**LETTER**

Top $(k_1, k_2)$ Query in Uncertain Datasets

**Fei LIU**,† Member, **Jiarun LIN**,† Student Member, and **Yan JIA**,† Nonmember

**SUMMARY** In this letter, we propose a novel kind of uncertain query, top $(k_1, k_2)$ query. The x-tuple model and the possible world semantics are used to describe data objects in uncertain datasets. The top $(k_1, k_2)$ query is going to find $k_2$ x-tuples with largest probabilities to be the result of top $k_1$ query in a possible world. Firstly, we design a basic algorithm for top $(k_1, k_2)$ query based on dynamic programming. And then some pruning strategies are designed to improve its efficiency. An improved initialization method is proposed for further acceleration. Experiments in real and synthetic datasets prove the performance of our methods.

**key words:** uncertain query, top $k$, x-tuple, possible world

1. Introduction

Top $k$ query is widely used in data mining, abnormal detection and many other research fields. At the same time, uncertainty is inherent in many datasets due to various factors like noise [1], privacy protection strategy [2], incompleteness of data and delay or loss in data transfer [3]. In this paper, we connect top $k$ query and uncertain data model, and propose a novel top $(k_1, k_2)$ query in uncertain datasets.

In uncertain space, a data object may be represented by many instances with distinct probabilities. We use the x-tuple model [4] and the possible world semantics [5] to describe uncertain data. An uncertain data object (abbreviated as object) is considered as an x-tuple, containing several tuples. Each tuple belonging to an x-tuple is a possible instance of the object. Each instance is with an appearance probability. Instances belonging to the same object are exclusive. Instances belonging to the different objects are independent. Sum of these instances’ probabilities is no more than 1. For example in Table 1, $t_{11}$ and $t_{12}$ are two tuples of x-tuple $T_1$. $t_{11}$ and $t_{12}$ cannot appear at the same time. $t_{21}$ with probability $p_{21}$ is the unique tuple of x-tuple $T_2$. If $p_{11} + p_{12} < 1$, $T_1$ would not appear with probability $1 - p_{11} - p_{12}$.

Based on the x-tuple model, a possible world is a set of tuples from different x-tuples. There would be no more than one tuple from the same x-tuple appearing in a possible world. Each possible world is with a probability. Table 2 lists 6 possible worlds produced from x-tuples in Table 1. For example, $\emptyset$ is a possible world, where no x-tuple appears. $\{t_{11}\}$ is a possible world with probability $p_{11}(1 - p_{21})$.

| Table 1 | x-tuple model. |
|---------|-----------------|
|         | x-tuple | probability |
| $T_1$   | $t_{11}$  | $p_{11}$    |
|         | $t_{12}$  | $p_{12}$    |
| $T_2$   | $t_{21}$  | $p_{21}$    |

| Table 2 | Possible worlds. |
|---------|------------------|
|         | possible world   | probability   |
|         | $\emptyset$     | $(1 - p_{11} - p_{12})(1 - p_{21})$ |
| $\{t_{11}\}$ | $p_{11}(1 - p_{21})$ |
| $\{t_{12}\}$ | $p_{12}(1 - p_{21})$ |
| $\{t_{21}\}$ | $(1 - p_{11} - p_{12})p_{21}$ |
| $\{t_{11}, t_{12}\}$ | $p_{11}p_{21}$ |
| $\{t_{12}, t_{21}\}$ | $p_{12}p_{21}$ |

In this letter, we focus on top $k$ x-tuples detection using possible world semantics. Several queries in uncertain space have been proposed, such as U-top$k$, U-kRank, PT-$k$, Global-top$k$, Expected rank and so on [6]. In these queries, every tuple has a score and tuples with highest highrank probabilities would be returned. G. Cormode et al. [7] analysed these models and indicated that most of them do not satisfy five key properties of uncertain query: Exact-k, Containment, Unique ranking, Value invariance and Stability. Also the Expected rank query satisfy all these properties [6], the expected value always leads to information loss. The Global-top$k$ [8] query returns $k$ highest-ranked tuples according to their probability of being in the top-$k$ answers in possible worlds. It satisfies all properties except ‘Containment’. Besides that, some of them would cause high time cost [6]. In this letter*, we propose a new top $(k_1, k_2)$ query for x-tuples query similar with Global-top$k$ query in uncertain datasets. It satisfies all these key properties. The remainder of the letter is organized as follows. In Sect. 2, we define the top $(k_1, k_2)$ query. A basic algorithm is described in Sect. 1. Some pruning strategies are designed in Sect. 4. An improved initialization strategy is proposed in Sect. 5. We experimentally evaluate our research in Sect. 6. Finally, we conclude our work in Sect. 7.

2. Top $(k_1, k_2)$ Uncertain Query

We use the x-tuple model and the possible world semantic to

---

*This research is supported the National Natural Science Foundation of China (No. 61202362) and the State Key Development Program of Basic Research of China (No. 2013CB329601)
describe an uncertain dataset. An uncertain dataset is constituted by many x-tuples. A large number of possible worlds would be produced as instances of the uncertain dataset. Let $D$ be the dataset. $T \in D$ is an x-tuple in $D$. $t \in T$ is a tuple belonging to $T$. $T$ can also be noted as $T(t)$. $s(t)$ is the score of $t$, $P(t)$ is $t$'s appearance probability and $P(T)$ is $T$'s appearance probability, where $P(T) = \sum_{t \in T} P(t)$. Let $W$ be the set of all possible worlds produced from $D$. For a possible world $w \in W$, $t \in w$ means tuple $t$ appears in $w$, and $T \in w$ means there is a tuple of $T$ appearing in $w$. Let $P(w)$ be the probability of $w$, and $P(w) = \prod_{T \in w} P(T) \prod_{T \notin w} (1 - P(T))$. Based on above assumptions, we propose following definitions.

Definition 1: Top $k_1$ tuple in a possible world: In a possible world $w$, a tuple $t \in w$ is a top $k_1$ tuple, if $t$ is ranked no less than $k_1$ according to its score in descending order.

Definition 2: $P_{k_1}$ probability of an x-tuple: For an x-tuple $T$, its $P_{k_1}$ probability, $P_{k_1}(T)$, is the probability sum of all possible worlds where one of $T$'s tuple is a top $k_1$ tuple. Formally, $P_{k_1}(T) = \sum_{w \in W(T)} P(w)$. $W(T)$ is the set of possible worlds, where there is a tuple in $T$ and $t$ is a top $k_1$ tuple.

Definition 3: Top $(k_1, k_2)$ query in an uncertain dataset: In an uncertain dataset $D$, the top $(k_1, k_2)$ query returns $k_2$ x-tuples with highest $P_{k_1}$ probabilities. It can be easily proved that top $(k_1, k_2)$ query satisfies five key properties [7]. Compared with the Global-topk query, the result set of the top $(k_1, k_2)$ query with $k_2' < k_2$ is contained by the result set with $k_2$ (Containment).

3. Basic Algorithm for Top $(k_1, k_2)$ Query

We propose a basic algorithm similar with M. Hua et al.'s research [3], which is used for top $k$ uncertain tuple query. We modify it for uncertain x-tuples query. For an x-tuple $T$, it can be proved that, $P_{k_1}(T)$ can be calculated as:

$$P_{k_1}(T) = \sum_{t \in T} P_{k_1}(t) P(t)$$

$P_{k_1}(t)$ is the probability that is no more than $k_1 - 1$ tuples with higher scores than $t$ appear. The proof of the Eq. (1) is given in [3]. $k_2$ x-tuples with largest probability $P_{k_1}(T)$ are returned as top $(k_1, k_2)$ uncertain query result. $P_{k_1}(t)$ can be calculated using a Dynamic Programming algorithm. Let $L = <t_1, t_2, \ldots, t_v>$ be the sorted tuple list in $D$ according to their scores in descending order. $P_{k_1}(t)$ is equal to the probability that more than $k_1 - 1$ x-tuples appear in front of $t_j$. Let $L(t_j) = <t_1, t_2, \ldots, t_{j-1}, t_j>$ be $t_j$'s prefix tuple list in $L$. Let $L'(t_j)$ be $t_j$'s prefix x-tuple list in $L$. An x-tuple in $L'(t_j)$ contains at least one tuple in $L(t_j)$. Let $L'(t_j) = <t_1, T_2, \ldots>$ without loss of generality. X-tuples in $L'(t_j)$ are in order. Suppose $t_x \in L(t_j)$, and for any tuple $t_x' \in T_{t_x}$ satisfying $t_x' \in L(t_j)$ but $x' \neq x$, we can get $x' > x$. We call $t_x$ the first tuple of $T_{t_x}$ in $L(t_j)$. Let $t_y$ be the first tuple of $T_{t_y}$ in $L'(t_j)$. If $x < y$, $T_{t_y}$ is in front of $T_{t_x}$ in $L'(t_j)$.

**Algorithm 1: Basic algorithm**

| Data: $D, k_1, k_2$ |
|---|
| Result: $k_2$ x-tuples |
| begin |
| $L_p \leftarrow \emptyset$; |
| sort all tuples in $D$ in descending order and construct list $L$; |
| for each x-tuple $T \in D$, do $P_{k_1}(T) \leftarrow 0$; |
| for each tuple $t_j \in L$ do |
| $L''(t_j) \leftarrow L_p - T(t_j)$ and calculate $P_{k_1}(t_j)$; |
| $P_{k_1}(T(t_j)) \leftarrow P_{k_1}(T(t_j)) + P_{k_1}(t_j)$; |
| if $T(t_j) \notin L_p$ then |
| $L_p \leftarrow L_p \cup T(t_j)$ and $P''(T(t_j)) \leftarrow P(t_j)$; |
| else |
| $P''(T(t_j)) \leftarrow P''(T(t_j)) + P(t_j)$; |
| output $k_2$ x-tuples in $D$ with largest $P_{k_1}$ probabilities. |

Let $L''(t_j) = L'(t_j) - T(t_j)$, then $P_{k_1}(t_j) = P(L''(t_j), k_1 - 1)$, which is the probability that no more than $k_1 - 1$ x-tuples in $L''(t_j)$ appear. Let $L''(t_j)[k]$ be the $k$th x-tuple in $L''(t_j)$. We set

$$P''(L''(t_j)[k]) = \sum_{1 \leq x < j - 1, T(t_x) = L''(t_j)[k]} P(t_x). \quad (2)$$

Let $L''(t_j)[1, k]$ be the list of first $k$ x-tuples in $L''(t_j)$. $P(L''(t_j)[1, k], i)$ is the probability the no more than $i$ x-tuples in $L''(t_j)[1, k]$ appear. Then $P(L''(t_j), k_1 - 1)$ can be calculated as follows:

$$P(L''(t_j)[1, 1, 0]) = 1 - P''(L''(t_j)[1]);$$

$$P(L''(t_j)[1, k_1, 0]) = P(L''(t_j)[1, k_1 - 1, 0]) (1 - P''(L''(t_j)[k]));$$

$$P(L''(t_j)[1, k_1, i]) = 1, \text{ if } 1 < k_1 \leq i;$$

$$P(L''(t_j)[1, k_1, i]) = P(L''(t_j)[1, k_1 - 1, i - 1]) P''(L''(t_j)[k]) + P(L''(t_j)[1, k_1 - 1, i]) (1 - P''(L''(t_j)[k])), \text{ if } 1 \leq i < k.$$ 

Then we can get:

$$P_{k_1}(t_j) = P(L''(t_j)[1, k_1 - 1]) = P(L''(t_j)[1, |L''(t_j)|], k_1 - 1),$$

$$j > 1 \text{ and } P_{k_1}(t_1) = P(L''(t_1), k_1 - 1) = 1.$$ 

For example in Tables 1 and 2, suppose $L_p = <t_{11}, t_{21}, t_{12}>$, $k_1 = 1$, $P(t_{11}) = 0.3$, $P(t_{12}) = 0.5$, $P(t_{21}) = 0.7$ and $t \notin T_1 \land t \notin T_2$, then $L''(t_1)[1] = T_1$ and $L''(t_2)[2] = T_2$. We can get:

$$P''(L''(t_1)[1]) = P(t_{12}) + P(t_{21}) = 0.3 + 0.5 = 0.8;$$

$$P''(L''(t_2)[2]) = P(t_{21}) = 0.7;$$

$$P(L''(t_1)[1, 1, 0]) = 1 - P''(L''(t_1)[1]) = 1 - 0.8 = 0.2;$$

$$P(L''(t_1)[1, 1, 1]) = 1;$$

$$P_{k_1}(t_1) = P(L''(t_1), 1) = P(L''(t_1), 1, 2) = P(L''(t_1)[1, 1], 0) \cdot P''(L''(t_1)[2]) + P(L''(t_1)[1, 1], 1) \cdot (1 - P''(L''(t_1)[2])) = 0.2 \cdot 0.7 + (1 - 0.7) = 0.44.$$ 

The basic algorithm is shown in Algorithm 1. $L_p$ is the prefix x-tuples list and initialized to be an empty set. All tuples in the dataset $D$ are sorted in list $L$ in descending order according to their scores. For each x-tuple $T$, $P_{k_1}(T)$ and $P''(T)$ are initialized as 0. For each x-tuple in $L_p$, at least
Algorithm 2: Improved algorithm with pruning

Data: $D$, $k_1$, $k_2$.
Result: $k_2$ x-tuples

begin
$H \leftarrow 0$, $L_0 \leftarrow \emptyset$, $L_O \leftarrow \emptyset$;
sort all tuples in $D$ in descending order and construct list $L$;
initialize $H(t_i)$, $H_1(T)$, $H_2(T)$ and $H(T)$ for each tuple $t_i$ and each x-tuple $T$;
for each tuple $t_i \in L$ do
  if $T(t_i)$ has been pruned, then update $L_p$, continue;
  if $H(T(t_i)) < H$, then prune $T(t_i)$, update $L_p$, continue;
  calculate $P_k(t_i)$ and add it to $L_p$;
  for each tuple $t_j$ in $L, i < j$ do
    update $H(t_j)$ and $H(T(t_j))$;
    if $H(T(t_j)) < H$, then prune $T(t_j)$;
    update $H_T(t(t_j), H_2(T(t_j))$ and $H(T(t_j)))$;
    if $H(T(t_j)) < H$, then prune $T(t_j)$, continue;
    if $T(t_j) \notin L_0$, then insert $T(t_j)$ into $L_0$ in order and remove the additional x-tuple if $|L_0| > k_2$;
else, resort x-tuples in $L_O$;
$H = H_2(L_0(k_2))$;
end
output $k_2$ x-tuples in $L_O$.

one of its tuples has been processed. Tuples in $L$ are processed one by one in the second for loop to calculate $P_k(t_i)$ and update $P_k(T(t_i))$. If $t_j$ is the first tuple of $T(t_j)$ to be processed, then $T(t_j)$ is added to the end of $L_p$. $P''(T(t_j))$ is set to be $P(t_j)$. Else, $P''(T(t_j))$ is updated as $P''(T(t_j)) + P(t_j)$.

4. Pruning Strategy

In the basic algorithm, all tuples and x-tuples have to be processed, which causes considerable time cost. However, query result often includes a small part of the entire data set. Some x-tuples with negligible $P_k$ probabilities can be pruned early. In this section, we estimate an upper bound of $P_k(T), H(T)$, for each x-tuple $T$. If $H(T)$ is smaller than a threshold $H$, $T$ could be pruned in advance.

In order to calculate $H(T)$ and $H$, we define two values $H_1(T)$ and $H_2(T)$ for each x-tuple $T$. We set $H_1(T) = \sum_{H \in D_1(T)} H(t) P(t)$, $H_2(T) = \sum_{H \in D_1(T)} P_k(t) P(t)$ and $H(T) = H_1(T) + H_2(T)$. $D_1(T)$ is the subset of tuples in $T$, where for each tuple $t \in D_1(T), P_k(t)$ has not been calculated yet. $D_2(T)$ is the subset of tuples in $T$, where for each tuple $t \in D_2(T), P_k(t)$ has been calculated. $H(t)$ is an upper bound of $P_k(t)$. For each x-tuple $T$, $H_2(T)$ is $P_k(T)$, where $k_2$th largest $H_2$ value all over the dataset can be the threshold $H$. For an x-tuple $T'$ with $H(T') < H$, there must be at least $k_2$ x-tuples, whose $P_k$ probabilities are larger than $P_k(T')$. For a tuple $t \in T$, if $P_k(t)$ has been calculated, then $H_1(T)$ would be updated by $H_1(T) = H_1(T) + H(t) P(t)$ and $H(T)$ would be updated by $H_2(T) + P_k(t) P(t)$, $H(T)$ is then updated by new values of $H_1(T)$ and $H_2(T)$.

Algorithm 2 describes the improved algorithm with the pruning strategy. In initialization stage, $H$ is 0. As in the basic algorithm $L_p$ is an empty set. $L_O$, the list of x-tuples sorted in descending order according to $H_2$ values, is also initialized to be an empty set. For each x-tuple $T$ and $t \in T$, $H(t)$, $H_1(T)$, $H_2(T)$ and $H(T)$ are initialized. Details of initialization method are given in Sect. 5. All tuples are sorted in list $L$ in descending order according to their scores. Suppose $L = \langle t_1, t_2, \ldots \rangle$. Tuples in $L$ are processed sequentially in the outside for loop. If $T(t_i)$ has been pruned, following tuples of $T(t_i)$ would not be processed. If $H(T(t_i)) < H$, $T(t_i)$ would be pruned. $L_p$ is updated as in the basic algorithm. In the inside for loop, for a tuple $t_j \in L$, where $i < j$ and $T(t_j)$ has not been pruned, $H(t_j)$ can be updated according to Theorem 1. If tuple $t_j$ does not satisfy Theorem 1, then we don’t update $H(t_k)$ for the following tuples, $j < k$, because $t_k$ would always not satisfy Theorem 1 too.

In order to prove Theorem 1, some lemmas are proposed.

Lemma 1: $P(L'(t_j) - T(t_j), k_1 - 1) \leq P(L'(t_j) - T(t_j), k_1 - 1), i \leq j$

Proof: This lemma can be proved according to the definition of $P(L''(t_j), k)$ and we ignore details for space limitation.

Lemma 2: $P(L''(t_j), k_1 - 1) \leq P(L''(t_j), k_1 - 1)$, if $P_k(t_i) \leq P_k(T(t_i))$ and $i \leq j, T(t_i) \neq T(j)$.

Proof: $P(L''(t_j), k_1 - 1) = P(L'(t_j) - T(t_j), k_1 - 1)$

\begin{align*}
&\leq P(L'(t_j) - T(t_j), k_1 - 1), \text{ according to Lemma 1} \\
&= P(L'(t_j) - T(t_j) - T(t_j), k_1 - 2) \cdot P_k(t_j) \\
&+ P(L'(t_j) - T(t_j) - T(t_j), k_1 - 1) \cdot (1 - P_k(t_j))
\end{align*}

(noted as $F_1$)

\begin{align*}
P(L'(t_j), k_1 - 1) &= P(L'(t_j) - T(t_j), k_1 - 1) \\
&= P(L'(t_j) - T(t_j) - T(t_j), k_1 - 2) \cdot P_k(t_j) \\
&+ P(L'(t_j) - T(t_j) - T(t_j), k_1 - 1) \cdot (1 - P_k(t_j))
\end{align*}

(noted as $F_2$)

We can prove that $F_1 - F_2 \leq 0$ (details are not shown for space limitation), that is $P(L'(t_j), k_1 - 1) \leq P(L''(t_j), k_1 - 1)$.

Then we prove Theorem 1:

In case of $T(t_j) = T(t_i)$, based on the definition,

\begin{align*}
P_k(t_j) &= P(L''(t_j), k_1 - 1) = P(L'(t_j) - T(t_j), k_1 - 1) \\
P_k(t_i) &= P(L''(t_i), k_1 - 1) = P(L'(t_i) - T(t_i), k_1 - 1)
\end{align*}

So that, $P_k(t_j) \leq P_k(t_i)$ according to Lemma 1.

In case of $T(t_j) \neq T(t_i)$ but $P_k(t_j) \leq P_k(t_i)$, we can get $P(L'(t_j), k_1 - 1) \leq P(L''(t_j), k_1 - 1)$ according to
Lemma 2, that is $P_k(t_j) \leq P_k(t_i)$. In summary, $P_k(t_j) \leq \min\{H(t_j), P_k(t_i)\}$.

5. Upper Bound Initialization

For any x-tuple $T$, $H_2(T)$ can be initialized according to its definition. $H_1(T)$’s initialization is based on that of $H(t)$, $t \in T$. It is obvious that $P_k(t) \leq 1$. So that, $H(t)$ can be initialized as 1.

In order to improve the pruning effect, the smaller $H(t)$ the better. Theorem 2 can be used to initialize $H(t)$ as a small value. In real applications, we can set $\epsilon \to 0$.

Theorem 2: For any tuple $t_i$ in $L = \langle t_1, t_2, \ldots \rangle$, $P_k(t_i)$ can be upper bounded by $H(t_i) = \exp[-v^2/2(v + k_1)] + \epsilon$, where $v = \sum_{1 \leq j < i} P(t_j) - 1 - k_1, \epsilon > 0$.

Proof: We use existing work of [3], [9]. If $\mu \geq F(p) = 1 + k_1 + \ln(1/p) + \ln^2(1/p) + 2k \ln(1/p)^{1/2}$, $P_k(t_i) \leq p$, where $\mu = \sum_{1 \leq j < i} P(t_j)$. Let $\mu = F(p'), we can get $p' = \exp[-v^2/2(v + k_1)]$. Because $F(p)$ is monotone decreasing with $p$, we can get $F(p' + \epsilon) \leq F(p' + \epsilon) = \mu$. According to Theorem 2 in [9], $P_k(t_i) \leq p' + \epsilon = \exp[-v^2/2(v + k_1)] + \epsilon$.

6. Experiments

In this section, we conduct experiments using a real dataset (the International Ice Patrol (IIP) Iceberg Sightings Dataset\(^1\)) and a synthetic dataset on a PC with 2.5 GHz intel Core i5 CPUs, 8.0 GB main memory, running Microsoft Windows 8 OS. The IIP dataset collects information on iceberg activities in the North Atlantic, which is used in [3] for uncertain top-k query. We consider each sighting record of iceberg in the dataset as a tuple. Tuples with similar time stamp and location construct an x-tuple. Each x-tuple contains as many as 30 tuples. We allocate each tuple an appearance probability as in [3]. The probability depends on the source of sighting, including: R/V (radar and visual), VIS (visual only), RAD (radar only) and so on. We also construct a synthetic dataset for our experiments. In the synthetic dataset, every object contains no more than 10 instances. The value of an instance satisfies normal distribution in every attribute. The probability is assigned to each tuple randomly.

We firstly test the effectiveness of the pruning strategy and the improved initialization strategy of $H(t)$. Experiments are processed in two datasets. We compare the basic algorithm without pruning (noted as basic), the algorithm using pruning strategy (noted as pruned) and the algorithm using the pruning strategy and the improved initialization strategy (noted as improved). Figure 1 shows the results of experiments with different values of parameter $k_1$. The horizontal axis shows the value of $k_1$ and the vertical axis shows the time cost (millisecond). It is obvious that, in two datasets, the pruning strategy and the improved initialization strategy can improve outlier detection efficiency significantly. Figure 2 shows the results of experiments with different values of parameter $k_2$. The horizontal axis shows the value of $k_2$ and the vertical axis is the time cost. The basic algorithm without pruning is hardly affected by $k_2$, because it has to calculate $P_k(t)$ for every tuple $t$, and $k_2$ is much smaller than the dataset size. With different values of $k_2$, the pruning strategy and the improved initialization strategy can accelerate query clearly.

We then show the scalability of our algorithms in Fig. 3. The horizontal axis shows datasets’ size, and the vertical axis shows the time cost (millisecond). We find that the basic algorithm costs much more time than others with various dataset size.

7. Conclusion

In this letter, we propose the top $(k_1, k_2)$ query in uncertain datasets. It is more suitable for uncertain query compared with existing research. The x-tuple model and the possible world semantics are used to describe uncertain datasets. We propose a basic algorithm for the top $(k_1, k_2)$ query. Then a pruning strategy and an improved initialization method are designed for acceleration. Experiments in real and synthetic datasets prove the performance of our method.

References

\[^1\]http://nsidc.org/data/g00807.html

[1] J. Li, B. Saha, and A. Deshpande, “A unified approach to ranking in probabilistic databases,” The VLDB Journal, vol.20, no.2.
[2] T. Bernecker, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Zuefle, “Scalable probabilistic similarity ranking in uncertain databases,” IEEE Trans. Knowl. Data Eng., vol.22, no.9, pp.1234–1246, 2010.

[3] M. Hua, J. Pei, W. Zhang, and X. Lin, “Ranking queries on uncertain data: A probabilistic threshold approach,” Proc. 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD ’08, pp.673–686, 2008.

[4] A. Parag, B. Omar, D.S. Anish, H. Chris, N. Shubha, S. Tomoe, and W. Jennifer, “Trio: A system for data, uncertainty, and lineage,” VLDB, 2006.

[5] N. Dalvi and D. Suciu, “Efficient query evaluation on probabilistic databases,” The VLDB Journal, vol.16, no.4, pp.523–544, 2007.

[6] G. Cormode, F. Li, and K. Yi, “Semantics of ranking queries for probabilistic data and expected ranks,” 2009 IEEE 25th International Conference on Data Engineering, pp.305–316, 2009.

[7] K. Yi, F. Li, G. Kollios, and D. Srivastava, “Efficient processing of top-k queries in uncertain databases with x-relations,” IEEE Trans. Knowl. Data Eng., vol.20, no.12, pp.1669–1682, 2008.

[8] X. Zhang and J. Chomicki, “Semantics and evaluation of top-k queries in probabilistic databases,” Distributed and Parallel Databases, vol.26, no.1, pp.67–126, 2009.

[9] T. Ge, S. Zdonik, and S. Madden, “Top-k queries on uncertain data: On score distribution and typical answers,” Proc. 35th SIGMOD International Conference on Management of Data, SIGMOD ’09, pp.375–388, 2009.