Recommendation System for Answering Missing Tuples

Pekhale Kanchan Anil[1] Dr. K. V. Metre[2]
[1]P.G Student
[1,2]Department of Computer Engineering
[1,2]MET's Institute of Engineering, Adgoan, Nashik, Maharashtra, India

Abstract— In the recent years, the quality and the usability of database systems have received more attention. The performance of database systems has gained more improvement in the past decades so, they are more and more difficult to use. The why-not questions are the users need to know why her expected tuples are not shown up in the query results i.e. the features of explaining missing tuples in database queries. Database systems is having the capability that enables users to seek clarifications on expected query results as well as the absence of expected tuples (i.e. missing tuples). It would be very helpful to users if they referred why-not questions to seek clarifications on expected tuples in query results. The problem of answering why-not questions on top-k queries, there are two algorithms to answer such why-not questions efficiently. In proposed system user can give a SQL query as input. The system includes Selection, Projection, Join, Union and Aggregation (SPJUA) query operations. After execution of query user expected answer is not in query result, then user can ask a why-not question for expected result. In a Why-Not Top-K Question algorithm the weighting value is calculated for an expected result. The Weighting value is used for the highest preference of user expected tuples which limits its practicability. In a Why-Not Top-K Dominating Question algorithm there is no need to specify the weighting value. These algorithms are able to return high quality explanations efficiently. Then Refined Query is generated for user expected answer. The ConQueR method generates the refined query by modifying the predicates value which is provided by user in the query. Many users love to pose those kinds of queries when they are making multi-criteria decisions and user need approximate information from the huge Database.

Key words: Top-k Question, Dominating Question, Skyline Queries, ConQueR Method

I. INTRODUCTION

In recent years, more work is done on database systems to improve the usability. A search engine can locate information sources based on keywords which help a user to answer a question. Explaining the features of missing tuples in database queries is called why-not questions. In recent years it has received more attention. A why- not question is occurred when a user need to know why her expected tuples are not appeared in the result. Why-not questions are helpful to users to seek clarification on missing tuples from theresult of query. Recently, a specific work is done on answering why-not questions on traditional SQL queries. However, on preference queries like top-k queries none of those can answer why-not questions yet. When users are making multi-criteria decisions then answering why-not questions on top-k queries is useful. However, they might feel lost when their expected answers are absentin the query result and they mightneed to know why: Is it since I have set k too small?, Or I have set my weightings badly?, Or because of both? For addressing the why-not questions on top-k queries there are two algorithms. Namely, a why-not top-k questions and why-not dominating questions. Top-k dominating queries are variable from top-k query that users might suggest why-not questions on. The highest numbers of objects in dataset are given by top-k dominating queries which are dominated by k data objects. Users are did not specify the set of weightings in a top-k dominating query for ranking the objects based on the number of objects that could dominate (e.g., if object a dominates 5 objects while object b dominates 2 objects, then a ranks higher than b). For example, the agent of Ram, a hot NBA player this year, may pose a top-50 dominating query about the best guards in NBA history. When Ram is not in the result, his agent may need to know the reason: Is that I have set my k too small? Finding the best clarification is expensive for both the algorithms. In the why-not top-k question 'W' as an input provided by the user, which limits its practicability while in why-not top-k dominating question users need not to provide W as input. In the scenario, missing tuples are already known to the user and explained why those objects are missing.

II. LITERATURE SURVEY

SQL Query Recommendation framework aim is the non-expert users of scientific databases are tracking their querying behavior and generating the personalized query recommendations. The framework is supported by two recommendation engines and the recommendation technique. In the first approach, database related interesting partsconsiderfor data analysis task by locating those database parts that accessed by similar users in the past. In the second algorithm similar queries are identified structurally which are posted by the current user. In a recommendation set of SQL queries result of both the approaches are provided to the user to modify or directly post to the database. The drawback of this is, it mainly focus on the improving the Usability and the Quality of the database systems [2][10].

Now a day there is a growing population of non-expert database users which are need to perform complex operations on their dataset, but it is difficult to writing SQL queries. When user types a query, SnipSuggest recommends add possible clauses in the query using relevant snippets which are collected from past queries. As a user writing a query, they can ask SnipSuggest for recommendations for specific clause of her query. In response, SnipSuggest recommends small SQL snippets [3].

In various scenarios,explaining missing answers of queries is useful, including query understanding and debugging. When queries are used to define multiple views, one may ask why, employee information is missing from both the employee register and the payroll views. Artemis algorithms are able to generate explanations for missing tuples over a set of queries which include selection, projection, join, union, and aggregation and grouping (SPJUA). For explanations, it encodes the problem into a set of constraints and it combines existing data with new data [4].

Database systems are having the capability to seek clarifications on anticipated query results. For explaining why-not question on query results there are two models. In the first approach modifying some objects in the dataset so, the original result and the specified missing tuples are included in the query result. The second approach, identifying the manipulation operations in the query plan for missing tuple explanation and that are responsible for excluding the missing tuples [5][9].

The k data objects are return by top-k dominating query which are dominate the highest number of objects in a dataset. The significance and practicability of the query is identified and some of its potential extensions are define. An evaluation technique for top-k dominating queries depends on skyline computation. The number of results is control by user it is the advantage of top-k query [6][14].

In many applications top-k queries are well known. Given a query function Q and a record set R, a top-k preference query returns k records from R, whose values of function Q on their attributes are the highest. For improving the query efficiency the Dominant Graph (DG) is used [7][12][17].

In the database community preference query computation has received more attention. To evoke a common approach, a user's preference which give an arrangement of objects to user, and based on his/her decision of the objects; this are strongly introduce the correct weights. For this order-based representative skyline is used and selects representatives which are represented in the order [8][11][13][15][16].

For avoiding the above drawbacks answering the why-not questions on top-k queries are used, there are two algorithms which are answer why-not questions efficiently. First is a why-not top-k question and second is why-not dominating question. The why-not questions are the users need to know why expected answer doesn't present in the results of query i.e explaining the features of missing objects in dataset queries [1].

III. PROPOSED SYSTEM

The why-not question is occurred when a user need to know why expected answer does not present in the result. Recently, research was done on answering why-not questions on traditional SQL queries. However, on preference queries like top-k queries none of those can answer why-not questions yet. The query autocompletion concept is used to help a user for formulating the SQL query, but it is complex process to user. The why-not questions are helpful to users to seek clarification on missingtuples from the result. For answering the why-not questions on top-k queries problem, there are two algorithms to answer such kind questions efficiently. First is a why-not top-k question and second is a why-not top-k dominating question. User can give an input query to the Why-not top-k question and why-not top-k dominating question algorithms. If user can specify the weighting value, the input query is given to why-not top-k question otherwise the query is given to why-not top-k dominating question. In the scenario, missing tuples are already known to the user and explained why those objects are missing.

IV. SYSTEM ARCHITECTURE

The two algorithms are used which takes user query as an input. In the first algorithm user specifies the weighting value for the query and also uses the scoring function as any monotonic function. The second algorithm is same as Why-Not Top-K question and there is no need to provide weighting value. After the executions of this algorithms result is returned to user by using Query Refinement Approach whichwill provide a missing result to user.

As shown in Fig. 1, the user can give an input query to the algorithms. If user can specify the weighting value, the input query is given to why-not top-k question otherwise the query is given to why-not top-k dominating question.

In the why-not top-k question users can give W as input, which limits its feasibility and in why-not top-k dominating question users need not to provide W as input. A query refinement technique is used for generation of result. Revise the user's original query by the query refinement technique for the missing tuples are included in the result. This technique defines a best refined query should be (a) similar – there are minimum edit operations than original query and (b) precise - in the result some extra tuples, including the missing objects and original result.

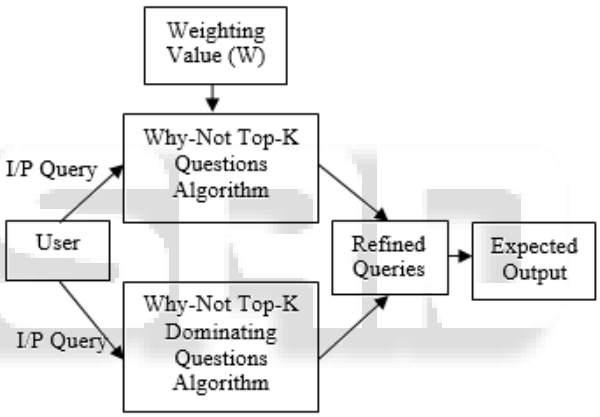

Fig. 1: System Architecture

A. Why-Not Top-K Question:

Suppose only one missing object which is represented by $\vec{m}$. Firstly, top-k query q'_0 based on the weighting vector $\vec{w}_0$ in the users original query q_0 is execute, using any evaluation algorithm. When $\vec{m}$ returned into the results with a ranking r_0 then stop the execution. If $\vec{m}$ does not present in the result, then $\vec{m}$ does not present in the dataset is convey to user and terminate the process.

In the dataset the $\vec{m}$ is present, then sample list of weighting vectors randomly $N = [\vec{w}_1, \vec{w}_2, \dots, \vec{w}_n]$ from the weighting space. For all weighting vector value $\vec{w}_i \epsilon N$, then reduce top-k query q'_i using $\vec{w}_i$ as the weighting. A top-k query algorithm executed each query q'_i, which reports each top ranking object, since the missing object $\vec{m}$ included in the result set with ranking r_i. As the answer the refined query $q'_i(r_i, \vec{w'}_i)$ with the minimum penalty (P) is returned to user. The least Penalty(P) can be calculated by using following formula:

$$P(k', \vec{w}') = \lambda_k \frac{\Delta k}{(r_0 - k_0)} + \frac{\Delta w}{\sqrt{1 + \sum w_0[i]^2}} \qquad (1)$$

In the above equation Δk and Δw are used calculating the quality for the refined query. Where $\Delta k = \max(0, k' - k_0)$ and $\Delta w = \|\vec{w}' - \vec{w}_0\|$. The k' is a refined query value is smaller than the original k_0 value. r_0 is the ranking of the missing tuple $\vec{m}$. A basic penalty model set the penalties λ_k and λ_w to Δk and Δw for capturing the user tolerance, where $\lambda_k + \lambda_w = 1$.

Algorithm 1: Why-NotTop-K Question

Input: Top-K Query $q_0(k_0, \vec{w}_0)$, Dataset D, missing object $\vec{m}$, Penalty setting λ_k , λ_w

Output: Result set X of top-k Query, Refined Query $q'(k', \vec{w'})$

Phase 1:

1) $(X_0, r_0) \leftarrow TOPK(q_0'(\vec{w}_0), Since\_see\_\vec{m})$;
2) if r_0=NULL then
3) return "$\vec{m}$ not present in Database"
4) end if
5) calculate the points a which can dominate $\vec{m}$
6) Determine sampling n;
7) sample n weighting from the weighting space and n is add into N
8) Sort N with respect to Δw_i values;

Phase 2:

9) $X \leftarrow (X_0, \vec{w}_0)$; // Store the result
10) $P_{min} \leftarrow P(r_0, \vec{w}_0)$;
11) $\Delta k_l \leftarrow \max(a + 1 - k_0, 0)$; // Calculating the lower bound ranking of missing object m
12) $\Delta w^f = \left(P_{min} - \lambda_k \frac{\Delta k_l}{r_0 - k_0}\right) \frac{\sqrt{1+\sum w_0[i]^2}}{\lambda_w}$; //Early termination
13) $for\ \vec{w}_i \in N\ do$
14) if $\Delta w_i > \Delta w_f$ then
15) break; // early termination
16) end if
17) $\Delta k^t \leftarrow \left\lfloor \left(P_{min} - \lambda_w \frac{\Delta w_i}{\sqrt{1+\sum w_0[i]^2}}\right) \frac{r_0 - k_0}{\lambda_k} \right\rfloor$;
18) $r^t \leftarrow \Delta k^t + k_0$;
19) $(X_i, r_i) \leftarrow TOPK(q_0'(\vec{w}_i), Since\_see\_\vec{m}\ or\ Since\_rank\_r^t)$;
20) $p_i \leftarrow P(r_i, \vec{w}_i)$;
21) Store result in X;
22) if $p_i < P_{min}$ then
23) $P_{min} \leftarrow p_i$;
24) $\Delta w^f = \left(P_{min} - \lambda_k \frac{\Delta k_l}{r_0 - k_0}\right) \frac{\sqrt{1+\sum w_0[i]^2}}{\lambda_w}$; // Early Termination
25) end if
26) end for

Phase 3:

27) best refined query returned to user and result with minimum penalty (P)

Algorithm:

Phase 1: Execute a top-k query since $\vec{m}$ occurred in the result, with rank r_0, using weighting $\vec{w}_0$. That operation is $(X_0, r_0) \leftarrow TOPK(q_0'(\vec{w}_0), Since\_See\_\vec{m})$. A best refined query is Y% if its penalty is minimum than (1-Y)% refined queries in the answer space. Getting the probability of refined query is at least one has more than a certain threshold p.

$$1 - (1 - Y\%)^n \geq p \quad (2)$$

Phase 2: Top-k query is executing by using the weighting vector and refined query is generated with the minimum penalty. Consider the threshold value r^t for stopping top-k operation.

Phase 3: Refined query answer is returned as a why-not answer to user with the minimum penalty.

B. Why-Not Top-K Dominating Question:

Assume only one missing object $\vec{m}$. Firstly, a top-k dominating query q_0' is execute by using a top-k dominating query algorithm. When $\vec{m}$ is appear in the result set then stop the execution and apply the ranking r_0. In the query result $\vec{m}$ is not occurred, then convey to user that $\vec{m}$ is not present in the dataset and terminates the process. In the database $\vec{m}$ is present, then manage a list of data value samples $N = [\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n]$. For all data value sample $\vec{x}_i \epsilon$ N, modify $\vec{m}$ values to be $\vec{x}_i$ and then execute top-k dominating query since $\vec{m}$ appear into the result with ranking r_i. After top-k dominating executions, the n+1 "modified values and refined queries"

pairs:$\langle q_0'(r_0), \vec{m} = \vec{m}\rangle, \langle q_1'(r_1), \vec{m} = \vec{x}_1\rangle, \dots, \langle q_n'(r_n), \vec{m} = \vec{x}_n\rangle$. As the answer, the pair and the minimum penalty is returned to the user. In this scoring function is used, which scores an object $\vec{p}$ by the number of points that it can dominate. The top-k objects have the highest scores is the result of query. User takes a result but their expected answer is not present then it pose a why-not question on q_0 with a missing objects $M = \{\vec{m}_1, \dots, \vec{m}_j\}$. A new value and a new value $\vec{m}_i'$ is required for the dominating question for all object $\vec{m}_i \in M$. The quality of the why-not answer is calculated by Δk and Δz, where $\Delta k = \max(0, k' - k_0)$ and $\Delta z = \sum_{i=1}^{j} \|\vec{m}_i' - \vec{m}_i\|_2$ for largest possible value.

Algorithm 2: Why-Not Top-K Dominating Question

Input: Top-K Query $q_0(k_0)$, Dataset D, missing object $\vec{m}$, Penalty setting λ_k , λ_d

Output: Refined Query $q'(k')$, Result List l for dominating Query, r_0 for ranking of missing objects

Phase 1:

1) $(l, r_0) \leftarrow DOMINATING(q_0'(\vec{w}_0), Since\_see\_\vec{m})$;
2) if r_0=NULL then
3) return "$\vec{m}$ not present in Database"
4) end if
5) calculate lower bound point $\vec{l}$
6) sample n weighting from the weighting space and n is add into N

Phase 2:

7) $P_{min} \leftarrow P(r_0, \vec{m})$;
8) $\Delta z^f = \left(P_{min} * \frac{\|\vec{l} - \vec{m}\|_2}{\lambda_z}\right)$; // calculated Δz intercept
9) Store result with ranking value
10) $for\ \vec{x}_i \in N\ do$
11) if $\Delta z_i > \Delta z^f$ then
12) continue; // Skip Compute Rank operation
13) end if
14) $\Delta k^t \leftarrow \left\lfloor \left(P_{min} - \lambda_z \frac{\Delta z_i}{\|\vec{l} - \vec{m}\|_2}\right) \frac{r_0 - k_0}{\lambda_k} \right\rfloor$;
15) $r^t \leftarrow \Delta k^t + k_0$;
16) if there are objects which dominates $\vec{x}_i$ then
17) Continue ; // use stored results
18) end if
19) $r_i \leftarrow Compute\_Rank(\vec{m}, \vec{x}_i)$;
20) Store result;
21) if $p_i < P_{min}$ then

22) $P_{min} \leftarrow p_i$;

23) $\Delta z^f = \left(P_{min} * \frac{\|\vec{l}-\vec{m}\|_2}{\lambda_z}\right)$; // Early Termination

24) end if

25) end for

Phase 3:

26) $k' and\ \vec{m}$ is returned with minimum penalty (P)

Algorithm:

Phase 1: Executes a top-k dominating query algorithm to represent the list *l* of objects with their scores in ranking since the missing objects is comes into the result. Denote this operation as $(l, r_0) \leftarrow DOMINATING(q_0'(\vec{w}_0), Since\_see\_\vec{m})$.

Phase 2: The missing object values are modified for determining the ranking. From the line no. 14 of the algorithm the threshold value is calculated and determines the ranking of missing objects.

Phase 3: From the refined queries and modified values are returned to user with least penalty.

C. *Skyline Queries:*

If dissimilarity and imprecision metrics are low then refined query is considered to be good [5]. The set of skyline refined queries defined as follows: Given two refined queries x and y, the x dominates y if (1) the dissimilarity and imprecision metrics of x are small than y and (2) x value is at least one of metrics lower than y. The x is not dominated by the other refined query then x is called a skyline query. To explain the question the skyline queries are computed.

D. *Conquer Method:*

ConQueR, is a Constraint-based Query Refinement, which is used by why-not questions explanation for generating more than one refined queries automatically. ConQueR is a similarity driven technique which generates refined queries with low dissimilarity values. Suppose, why-not question takes (S,C) is the input for a query O on dataset D. ConQueR technique modifies selection predicate(s) in O to derive O' by minimizing the imprecision metric. When refined queries are present, skyline queries are generated by the ConQueR method that contains the O as a query schema. If refined queries are not present then ConQueR method finds refined queries which have a different query schema.

In ConQueR method searching for refined queries, the sequence of query schemas $OS_1,\ldots, OS_k$ is iterated effectively. The query schema of the input query O is the OS_1, and if there are no refined queries with schema $OS_1,\ldots,OS_i$ then consider OS_{i+1} schema. The skyline queries are generated by the ConQueR method with schema OS_k and also generate the why-not question explanation.

V. Experimental Results

A. *Experimental Setup:*

Dataset: The NBA database is the knowledge dataset which contains the game statistics of all NBA players.

IMDB (Internet Movie Database) in which all the records of movies are present.

Setup: JDK environment is used for implementation. The experiment is done on Windows with Intel core 2 dual processor, speed 2.20 GHz and RAM 1GB.

Result Analysis

The Fig. 2 shows the Top-k and Missing Value graph. In this graph, the system gives a top-k result efficiently and effectively to the user. The Top-K results are actual results exclude the missing result.

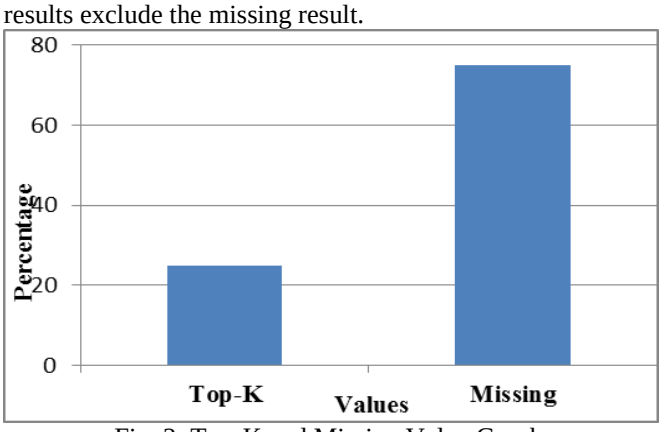

Fig. 2: Top-K and Missing Value Graph

The Fig. 3 shows the Top Scoring graph of the system. In this the top score is calculated from the result which is returned to user. The system can take a threshold value for calculating the score. In the graph from the dataset the values of columns are taken PTS, BLK, FG and FT for calculating the Top Scoring value. This shows the accuracy, effectiveness and efficiency of the system.

Sr. No.	Rank	Player	PTS	BLK	FG	FT	Scoring
19	1	LA Clippers	112.0	3.0	44.0	12.0	42.75
17	2	Golden State	99.0	3.0	40.0	10.0	38.0
7	3	Atlanta	92.0	2.0	37.0	16.0	36.75
5	4	Orlando	93.0	1.0	40.0	11.0	36.25
29	5	Atlanta	90.0	3.0	38.0	7.0	34.5

Table 1: Top Scoring Value

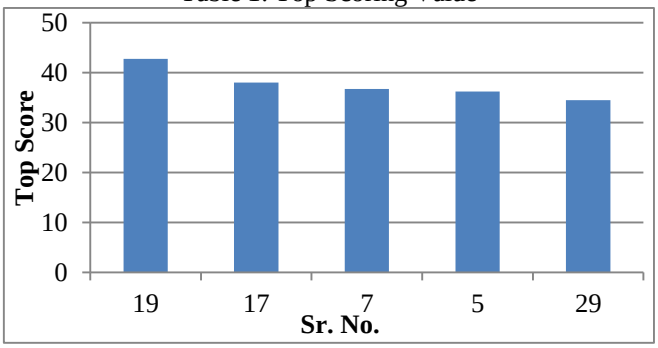

Fig. 3: Top Scoring Value Graph

VI. Conclusion

Users do not understand why her expected answers are missing in the query result. For this, answering why-not questions on two types of top-k queries, the basic top-k query where users need to specify the set of weightings, and the top-k dominating query where users do not need to specify the set of weightings because the ranking function ranks an object higher, if it can dominate more objects. A refined query with approximately minimal changes to the k value and their

weightings is returned to the user. It can be used for multi-criteria decision as well as for approximate information from database. The work will be performed on numeric data as well as non-numeric data. Both the algorithms will able to return high quality explanations efficiently and effectively. The accurate answers are return to user with the refined query.

REFERENCES

[1] Zhian He, Eric Lo, "Answering Why-Not Questions on Top-K Queries", IEEE, vol.26, no.6, June 2014.
[2] J. Akbarnejad et al., "SQL query recommendations", PVLDB, vol. 3, no. 2, pp. 15971600, 2010.
[3] M. Balazinska, N. Khoussainova, Y. C. Kwon, and D. Suciu, "SnipSuggest: Context-aware autocompletion for SQL", PVLDB, vol. 4, no. 1, pp. 2233, 2010.
[4] M. Herschel and M. A. Hernndez, "Explaining missing answers to SPJUA queries", PVLDB, vol. 3, no. 12, pp. 185196, 2010.
[5] Q. T. Tran and C.-Y.Chan, "How to ConQueR why-not questions", in Proc. ACM SIG-MOD, New York, NY, USA, 2010, pp. 1526.
[6] M. L. Yiu and N. Mamoulis, "Efficient processing of top-k dominating queries on multi-dimensional data", in Proc. VLDB, Vienna, Austria, 2007, pp. 541552.
[7] L. Zou and L. Chen, "Dominant graph: An efficient indexing structure to answer top-k queries," in Proc. ICDE, Washington, DC, USA, 2008, pp. 536–545.
[8] F. Zhao, K.-L. Tan, G. Das, and A. K. H. Tung, "Call to order: A hierarchical browsing approach to eliciting users' preference," in Proc. ACM SIGMOD, New York, NY, USA, 2010, pp. 27–38.
[9] Kaushik Chakrabarti and Sharad Mehrotra, "Evaluating Refined Queries in Top-k Retrieval Systems", in IEEE, VOL. 16, NO. 2, FEBRUARY 2004.
[10] Thanh Tran and Lei Zhang, "Keyword Query Routing", in IEEE, VOL. 26, NO. 2, FEBRUARY 2014.
[11] Bagus Jati Santoso and Ge-Ming Chiu, "Close Dominance Graph: An Efficient Framework for Answering Continuous Top-k Dominating Queries", in IEEE, VOL. 26, NO. 8, AUGUST 2014.
[12] Xixian Han, Jianzhong Li and Hong Gao, "Efficient Top-k Retrieval on Massive Data", in IEEE, VOL. 27, NO. 10, OCTOBER 2015.
[13] Saiful Islam, Chengfei Liu and Jianxin Li, "Efficient Answering of Why-Not Questions in Similar Graph Matching", in IEEE, VOL. 27, NO. 10, OCTOBER 2015.
[14] Albert Yu, Pankaj K. Agarwal and Jun Yang, "Top-k Preferences in High Dimensions", in DOI IEEE Transactions on Knowledge and Data Engineering, 2015.
[15] Liming Zhan, Ying Zhang, Wenjie Zhang and Xuemin Lin, "Finding Top k Most Influential Spatial Facilities over Uncertain Objects", in DOI IEEE Transactions on Knowledge and Data Engineering, 2015.
[16] Xiaoye Miao, Yunjun Gao, et.al, "Top-k Dominating Queries on Incomplete Data", in DOI IEEE Transactions on Knowledge and Data Engineering, 2015.
[17] Eleonora Ciceri, Piero Fraternali, Davide Martinenghi and Marco Tagliasacchi, "Crowdsourcing for Top-K Query Processing over Uncertain Data", in DOI IEEE Transactions on Knowledge and Data Engineering, 2015.

