IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 10, December 2014.
www.ijiset.com
ISSN 2348 – 7968

Estimation of Potential Product Using Reverse Top-k Queries

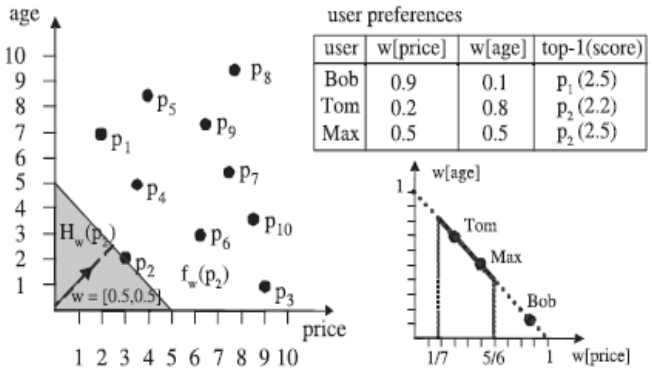
T.Sathis Kumar
Assistant Professor, Department of Computer Science and Engineering,
Saranathan College of Engineering,Trichy,Tamilnadu,India.
Sathis_trichy@yahoo.com

Abstract—Atpresent, most of the applications return to the user a limited set of ranked results based on the individual user's preferences, which are commonly validated through top-k queries. From the perspective of a manufacturer, it is imperative that the products appear in the highest ranked positions for many different user preferences, otherwise the product is not visible to the potential customers. In this paper, I define a novel query type, namely the reverse top-k query, that covers the requirement of potential product, which are the user preferences that make this product belong to the top-k query result set. Reverse top-k queries are essential for manufacturers to assess the impact of their products in the market based on the competition. I formally define reverse top-k queries and introduce two versions of the query, monochromatic and bichromatic. First, I provide a geometric interpretation of the monochromatic reverse top-k query to acquire an intuition of the solution space. Then, I study in detail the case of bichromatic reverse top-k query, and I propose two techniques for query processing, namely an efficient threshold-based algorithm and an algorithm based on materialized reverse top-k views. My experimental evaluation demonstrates the efficiency of the reverese Top-K Queries.

Keywords: Top-KQuery,ReverseTop-KQuery,WeightingVectors.

Introduction

Data Mining is the process of discovering interesting knowledge from large amounts of data stored either in databases, data warehouses, or other information repositories.
Information leads to poor and success, and sophisticated technologies such as computers, satellites, etc., tremendous amounts of information have been collected. Initially, with the advent of computers and means for mass digital storage, started collecting and storing all sorts of data. These massive collections of data stored on disparate structures very rapidly became overwhelming. This initial chaos has led to the creation of structured databases and database management systems. The efficient database management systems are applicable to any kind of information repository. Data mining is being put into use for relational databases, data warehouses, transactional databases, World Wide Web, spatial databases, multimedia databases, time-series databases and textual databases. All database systems must be able to respond to requests for information from the user—i.e. process queries. Obtaining the desired information from a database system in a predictable and reliable fashion is the scientific art of ***Query Processing***. Getting these results back in a timely manner deals with the technique of ***Query Optimization***. The basic steps in query processing involves 1) the scanning, parsing, and validating module produces an internal representation of the query. 2) The query optimizer module devises an execution plan which is the execution strategy to retrieve the result of the query from the database files.3) A query typically has many possible execution strategies differing in performance, and the process of choosing a ***reasonably efficient*** one is known as query optimization. (Query optimization is beyond this course) The code generator generates the code to execute the plan. The runtime database processor runs the generated code to produce the query result. The aim of query processing is to find information in one or more databases and deliver it to the user quickly and efficiently. Traditional techniques work well for databases with standard, single-site relational structures, but databases containing more complex and diverse types of data demand new query processing and optimization techniques. Most real-world data is not well structured. Today's databases typically contain much non-structured data such as text, images, video, and audio, often distributed across computer networks. In this complex milieu (typified by the World Wide Web), efficient and accurate query processing becomes quite challenging.



I Top k query processing : Top-k queries, produce results that are ordered on some computed score. A top- k query over defined subsystems returns the objects with the least aggregated scores. A top-k query returns the subsets of most relevant results instead of all results to minimize the cost metric that is associated with the retrieval of all results and maximize the quality of the result set, such that the user is not overwhelmed with irrelevant results.

ISSN 2348 – 7968

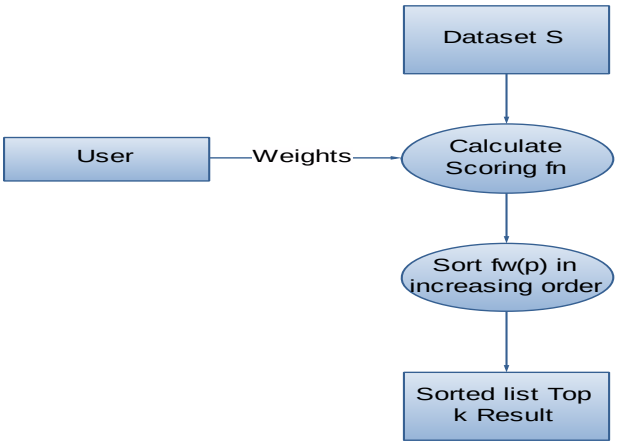


Fig. 1 Top k query processing

Input

Weights (i.e.) Preferences, top-k (e.g. k=100,150...) and Data set S.

Output

List of data points (i.e.) product sorted on the increasing order of the scoring function.

II Monochromatic reverse top k query: Reverse top-k query that covers this requirement: "Given a potential product, which are the user preferences that make this product belong to the top-k query result set?" Reverse top-k queries are essential for manufacturers to assess the impact of their products in the market based on the competition. In monochromatic reverse top-k query, there is no knowledge of user preferences and the aim is to estimate the impact of a potential product in the market when no user preferences are given, but the distribution of them is known. Monochromatic reverse top-k queries return partitions of the solution space and satisfy the query.

Algorithm 1. Monochromatic RTOPk Algorithm

1: Input: S, q
2: Output: mRTOP$_{k(q)}$
3: $W^1 \leftarrow \emptyset$, R←ø, RES←ø
4: for (p_i 2 S) do
5: if (q 6_ p_i and p_i 6_ q) then
6: w_i½←R
7: $W_0 \leftarrow W_{0+1}$
8: end if
9: end for
10: sort W_0 based on increasing value of w_i½
11: w_0 ½←w; $w_j w_{0jb1}$ ½←w;
12: R fp : p lies in H_{w_0} qÞg
13: k_w jRj //number of points in R
14: for (8w_i 2 W_0) do
15: if (k_w _ k) then
16: RES RES [fw_i; w_{ib1}Þg]
17: end if
18: if (p_{ib1} 2 R) then
19: k_w k_w _ 1
20: else
21: k_w k_w Þ 1
22: end if
23: end for
24: return RES

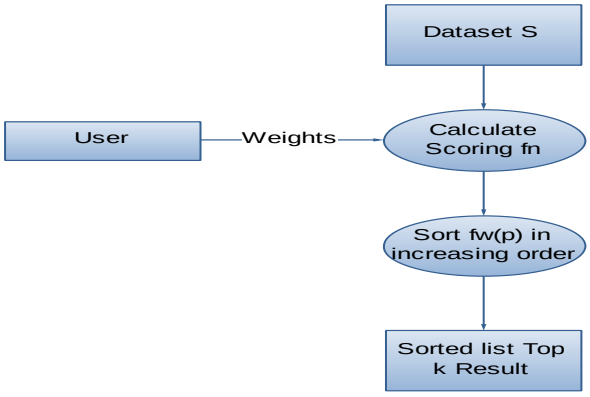


Fig .2 Monochromatic Reverse Top-k query

Input

Data set S, Query point q, Top-k (e.g. k=100,150…)

Output

Impact of the product in percentage.

Example of mRT OP_k ðqÞ for k ¼ 1. Consider for example the data set depicted in Fig. 3a. Since the only points that belong to the convex hull [7] are p, q, and r, I conclude that 1) only these points belong to the top-1 result set for any weighting vector, and 2) there exists at least one weighting vector w_i for which q 2 T OP_1 ðwi Þ, exactly one partition W_i 2 mRT OP_1 ðqÞ. The boundaries of the partition W_i are defined by the weighting vectors w_{pq}, w_{qr} for which the relative order between q and p or r changes. All weighting vectors w for which the following inequality holds are in the reverse top-1 result set of q: w_{qr} The result set of mRT OP_1 ðqÞ is a segment (partition) of the line w_1 Þ w_2 ¼ 1 in the 2-dimensional solution space defined by w_{pq} and w_{qr}.

Even though the result set mRT OP_k for k ¼ 1 contains at most one partition, for a reverse top-k query with k > 1, the result set may contain more than one partitions W_i . Consider, for example, the three data points and assume we are interested to compute the mRT OP_k ðqÞ for k ¼ 2. Query point q is in the top-2 result set for both weighting vectors w_1 and w_3 . However, when weighting vector w_2 is considered, with angle between w_1 and w_3 , it is obvious that q no longer belongs to the top-2. Thus, in this small example, the

ISSN 2348 – 7968

monochromatic reverse top-k query would return two partitions W_i.

Monochromatic RTOPk algorithm: Algorithm describes the monochromatic reverse top-k Data points that are dominated[3] by q are always ranked after q for any weighting vector w, while points that dominate q are ranked before q for any weighting vector w., p_5 is worse (ranked lower) than q, whereas p_6 is better (ranked higher) than q for any w. Points of the data set that are neither dominated by nor dominate q are ranked higher than q for some weighting vectors and lower than q for other vectors.Thus, the algorithm examines only such incomparable points $fp_i g$ to q (line 5), because they alter the rank of q. The boundaries of the partitions of mRT OP_k are defined by a subset of the weighting vectors w_i ¼ w_{pi} q.

III Bichromatic reverse top k query: A bichromatic reverse top-k query returns those preferences that rank a potential product highly. For the bichromatic version of the reverse top-k query, the result set contains a finite number of weighting vectors. It is important that a product is returned in the highest rank position for as many user preferences as possible. User history is being observed, and the product is advertised to the potential customers.

Threshold-Based Algorithm (RTA): RTA aims to reduce the number of top-k query evaluations, based on the observation that top-k queries defined by similar weighting vectors return similar result sets .Hence, RTA exploits already computed top-k result sets to avoid evaluating weighting vectors that cannot be in the reverse top-k result set. Therefore, in each repetition a threshold is set based on the previously computed top-k result set P.

Algorithm 2. RTA: RTOPk Threshold Algorithm
1: Input: S, W, q, k
2: Output: bRTOP$_k$qÞ
3: W_0 ;, buffer ;
4:w←q-1;
5: for (each w_i 2 W) do
6: if (f_{wi} qÞ←w) then
7: buffer TOP$_k$$w_i$Þ
8: if (f_{wi}qÞ ←$_{wi}$ bufferÞ) then
9: W_0 W_0 [fw_ig]
10: end if
11: end if
12: $_{wibi}$ bufferÞ
13: end for
14: return W_0

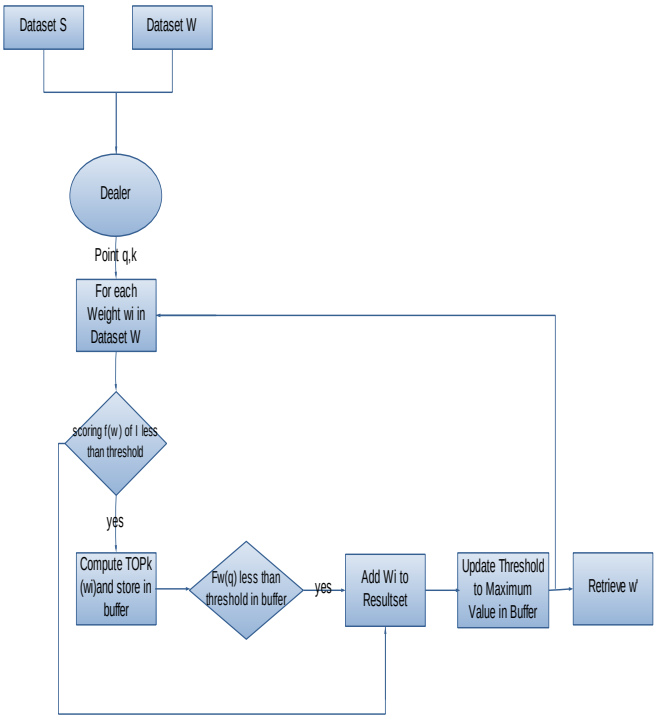


Fig .3 Bichromatic Reverse Top-k query

Input

Data set S, Data set W, query point q, top-k (e.g. K=100,150…)

Output

List of weights (i.e.) preferences that make the product to rank high.

Where, Data set S- set that contains products as data points.

Data set W- list of weights observed from the user.

IV Monochromatic RTOPk Query:

Properties of monochromatic RTOPk query.

In the following, I present some useful properties of RTOPk queries and discuss how the boundaries of the partitions W_i can be determined. I assume that there exist an ordering $w_1 ; \ldots ; w_{jW\,j}$ of the weighting vectors of jW j, such that a weighting vector w_i precedes another vector w_j, if w_i ½1 < w_j ½1 . Thus, the weighting vectors w_i is ordered based on increasing angle of w_i with the y-axis.

Lemma 1. Given two points p and q such that f_{w1} ðqÞ f_{w1} ðpÞ, there exists at most one weighting vector w such that f_{wi} ðqÞ < f_{wi} ðpÞ for w_i < w, and f_{wi} ðqÞ > f_{wi} ðpÞ for w_i > w.

Based on the above lemma, the relative order of p and q changes for weighting vectors with smaller and

ISSN 2348 – 7968

larger angles than w. If p had a lower rank than q for vectors with smaller angle than w, then p has a higher rank for vectors with larger angle than w. If there exists such a weighting vector, then I denote it as w_{pq} and refer to it as the weighting vector for which the relative order of q and p changes.

Lemma 2. Given two points p and q, if there exists a weighting vector w_{pq} for which the relative order of p and q changes, then it holds that $f_{w_{pq}}(q) = f_{w_{pq}}(p)$.

Equivalently, w_{pq} is the weighting vector that is perpendicular to the line segment pq, with $w_{pq}[1] = \frac{}{pq}$, where $pq = q[1] - p[1]$ is the slope of line segment pq. The above equation is derived by the property that $w_{pq} \perp pq$. The boundaries of any partition W_i are defined by weighting vectors w_{pq} for which the relative order of q and points $p \in S$ changes (additionally, the first and last partition are defined by the weighting vectors $(-0; 1)$ and $(-1; 0)$, respectively). Intuitively, as long as the relative order between any two points does not change, the top-k result is not affected and thus the rank of q remains the same.

Lemma 3. There exists at most one partition W_i, such that for all the weighting vectors $w \in W_i$ it holds that $q \in TOP_1(w)$.

Since the relative order between q and any data point p changes only once, if the rank of p becomes higher than q, then it cannot change again for the next vectors. Thus, q cannot be in the top-1 result set for any $w > w_{pq}$.

For a bichromatic reverse top-k query, two data sets S and W are given, where S contains the data points and W the different weighting vectors that represent user preferences. Then, the aim is to find all weighting vectors $w_i \in W$ such that the query point $q \in TOP_k(w_i)$.

A brute force (naive) approach is to process a top-k query for each $w_i \in W$ and test whether q belongs to $TOP_k(w_i)$.

Properties of reverse monochromatic RTOPk query.

Proof. I first show that VOP belongs to NP. Given an instance of VOP and a candidate solution, the verification algorithm checks that the ordering contains each vector exactly once, sums up the cost values, and checks whether the sum is at least c. This process can be done in polynomial time. To prove that VOP is NP-complete, I show that the Traveling Salesman Problem (TSP) is polynomial time reducible to the VOP (TSP $\leq_P$ VOP). Let $\langle G(V, E), w_e, c\rangle$ be an instance of TSP. I construct an instance of VOP as follows then form the set of vectors based on W.

Based on the above lemma, the relative order of p and q changes for weighting vectors with smaller and larger angles than w. If p had a lower rank than q for vectors with smaller angle than w, then p has a higher rank for vectors with larger angle than w.If there exists such a weighting vector, then I denote it as w_{pq} and refer to it as the weighting vector for which the relative order of q and p changes.

Example of mRT $OP_k(q)$ for k = 1:Consider for example the data set depicted in Fig. 1.Since the only points that belong to the convex hull [7] are p, q, and r, I conclude that, 1) only these points belong to the top-1 result set for any weighting vector, and 2) there exists at least one weighting vector w_i for which $q \in TOP_1(w_i)$, and based on Lemma 3 exactly one partition $W_i \in$ mRT $OP_1(q)$. The boundaries of the partition W_i are defined by the weighting vectors w_{pq}, w_{qr} for which the relative order between q and p or r changes. All weighting vectors w for which the following inequality holds are in the reverse top-1 result set of q: $w_{qr}[1] \cdot w[1] \leq w_{pq}[1] \cdot w[1]$. The result set of mRT $OP_1(q)$ is a segment (partition) of the line $w[1] + w[2] = 1$ in the 2-dimensional solution space defined by w_{pq} and w_{qr}, as shown in Fig. 3.

Even though the result set mRT OP_k for k = 1 contains at most one partition, for a reverse top-k query with k > 1, the result set may contain more than one partitions W_i. Consider, for example, the three data points in Fig. 3 and assume we are interested to compute the mRT $OP_k(q)$ for k = 2. Query point q is in the top-2 result set for both weighting vectors w_1 and w_3.However,when weighting vector w_2 is considered, with angle between w_1 and w_3, it is obvious that q no longer belongs to the top-2. Thus, in this small example, the monochromatic reverse top-k query would return two partitions W_i.

For example in Fig. 3, p_5 is worse (ranked lower) than q, whereas p_6 is better (ranked higher) than q for any w. Points of the data set that are neither dominated by nor dominate q are ranked higher than q for some weighting vectors and lower than q for other vectors.Thus,the algorithm examines only such incomparable points $\{p_i\}$ to q (line 5), because they alter the rank of q.

Higher Dimensional Data:

In higher dimensions (d > 2), all valid weighting vectors of the RTOPk query form a (d − 1)-dimensional hyperplane that contains the points $w_i[j] = 0 \ \forall j = i$ and $w_i[j] = 1$ for $j = i$ and $1 \leq i \leq d$. A monochromatic RTOPk query returns the partitions W_i of the hyperplane, for which the query point q is in the $TOP_k(w)$, $\forall w \in W_i$.In the following,I provide an example for finding the partitions for d > 2.

Let us consider a 3-dimensional data set S_1 containing only three points $A = (1; 0; 0)$, $B = (0; 1; 0)$, and $C = (0; 0; 1)$ I denote as W_A, W_B, and W_C are the partitions for which A, B, and C are the top-1 data point, respectively. Similar tothe 2-dimensional case,the borders of the partitions are defined by the weighting vectors for which

ISSN 2348 – 7968

the relative order between two points changes. To define the borders of the partitions W_A and W_B, I need to examine the locus of weights w^0 for which $f_{w^0}(A) = f_{w^0}(B)$. Furthermore, I seek only the weighting vectors for which $f_{w^0}(A) = f_{w^0}(B) < f_{w^0}(C)$, since otherwise C is the top-1 point. Thus, if I take also into consideration that $f_{w^0}(A) < f_{w^0}(C)$, then an additional constraint is formed, namely $c > 1$. Therefore, the border between the partitions W_A and W_B is the line segment defined by the points ½1=3; 1=3; 1=3 and ½1=2; 1=2; 0 .By repeating the same procedure for the other pairs of points, the result partitioning is depicted in Fig. 4

Notice that there exists a single weighting vector w ¼ ½1=3; 1=3; 1=3 for which all the three data points have the same score for the data set S_1.

Fig. 3 depicts the partitions of the solution space for another data set S_2 containing the points A ¼ ½1=2; 0; 1=2 , B ¼ ½1=2; 1=2; 0 , and C ¼ ½0; 1=2; 1=2 . For data set S_2 , in order to detect the border between W_A and W_B.on the k data objects in the buffer. Top-k queries defined by similar weighting vectors return similar result sets[1]. Thus, if the buffered result set was obtained by a similar weighting vector w^0 with the currently processed w, then the probability that the threshold can discard w is high. As a result, the order in which the weighting vectors are examined influences the performance of RTA, and it is beneficial to access similar weighting vectors in consecutive steps. Consequently, the weighting vectors W are sorted based on their pairwise similarity. Given a jW **j** jW j similarity matrix M with nonnegative values M½i; **j** that represent the similarity between w_i and w_j, and ordering of the weighting vectors e ¼ w_{i1} ; . . . ; $w_{jW\,j}$order changes.

Sorted Access to Weighting Vectors:

In each repetition, RTA sets a threshold exploiting previously computed top-k result sets, in order to discard weighting vectors that cannot be in the query result set. The effectiveness of the threshold depends obtaining the partition boundaries in higher dimensions is a complicated process that goes far beyond the task of identifying the weighting vectors for which the relative defined as the line segment defined by the points ½1=3; 1=3; 1=3 and ½1; 0; 0 . This discussion shows that the order between the partitions W_A and W_B is Later, in [11], the dominant graph is proposed as a structure that captures dominance relationships between points. Another family of algorithms focuses on computing the top-k queries over multiple sources, where each source provides a ranking of a subset of attributes only. Fagin et al. [12] introduce TA and NRA algorithms of preprocessing. Efficient maintenance of materialized views for top-k queries is discussed in [10]. The robust index [2] is a sequential indexing approach that improves the performance of Onion [7] and Prefer [1]. The main idea is that a tuple should be placed at the deepest layer possible, to reduce the probability of accessing it at query processing time, without compromising the correctness of the result.

3

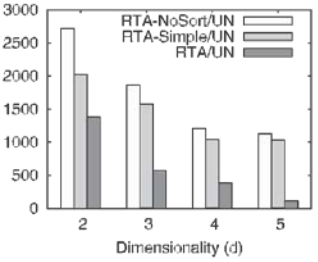


Fig. 4. Effect of sorting W .

3

V RELATED WORK

As reverse top-k queries are inherently related to top-k query processing, I summarize some representative work here. One family of algorithms are those based on preprocessing techniques. Onion [7] precomputes and stores the convex hulls of data points in layers. Then, the evaluation of a linear top-k query is accomplished by processing the layers inwards, starting from the outmost hull. Prefer [1] uses materialized views of top-k result sets, according to arbitrary scoring functions. Onion and Prefer are mostly appropriate for static data, due to the high cost.

VI CONCLUSIONS

In this paper, I introduce the reverse top-k query which retrieves all weighting vectors for which the query point belongs to the top-k result set. The proposed query type is important for market analysis and for estimating the impact of a product based on the user preferences and the competitors products.

VII REFERENCES

[1] V. Hristidis, N.Koudas, and Y. Papakonstantinou, “Prefer: A System for the Efficient Execution of Multi-Parametric Ranked Queries,” Proc. ACM SIGMOD Int’l Conf. Management of Data, pp. 259-270, 2001.

[2] D. Xin, C. Chen, and J.Han, “Towards Robust Indexing for Ranked Queries,” Proc. 32nd Int’l Conf.

ISSN 2348 – 7968

Very Large Data Bases (VLDB '06), pp. 235-246, 2006.
[3] A.Vlachou, C.Doulkeridis, Y. Kotidis, and K. Nørvå g, "Reverse Top-k Queries," Proc. IEEE 26th Int'l Conf. Data Eng. (ICDE), pp. 365-376, 2010.
[4] F.Korn and S. Muthukrishnan, "Influence Sets Based on Reverse NearestNeighbor Queries," Proc. ACM SIGMOD Int'lConf. Management of Data, pp. 201-212, 2000.
[5] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries," Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB '07), pp. 291-302, 2007.
[6] X.Lian and L. Chen, "Monochromatic and Bichromatic Reverse Skyline Search over Uncertain Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 213-226, 2008.
[7] Y.-C. Chang, L.D. Bergman, V. Castelli, C.-S. Li, M.-L. Lo, and J.R. Smith, "The Onion Technique: Indexing for Linear Optimization Queries," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 391-402, 2000.
[8] D.J. Rosenkrantz, R.E. Stearns, and P.M. Lewis II, "An Analysis of Several Heuristics for the Traveling Salesman Problem,"SIAM J. Computing, vol. 6, no.3,pp.563-581, 1977.
[9] S. Bö rzsö nyi, D. Kossmann, and K. Stocker, "The Skyline Operator," Proc. 17th Int'l Conf. Data Eng. (ICDE), pp. 421-430, 2001.

[10] K. Yi, H. Yu, J. Yang, G. Xia, and Y. Chen, "Efficient Maintenance of Materialized Top-k Views," Proc. 19th Int'l Conf. Data Eng. (ICDE), pp. 189-200, 2003.
[11] L.Zou and L. Chen, "Dominant Graph: An Efficient Indexing Structure to Answer Top-k Queries," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08), pp. 536-545, 2008.
[12] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware," Proc. 20th ACM SIGMOD-SIGACT- SIGART Symp. Principles of Database Systems (PODS), pp. 102-113, 2001.