] and *entity search* [2], and there is a great advantage so it can be mapped easily to either XML or relational data. Further, if a KQI method depends on the internal complexities of the database design, it will not be strong and will have considerably low degree of effectiveness over different databases. Therefore, the principled formal models have developed that cover all databases and data formats reasonably well, we do not consider the internal complexities or the database design structure or data format in models.

A *keyword query* is a defined as set $Q = \{q1. . . q1_{|Q|}\}$ of terms, where the number of terms in *Q is denoted as* $|Q|$. An entity *E* is treated as an *answer* to *Q* if a term q_i in *Q* is present in one of its attribute values *A*, written as $q_i Є A^1$. Given a database *DB* and a query *Q*, the retrieval function *g (E, Q, DB)* gives a real number that reflects the relevance of the entity *EЄDB* to the keyword query *Q*. Given a database *DB* and a query *Q*, the system returns entities as a ranked list in *DB* named *L(Q, g, DB)* the entities *E* are placed in order or decreasing of the value *g(E,Q,DB).*

IV. RANKING ROBUSTNESS PRINCIPLE

The degree of the difficulty of a hard query is correlated with the robustness of ranking across the corrupted and the original versions of the database called the Ranking Robustness Principle.

PROPERTIES OF HARD QUERIES ON DATABASE

It is a well-established fact that *the candidate answers of a query are diverse to a greater extent; the harder the query is* over a collection of text documents. We use this idea and extends it for queries over relational databases and suggest three sources of difficulty for answering a keyword query over a database as follows:

1) If the query term matches more entities, then there is less specificity of the query and the harder it is to answer properly. For instance, there is more than one player named Shane in the T20 data set. If a user submits the query *Q*2: *Shane*, the KQI must resolve the expected *Shane* that satisfies the user's information need. As

opposed to *Q*2, *Q*3: *Sehwag* matches smaller number of people in the cricket DB, so it is easier for the KQI to return its relevant results.

2) Each attribute describes various aspects of an entity and determines context of terms in its attribute values. If various attributes are matched with a query in its candidate answers, then it will have a wider collection of potential answers in the database, and hence it has higher level of ambiguity at the attribute level. For instance, some candidate answers for query *Q*4: *AnilKumble* in IPL DB contain its term in their *mentor* and some contain its term in their *Ex-captain*. A KQI must identify the desired matching attribute for *AnilKumble* to find its relevant answers. As opposed to*Q*4, query *Q*5: *Virat Kohli does* not match any instance of attribute *mentor*. Hence, a KQI already knows the desired matching attribute for *Q*5 and the task is easier to perform.

3) Each entity set comprises the data about a various type of entity and defines another level of context for terms. Therefore, if a query is matching entities from major entity sets, it will have greater level of ambiguity at entity level. For instance, T20DB contains the information about teams in an entity set called *team* and the information about the mentor involved within the team in another entity set called *mentor*. Consider query *Q*6: *ex-player* over IPL DB data set whose candidate answers come from both entity sets. However, teams having mentor and ex-player cannot both satisfy information need of query *Q*6. A KQI has a difficult task to do as it has to identify if the information need behind this query is to find players who are ex-players or current players. In contrast to *Q*6, *Q*7: *owner wicket-keeper* matches only entities from *team* entity set. It is less hard for a KQI to answer *Q*7 than *Q*6 as *Q*7 has only one potential desired entity set.

V. STRUCTURED ROBUSTNESS

Corruption of organized data: The primary challenge we face with the usage of the Ranking Robustness Rule for databases is with defining the data corruption for structured data. For defining the data corruption, we create a model database using a conceptive and probabilistic model based on the building blocks which comprise of terms, entity sets, attributes and attribute values. A sample at random of such a probabilistic model is what we define as a corrupted version of the DB. For retrieval function h and query S, we will rank the candidate answers in the DB and its corrupted versions say DB', DB"… and the ranked lists are acquired say L, L', L"… respectively. The lesser the similarity between L and L', L" … the more hard Q will be.

Structured Robustness computation: We use Spearman rank correlation [13] to compute the similarity of the answer lists. This value ranges between -1 and 1, where -1 indicated perfect negative correlation, 0 indicates almost no correlation and 1indicates perfect positive correlation. The Structured Robustness score (*SR* score), for a query *Q* over the database *DB the* given recuperation function *h*:

$$SR(S, h, DB, XDB) = E\{Sim(L(S, h, DB), L(S, h, X_{DB}))\}$$

Here the Spearman rank interdependencies denoted by *Sim* between the ranked answer lists.

VI. STRUCTURED ROBUSTNESS(SR) PRINCIPLE

The Structured Robustness Principle, based on the top K result entities computes the exact SR score. Some statistics were used by each ranking principle about attributes values or query terms over the whole content of DB. The examples of that census are the number of accuracies of a query term in overall attributes values of the DB or in each attribute and entity set the total number of attribute values that occur.

VII. APPROXIMATION PRINCIPLESS

Here we propose approximation principles to improve the proficiency of SR principle. The methods proposed are autonomous of the prime ranking principle.

Query-specific Attribute values Only Approximation: It padded only those attribute values in the database that match at least one query term. This approximation principle makes use of the following observations:

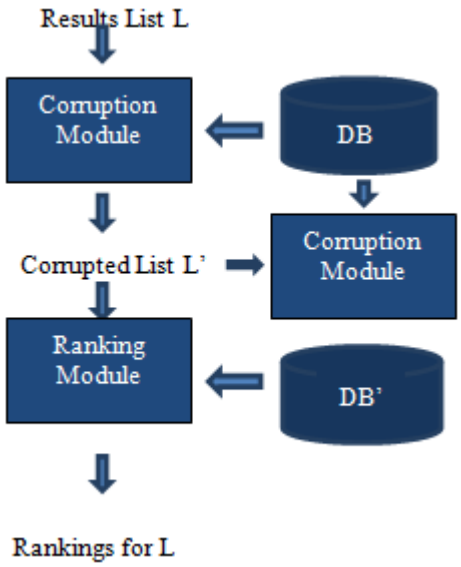
Observation 1.*The corruption effect is much lesser than the noise in the attributes that comprise query terms.*

Observation 2.*The number of all attribute values in each entity is much larger than the number of attribute values that comprise at least one query term.*

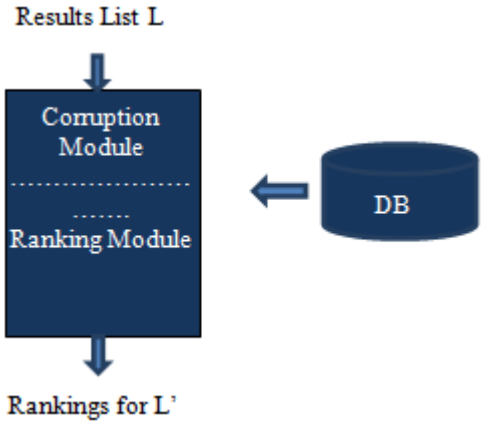
Hence, the time spent on corruption is significantly decreased if we corrupt only the attribute values that contain query terms.

Static Global Stats Approximation: The following observations are used in SGS-Approximation:

Observation 3.*If only the top-K result entities are padded, the change in the global DB statistics is not much.*



(a) SR Principle



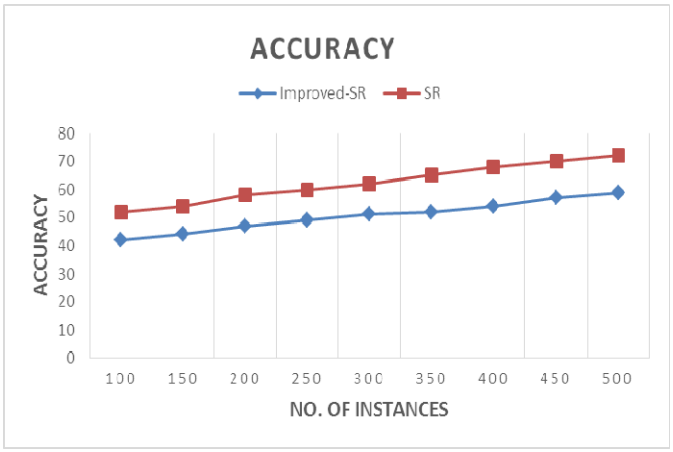
(b) SGS-Approx

Figure 2. Execution flows of SR Principle and SGS-Approx: (a) SR Principle. (b) SGS-Approx.

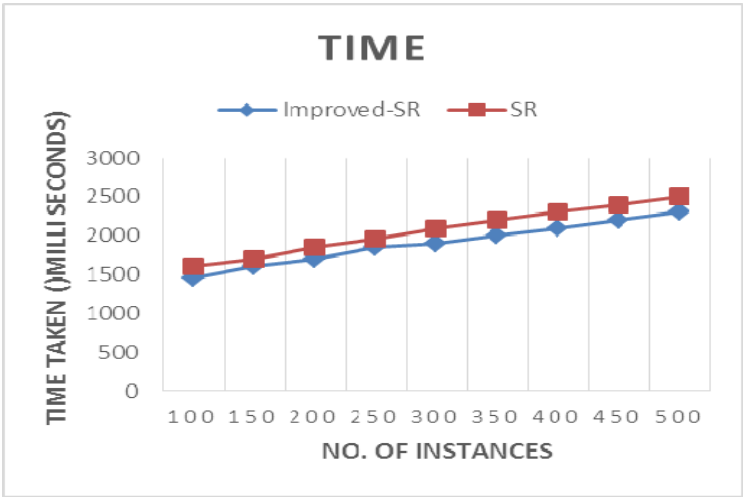
Fig. 2(a) illustrates the flow of execution for SR Principle. After the ranked list of top *K* entities for *S is got*, the corruption module produces the entities that are corrupted and the global stats of *DB are updated*. Then, the corrupted results are passed by the SR Principle and the global statistics is updated to the ranking module to compute the corrupted ranking list. A large portion of the robustness calculation time is spent by the SR Principle on the loop that re-ranks the corrupted database, and for updating global statistics. Here the value of *K* is very small compared to the number of entities in the DB; the top *K* entities comprise a very little portion of the DB. Therefore, the global statistics change a little or remain unchanged. So, the global statistics is used for the original version of the DB and the corrupted entities are re-ranked. We are restricting the updating of global statistics by combining the corruption and the ranking modules together. So, during corruption the re-ranking is done progressively. *SGS-Approx* principle is shown in Fig. 2(b).

VIII. IMPROVED SR-PRINCIPLE

The main problem in the current SR-principle is that corrupting each column will take a lot of time for query processing. Instead, the entropy is calculated in the columns and will decide whether to corrupt the columns or not based on the obtained entropy value. The entropy of the column is calculated, if the entropy is above certain threshold the corruption is done. If the entropy is below the threshold value the corruption is not needed. By this way the query execution time is reduced. For example input is given to the Keyword Search system, then the time taken to execute the query and accuracy using SR and Improved SR-Principles are shown in the graphs.



3(a)



3(b)

Figure 3. Comparing (a) Accuracy and (b) Time using SR-Principle and Improved SR-Principle

Fig. 3(a) and 3(b) shows the results using SR and Improved SR-Principle. In fig 3(a) the X-axis is taken as number of instances and y-axis as accuracy. In fig 3(b) the X-axis is taken as number of instances and Y-axis as time taken in milliseconds. Thus, using the Improved SR-Principle the accuracy is increased and query execution time is decreased compared to SR-Principle.

IX. CONCLUSION

By analyzing the characteristics of a query the hard queries are predicted. A novel Principle called SR Principle is used to predict the hardness of a query and QAO-Approximation for corruption of attributes over original database and the similarity is computed over the corrupted and the original versions of database and the effectiveness of keyword queries is predicted. To improve the accuracy and reduce the query execution time the entropy is calculated for the columns and will decide to corrupt the database or not.

ACKNOWLEDGMENT

I am grateful to Dr. K. F. Bharati, Assistant professor, in the Department of Computer Science and Engineering. I am extremely thankful and indebted to her for sharing expertise, and sincere and valuable guidance and encouragement extended to me.

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] 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.
[3] 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.
[4] 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,
[5] S. C. Townsend, Y. Zhou, and B. Croft, “Predicting query performance,” in Proc. SIGIR ’02, Tampere, Finland, pp. 299–306.
[6] B. He and I. Ounis, “Query performance prediction,” Inf. Syst., vol. 31, no. 7, pp. 585–594, Nov. 2006.

[7] K. Collins-Thompson and P. N. Bennett, “Predicting query performance via classification,” in Proc. 32nd ECIR, Milton Keynes, U.K., 2010, pp. 140–152.
[8] A. Shtok, O. Kurland, and D. Carmel, “Predicting query performance by query-drift estimation,” in Proc. 2nd ICTIR, Heidelberg, Germany, 2009, pp. 305–312.
[9] Y. Zhao, F. Scholer, and Y. Tsegay, “Effective pre-retrieval query performance prediction using similarity and variability evidence,” in Proc. 30th ECIR, Berlin, Germany, 2008, pp. 52–64.
[10] C. Hauff, L. Azzopardi, and D. Hiemstra, “The combination and evaluation of query performance prediction methods,” in Proc.31st ECIR, Toulouse, France, 2009, pp. 301–312.
[11] C. Hauff, V. Murdock, and R. Baeza-Yates, “Improved query difficulty prediction for the Web,” in Proc. 17th CIKM, Napa Valley, CA, USA, 2008, pp. 439–448.
[12] E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow, “Learning to estimate query difficulty: Including applications to missing content detection and distributed information retrieval,” in Proc.28th Annu. Int. ACM SIGIR Conf. Research Development Information Retrieval, Salvador, Brazil, 2005, pp. 512–519.
[13] J. D. Gibbons and S. Chakraborty, Nonparametric Statistical Inference. New York, NY: Marcel Dekker, 1992.

BIOGRAPHIES

Gayatripriya Madhira received B.Tech degree in Computer Science and Engineering from Sri Padmavathi Mahila University, Tirupathi, A.P, India, during 2009 to 2013. Currently pursuing M.Tech in Computer Science from Jawaharlal Nehru Technological University College of Engineering, Anantapuramu, A.P, India, during 2013 to 2015 batch. Her Area of interests includes Data Mining, Cloud Computing, and Computer Networks.

Dr.K.F.Bharati is currently working as an Assistant Professor in the Department of Computer Science and Engineering in JNTUA College of Engineering, Anantapur, A.P, India. She has received her Ph.D. from JNTU Anantapur. She obtained her M.Tech from Visveswariah Technological University, Belgaum. She did her B.Tech from University of Gulbarga. She is also Officer in charge for Central Computer Center, JNTUCEA.