J. Appl. Environ. Biol. Sci., 4(9S)275-280, 2014

ISSN: 2090-4274
Journal of Applied Environmental and Biological Sciences
www.textroad.com

Probability Computation for Rank Joins Using Weibull Distribution

Zaheer Ahmad[1], Adnan Abid[2]

[1]Faculty of Information Technology, University of Central Punjab Lahore, Pakistan
[2]Department of Computer Science, University of Management and Technology, Lahore, Pakistan

Received: September 12, 2014
Accepted: November 23, 2014

ABSTRACT

Ranking queries produce results that are ranked on some pre-computed score. Typically, these queries involve joins, where users are usually interested only in the top-K join results. Current relational query processors handle ranking queries efficiently, however, in case of top-k join queries involving distributed or web based data sources which have a non negligible response time for data extraction, the existing algorithms do not perform well in terms of time taken. Focus of this paper is to compute the join results efficiently, while minimizing the time to compute top-k join results, as well as reducing the number of data extractions from these data sources. As a principal contribution, we present a probabilistic method to compute the top-k join results efficiently, and we have found the initial results of this research promising.

KEYWORDS*:* Rank Joins, Top-K, Deterministic Reporting, Provisional Reporting, Score Aggregation, Threshold

1 INTRODUCTION

People who are using internet for their source of information are used to thinking of web as source of answering their questions. WebPages presenting the information like weather of some certain city or place, a rout to some destination or result of a sports match can be obtained through a keyword search. The simplest solution of a question on internet is that the target webpage which has the answer to our query may be linked by the search engine and showed on the first page of search engine results. A survey of top-k techniques has been presented by Ilyas et. al. (2008) [8] Searching becomes a bit complex when user renders a query whose required information is spanned over multiple Web Pages and cannot be obtained from single type of websites.

Example: Which is the best restaurant offering Italian Pizza in nearby locations. It can be solved by surfing the web multiple times and by collecting the partial results. First of all website containing the best restaurants will be identified and some number of top restaurants will be kept in mind. Secondly we will go for website containing best Italian Pizza and store its top results too on some place. Then finally we will determine which of the results of both searches belong to our nearby locations. While doing so we are performing information integration in our mind specifically, we are applying ranking while extracting restaurants and Italian Pizzas then matching based on nearby locations. The main question is how to get desired results efficiently and produce a specific number of top joined results against a query during the process of getting information from different web sources. This involves various data sources that generally store the data in relational databases, and are accessible through the Web. We go through the existing rank join algorithms, which are also known as top-K join algorithms. Main concern of this research is to compute the join results efficiently, while minimizing the number of data extracts to these data sources. In addition to this, how to improve the efficiency of the ranking process to report a top join results to the users. In order to accomplish this we use probabilistic method to provisionally report a join result. Rank joins have a well-known applicability in many domains. As a result several algorithms have been introduced in the literature to process top-K queries efficiently (Bruno et al. 2002 a, b) [5, 6]; Fagin and Wimmers 1997 [77]; Ilyas et al. 2004 [9]; Marian et al. 2004 [1010]; Theobald et al. 2004[**Error! Reference source not found.**11]. A majority of such algorithms involves deterministic measures for processing the data, that is, the algorithm stops when there is an absolute certainty that correct top-K results have been identified. The HRJN is one of these algorithms [9] which is presented by Ilyas et al in 2004. This algorithm has been used as baseline to provisionally report the join result.

*** Corresponding Author:** Zaheer Ahmad, Faculty of Information Technology, University of Central Punjab Lahore, Pakistan. zheer190@gmail.com

2. PROBLEM SETTINGS

This algorithm produces top-K joins from data sources of different set of objects, combined on the join attribute value. The objects in all lists are sorted in descending order of scores. Sequential data access to these objects has been used, here we can say an object at depth d will be retrieved before the object at depth greater than this i.e. d+1. During execution, this algorithm calculates join results and stores them in an output buffer s_{buff}. The results are stored in output buffer s_{buff} in descending order of their aggregate score. The size of this output buffer is bounded by the value K, which are the number of required top join results by user. Just like remaining rank join algorithms, HRJN algorithm also asserts an upper bound on the scores of join results that can be created by using score aggregation function with the unseen data. Generally, we call this upper bound as threshold value, and this threshold value decreases with every iteration of extraction of data. The algorithm reveals those join results to the user which have combined score greater than or equal to the threshold.

3. METHODOLOGY

Figure 1 shows a snapshot taken during the execution of HRJN on two data sources S1 and S2, for K = 5. Let us assume that the rows with gray background have been extracted, and the rest are yet to be extracted. The score aggregation function in this case is the sum of individual scores. Furthermore, based on this score aggregation function we can see that for S1 potentially maximum possible score of a join result obtained from its unseen data can be 1.97, which is obtained by joining its last observed score with the maximum score of S2. Similarly, for S2 it is 1.96. The maximum of these two is the threshold which in this case is 1.97. Hence, at this stage, the first two join results in S_{buff} can be reported to user as top join results with absolute certainty. However, at the same time we can observe that there are other join results in the output buffer. So, in order to be sure whether these other join results in the output buffer belong to the top-K set or not, we have to wait until the threshold becomes less than or equal to 1.94, i.e. score of the K^{th} join result in the output buffer. This is possible if we extract two more tuples from S1 and one more tuple from S2. However, by analyzing the data shown in Figure 1 we find that even after extracting these tuples we shall not be able to find any new join result having score greater than the K^{th} join result in the buffer.

R1

ID	Join Attrib	Score
R1.1	A	1.0
R1.2	B	0.99
R1.3	C	0.98
R1.4	A	0.97
R1.5	D	0.95
R1.6	B	0.93
R1.7	A	0.92

R2

ID	Join Attrib	Score
R2.1	B	1.0
R2.2	C	0.98
R2.3	A	0.97
R2.4	B	0.96
R2.5	D	0.94
R2.6	E	0.92

S_{buff}

Join ID	Join Attrib	Score
R1.1–R2.1	B	1.99
R1.1–R2.3	A	1.97
R1.3–R2.2	C	1.96
R1.2–R2.4	B	1.95

Figure 1: Example for rank join computed on some aggregation score function

4. PROVISIONAL REPORTING OF JOIN RESULTS:

By analyzing the data shown in Figure 1 it is found that even after extracting these tuples we shall not be able to find any new join result having score greater than the K^{th} join result in the buffer. Therefore, this can be argued that there should be a method to compute the confidence with which an unreported join result in the output buffer will be a part of the final top-K join results. If this confidence is high enough e.g. greater than 0.80 or 0.90 then this join result can be reported to the user even if its score is below the current threshold value. This will help in reporting join results to the user quickly. Every data fetch from a

Web service involves a reasonable time [2], and data extraction k2time overcomes the time needed to process it. Figure 2 shows such scenario, where join results obtained from joining the data of Web services are kept in the output buffer for a reasonable time before they can be reported. Here, it can be seen that after 2000ms only two join results have been reported, and there is considerable difference between their creation time and deterministic reporting time. In order to report a join result in the output buffer with deterministic guarantees the threshold of the system has to become less than or equal to its score, which is only possible with further data extraction. Therefore, in order to report such join result it is necessary to wait for the amount of time needed for next data extractions to be reported with deterministic guarantee.

That approach which computes the confidence by help of probability is called the provisional reporting of rank join results. The technique provisional reporting is based on measuring of confidence of result and in turn confidence is measured by calculating the probability of candidate result. With which an unreported join result will be among the final top-K. The provisional reporting technique reports all such un-reported join results which have probability greater than a given threshold probability. This will help reporting join results in quick time. Several experimental studies have been conducted [1, 3, 4, 11] with various settings of different operating parameters which validate the importance of the proposed approach on both real and synthetic data sets.

Join Results	Score	Cr. Time (ms)	Deterministic Reporting Time
$S_{buf}(1)$	1.95	500	1000
$S_{buf}(2)$	1.92	1000	2000
$S_{buf}(3)$	1.91	500	
$S_{buf}(4)$	1.89	1500	
$S_{buf}(5)$	1.87	2000	
$S_{buf}(6)$	1.84	500	
$S_{buf}(7)$	1.83	1500	
$S_{buf}(8)$	1.80	2500	

Time (ms)	Threshold
500	1.96
1000	1.94
1500	1.93
2000	1.90
2500	1.87

Figure 2: Execution of rank join using web services

In the method of provisional reporting we use Weibull probability distribution to compute the confidence of unreported join result in output buffer s_{buff}. Amongst large number of continuous probability distribution, we choose Weibull distribution due to its properties which support this type of probability estimation. The Weibull distribution is generally used in survival analysis, in reliability engineering and failure analysis, in industrial engineering to represent manufacturing and delivery times. We make use of this probability distribution for ranking the join results provisionally.The aggregation scoring function of Rank Join generates score values which are not integers but Real Numbers. These values are continuous in nature, and may marginally differ by points e.g. 1.99, 1.97, 1.84 etc. While handling such type of values we are essentially dealing with the continuous probability distributions, and Weibull is a continuous probability distribution.

Weibull Probability Distribution: The random variable of Weibull Distribution has probability density function (pdf) as under:

$$f(x;\alpha,\beta) = \begin{cases} \frac{\alpha}{\beta^{\alpha}} x^{\alpha-1} e^{-\left(\frac{x}{\beta}\right)^{\alpha}} & x \geq 0 \end{cases}$$

Where α and β are positive values and called shape & scale parameters respectively.

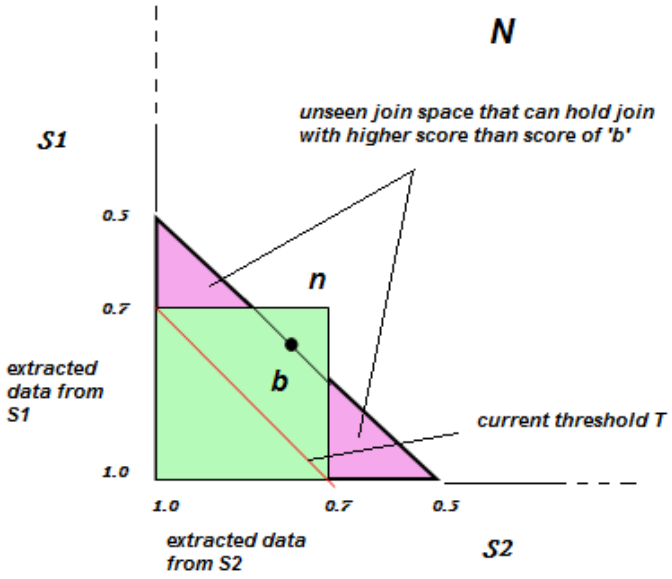

Figure 3: Join space of HRJN with web services S1 & S2

Figure 3 which shows the snapshot during the execution of HRJN using two Web services. It shows that we have extracted a certain amount of data from both Web services, and we have computed 'n' join results from this data. All the observed join results lie in the region covered by the small square. The diagonal line drawn in the small square depicts the current threshold τ. Now, all the join results which are below this diagonal can be deterministically reported by adding them to the top-K result set.

We can see in Figure 3 a join result *'b'* which has score lower than τ. Let us assume that *'b'* is in the output buffer at position *'y'*, we represent it as t(y) buff. In order to report *'b'* with deterministic guarantee the threshold should become lower than or equal to the score of *'b'*, i.e. $\tau \leq \sigma$ (t(y)buff). We may or may not find some new join results with score greater than σ(t(y)buff). Hence, the new position for *'b'* in the output buffer will be z, where $z \geq y$. Figure 3 shows a diagonal line passing through *'b'* which defines minimum threshold value to report *'b'* as a top-K join result deterministically. We can also see two triangles outside the small square, and all the join results which may exist in these two triangular regions will have score greater than σ(t(y)buff). Let p denote the probability with which an unseen result will have an aggregate score larger than that of σ(t(y)buff). For our aggregation function, which is linear addition, q will be the ratio between the explored area to the total area of data.

Therefore, while computing probabilities for join results in s_{buff}, when we encounter first join result whose score is less than current threshold. We will compute the probability of the range from that join result to the last reported result. The probability has been computed by the help of probability distribution function (pdf) of Weibull Distribution According to Fig. 1 third result in buffer has score 1.96 which is less than current threshold and has not been reported yet. The probability of occurrence of record between the last reported join result and first unreported join result (in this case 1.97 and 1.96) will be calculated and then this value is multiplied by the ration q which has already been computed. By doing this we will be calculating the actual probability of occurrence of join result(s) between the given range. Lastly, this resultant value is multiplied to total number of possible join results which could be obtained by joining two data sources. This value gives us a number in terms of join results which may fall between the given range of score, which in this case is $P(1.96 \leq x < 197) = WeibullPDF(\ )$

Experiments have been conducted using synthetic data sources. Many different synthetic data sets have been generated using different values of main operating parameters shown in Table 1. We choose to use chunk size (CS) = 10, and response time (RT) = 500ms.

Table 1: Operating Parameters

Parameter Name	Symbol	Range of Values
Probability Distribution	PD	Deterministic reporting, Weibull Distribution
Number of results	K	5, 10
Average difference in Creation Time and Provisional Reporting time of each result		
Average difference in Creation Time and Deterministic Reporting time of each result		

5. Results: It is observed that for various different values of threshold, our proposed method reports the top join results earlier than the deterministic method. The reason is that while experimenting with the synthetic data set top-K results are observed in early stages, but they cannot be reported by the baseline HRJN algorithm, as their score is lower than the threshold. So, in this case, the deterministic approach makes more data extractions to bring threshold down and ensure that there is no other higher scoring join result. Whereas, the probabilistic method reports these top join results earlier and stops.

Table 2: Comparison of Provisional and Deterministic Reporting (K=10)

Reporting method	I/O Cost	Time (ms)	Late Reported	Wrong Reported
Provisional (Weibull)	320	16000	2	1
Deterministic	380	19000	-	-

Figure 4 shows comparison in terms of average time taken to report a tuple after its creation by HRJN and proposed approach for Weibull distribution values. On average, this proposed approach reports a top join result after 2.2 seconds of its creation, whereas, in case of deterministic reporting it takes 4.8 seconds on average. Hence, on average a result is reported provisionally 2.6 seconds before it is reported deterministically using HRJN. It has also been observed during the experiments that there were negligible number of wrong reported join results for all experiments.

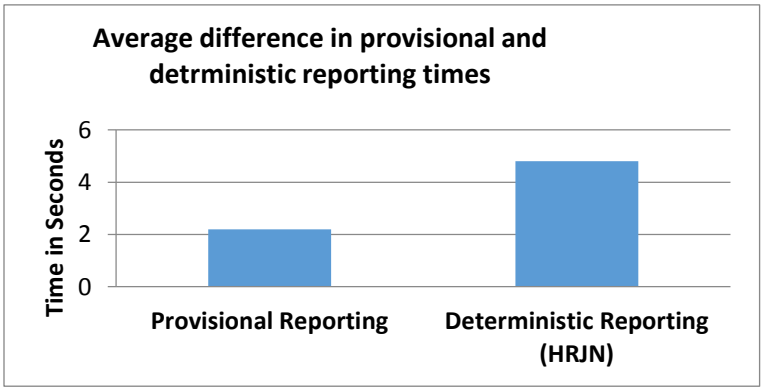

Figure 4: Comparison of Provisional and Deterministic Reporting

6. CONCLUSION AND FUTURE DIRECTIONS

The need of provisional reporting has been highlighted in this research while using rank joins over the Web based data sources. An approach is provided which is meant to compute the confidence with which a join result may be reported to user in a quick time after it is created. The results show that our proposed algorithm works fine with present parameter settings. It is worthwhile to mention here with this proposed method works very efficiently and refrains from useless computations during the computation of the probability.

In future, we intend to extend this work with more detailed experiments to evaluate its credibility for a larger range of problem settings. Similarly, we would like to incorporate Gamma probability distribution to compute the confidence measure.

REFERENCES

1. Abid, A.; Tagliasacchi, M. (2012), Provisional reporting for rank joins, J Intell Inf Syst DOI 10.1007/s10844-012-0234-3.
2. Abid, A.; Tagliasacchi, M. (2011) Parallel Data Access for Multi-way Rank Joins, 11th International Conference on Web Engineering (ICWE) 2011.
3. Arai, B.; Das, G.; Gunopulos, D.; Koudas, N. (2007). Anytime measures for top-k algorithms. In *Proceedings of the 33rd international conference on very large data bases (VLDB '07) VLDB endowment* (pp. 914–925).
4. Arai, B.; Das, G.; Gunopulos, D.; Koudas, N. (2009). Anytime measures for top-k algorithms on exact and fuzzy data sets. *The VLDB Journal, 18*, 407–427.
5. Bruno, N.; Chaudhuri, S.; Gravano, L. (2002). Top-k selection queries over relational databases: mapping strategies and performance evaluation. *ACM Transactions on Database Systems, 27*, 153–187. J Intell Inf Syst
6. Bruno, N.; Gravano, L.; Marian, A. (2002). Evaluating top-k queries over web-accessible databases. In *ICDE* (p. 369).
7. Fagin, R.; Lotem, A.; Naor, M. (2003), Optimal aggregation algorithms for middleware, Journal of Computer and System Sciences 66 (2003) 614–656
8. Ilyas, I.; Beskales, G.; Soliman, M. (2008), A Survey of Top-KQuery Processing Techniques in Relational Database Systems, ACM Computing Surveys, Vol. 40, No. 4, Article 11.
9. Ilyas, I.; Aref, W.; Elmagarmid, A. (2004). Supporting top-K join queries in relational databases. *The VLDB Journal, 13*(3), 207–221.
10. Marian, A.; Bruno, N.; Gravano, L. (2004). Evaluating top-*k* queries over web-accessible databases. *ACM Transactions on Database Systems, 29*(2), 319–362.
11. Theobald, M.; Weikum, G.; Schenkel, R. (2004). Top-k query evaluation with probabilistic guarantees. In *Proceedings of the thirtieth international conference on very large data bases, VLDB endowment (VLDB '04)* (Vol. 30, pp. 648–659).