A Truly Dynamic Data Structure for Top-k Queries on Uncertain Data

Manish Patil, Rahul Shah, Sharma V. Thankachan

Louisiana State University, USA

July 20, 2011

Uncertain Data

- Uncertainty is prevalent in numerous application domains
  - data cleaning, data integration
  - sensor networks
  - moving object tracking
- Nature of uncertainty in data is quite varied, and often depends on the application domain
- Models which can handle uncertainty and semantics to define useful queries on uncertain data
  - Mystiq: Nilesh Dalvi and Dan Suciu (2004)
  - Trio: J. Widom (2005)
  - MayBMS L. Antova et al. (2007)
  - Orion: Sarvjeet Singh et al. (2008)
  - PrDB: Prithviraj Sen et al. (2009)

Modeling Uncertainty

- X-relation model
  - Tuple uncertainty: Probability captures the likelihood of the given tuple being present in the given relation
  - Correlation: An x-tuple τ specifies a set of mutually exclusive tuples, subject to the constraint $\sum_{t_i \in \tau} Pr(t_i) \leq 1$

Table: Traffic monitoring data: t_1 ,$\{t_2, t_4\}$, $\{t_3, t_6\}$, t_5

Time	Car Location	Plate Number	Speed	Probability	Tuple Id
11:55	L1	Y-245	130	0.30	t_1
11:40	L2	X-123	120	0.40	t_2
11:50	L4	X-123	105	0.50	t_4
12:05	L3	Z-541	110	0.20	t_3
12:15	L6	Z-541	80	0.45	t_6
12:10	L5	L-110	95	0.30	t_5

Modeling Uncertainty

- Possible World Semantics
  - An uncertain relation is viewed as a set of possible instances (worlds) and correlation among the tuples governs generation of these worlds
  - Probability of a possible world is computed based on the existence probabilities of tuples present in a world and absence probabilities of tuples in the database that are not part of a possible world

Traffic monitoring data: t_1 ,$\{t_2, t_4\}$, $\{t_3, t_6\}$, t_5

Probability	Tuple Id
0.30	t_1
0.40	t_2
0.50	t_4
0.20	t_3
0.45	t_6
0.30	t_5

Notation:

$p_i = Pr(t_i), \quad \bar{p}_i = 1 - Pr(t_i)$

$pw_1 = \{t_1, t_2, t_3\}$

$Pr(pw_1) = p_1 * p_2 * p_3 * \bar{p}_5 =$ 0.0168

Problem Definition

Given an uncertain relation $T = \{\tau_1, \tau_2, ..., \tau_N\}$, such that each tuple $t_i \in \tau$ is associated with a membership probability value $Pr(t_i) > 0$ and a score $score(t_i)$ computed based on a scoring function, the goal is to retrieve the top-k tuples.

- Top-k for Traditional database
  - Return k tuples with the highest score
- Top-k for Uncertain database
  - Answer depends not only on the scores but also on the membership probabilities of tuples
  - Various Top-k definitions covering different aspects of score-probability interplay have been proposed

Soliman et al. (2007) :	Uncertain top-k, Uncertain Rank-k
Hua et al. (2008) :	Probabilistic Threshold Query
Cormode et al. (2009) :	Expected Rank, Expected Score
Jestes et. al. (2010) :	Quantile Rank

Top-k Definitions

- Uncertain Rank-k: It returns a tuple for each i, such that it has maximum probability of appearing at rank i across all possible worlds.
- Computing Uncertain Rank-2:

  Let $T = \{t_1, t_2, .., t_N\}$ denote independent tuples sorted in non-increasing order of their score.

  When all tuples are independent, tuple t_i appears at position 2 in a possible world pw if and only if exactly 1 tuple with a higher score value appear in pw.

$$Pr(t_1, 2) = 0$$

$T =$	t_1	t_2	t_3
	0.1	0.5	0.2

$$Pr(t_2, 2) = p_1 * p_2 = 0.05$$

$$Pr(t_3, 2) = p_1 * \bar{p_2} * p_3 + \bar{p_1} * p_2 * p_3 = 0.1$$

$$Pr(t_2, 2) < Pr(t_3, 2)$$

Prior Work

- Focused on answering top-k queries on a static uncertain data
- Linear scan of tuples achieves the best bound so far (query time of an algorithm depends on the choice of a top-k definition)

- Chen and Yi (2007) proposed a linear space dynamic data structure to support arbitrary insertions and deletions
  - Works only for independent tuples
  - Can be built for some fixed k and can not answer top-j query, $j > k$
  - Time required for handling update (insert/delete) depends on k
- Jin et al. (2008) proposed a framework for sliding window top-k queries on uncertain streams
  - Assumes random-order stream model which significantly reduces the space requirement
  - In the worst-case scenario takes linear space

Handling Dynamic Data

- Computing Uncertain Rank-2:

  Existing top-k answers do not provide any useful information for re-computation of query answers after insertion/deletion of a tuple.

  Let $T = \{t_1, t_2, .., t_N\}$ denote independent tuples sorted in non-increasing order of their score.

$T =$	t_1	t_2	t_3
	0.1	0.5	0.2

$$Pr(t_2, 2) = 0.05 < Pr(t_3, 2) = 0.1$$

$T =$ t_0	t_1	t_2	t_3
0.25	0.1	0.5	0.2

$$Pr(t_2, 2) = 0.15 > Pr(t_3, 2) = 0.0975$$

$$Pr(t_2, 2) = p_0 * \bar{p}_1 * p_2 + \bar{p}_0 * p_1 * p_2 = 0.15$$

$$Pr(t_3, 2) = p_0 * \bar{p}_1 * \bar{p}_2 * p_3 + \bar{p}_0 * p_1 * \bar{p}_2 * p_3 + \bar{p}_0 * \bar{p}_1 * p_2 * p_3 = 0.0975$$

Parameterized Ranking Function

- Parameterized Ranking Function (*PRF*) [Li et al. 2009]:

  $PRF^e(\alpha)$ is defined as,

$$\Upsilon(t_i) = \sum_r \alpha^{r-1} \times Pr(t_i, r)$$

  - α is a constant
  - $Pr(t_i, r)$ denotes the probability of a tuple t_i being ranked at position r across all possible worlds
  - Returns k tuples with the highest Υ values

- For some value of α with the constraint $\alpha < 1$, PRF^e can approximate many existing top-k definitions [Li et al. 2009]

- $PRF^e(\alpha)$ is well suited for answering top-k queries on a dynamic collection of tuples

Computing $\Upsilon(t_i)$ - Independent tuples

$$\Upsilon(t_i) = \sum_r \alpha^{r-1} \times Pr(t_i, r)$$

We assume all scores to be unique and let $t_1, t_2, \ldots, t_N$ denotes ordering of the tuples in T when sorted in descending order of the score.

Let $S_{i,r}$ be the probability that a randomly generated world from $\{t_1, t_2, \ldots, t_i\}$ has exactly r tuples [Yi et al. 2008].

$$Pr(t_i, r) = p_i S_{i-1,r-1}$$

where,

$$S_{i,r} = \begin{cases} p_i S_{i-1,r-1} + (1-p_i) S_{i-1,r} & \text{if } i \geq r > 0 \\ 1 & \text{if } i = r = 0 \\ 0 & \text{otherwise.} \end{cases}$$

$$\boxed{\Upsilon(t_i) = p_i \prod_{j<i} (1 - (1-\alpha) p_j)}$$

Computing $\Upsilon(t_i)$ - Independent tuples

$$\Upsilon(t_i) = \sum_r \alpha^{r-1} Pr(t_i, r) = \sum_r \alpha^{r-1} p_i S_{i-1,r-1}$$

$$\frac{\Upsilon(t_i)}{p_i} = \sum_r \alpha^{r-1} S_{i-1,r-1} = \sum_r \alpha^r S_{i-1,r}$$

Similarly,

$$\begin{aligned}\frac{\Upsilon(t_{i+1})}{p_{i+1}} &= \sum_r \alpha^r S_{i,r} = \sum_r \alpha^r (p_i S_{i-1,r-1} + (1-p_i) S_{i-1,r}) \\ &= \alpha p_i \sum_r \alpha^{r-1} S_{i-1,r-1} + (1-p_i) \sum_r \alpha^r S_{i-1,r} \\ &= (1-(1-\alpha)p_i)\Upsilon(t_i)/p_i\end{aligned}$$

We have the base case, $\Upsilon(t_1) = p_1$. Therefore,

$$\Upsilon(t_i) = p_i \prod_{j<i} (1-(1-\alpha)p_j)$$

Computing $\Upsilon(t_i)$ - Independent tuples

Theorem

When all tuples in T are independent, $\Upsilon(t_i)$ can be computed as follows,

$$\Upsilon(t_i) = m_i \prod_{j<i} c_j$$

where $m_i = p_i$ and $c_j = 1 - (1 - \alpha)p_j$

- Tuple t_i contributes $m_i = p_i$ for the computation of its own Υ value.
- Tuple t_i contributes $c_i = 1 - (1 - \alpha)p_i$ of computing Υ value for all tuples having score less than that of

Computing $\Upsilon(t_i)$ - Independent tuples

- Apply Divide and Conquer for answering top-k query as shown below
- Relative ordering among tuples $t_1, .., t_k$ and tuples $t_{k+1}, .., t_N$ based on their Υ value is preserved
- Top-k answers computed over relations T_{left} and T_{right} can be utilized for efficient computation of query answers for merged relation T

$$T_{left} = \underbrace{t_1 \quad t_2 \quad t_3 \quad \ldots \quad t_k}$$

$$\Upsilon_{left}(t_i) = m_i \prod_{1 \leq j < i \leq k} c_j$$

$$T_{right} = \underbrace{t_{k+1} \quad t_{k+2} \quad \ldots \quad t_N}$$

$$\Upsilon_{right}(t_i) = m_i \prod_{k+1 \leq j < i \leq N} c_j$$

$$T = \underbrace{t_1 \quad t_2 \quad t_3 \quad \ldots \quad t_k} \quad \underbrace{t_{k+1} \quad t_{k+2} \quad \ldots \quad t_N}$$

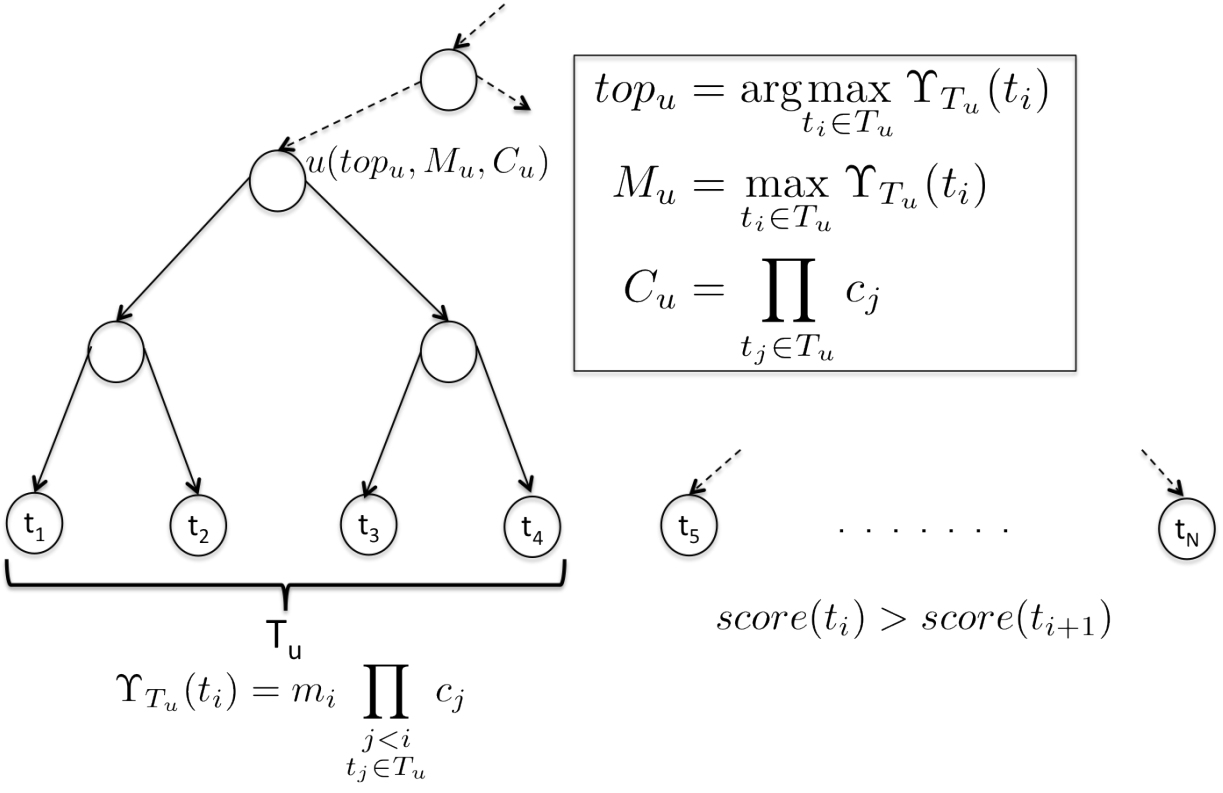
$$\Upsilon(t_i) = \Upsilon_{left}(t_i)$$

$$\Upsilon(t_i) = \Upsilon_{right}(t_i) * C_{left}$$

$$= \Upsilon_{right}(t_i) * \prod_{j=1}^{k} c_j$$

Our Data Structure

Our structure is a balanced binary search tree Δ (e.g. Red black tree)

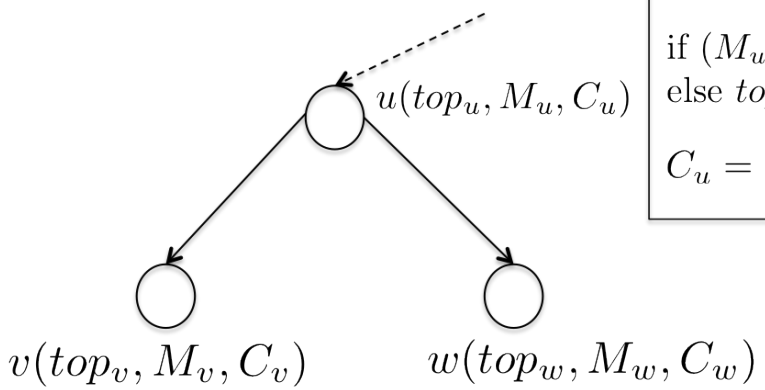


- top_u is the tuple with highest Υ value i.e M_u among tuples in sub-relation T_u (computed over sub-relation T_u)
- C_u is the contribution of all tuples in T_u towards Υ value of tuple t_i with score value lower than that of any tuple in the subtree of node u

Our Data Structure

- Total space requirement of our data structure is $O(N)$: Only a constant number of information is stored at each node, and the number of nodes are bounded by $O(N)$

- Computing necessary information at each node u (top_u, M_u, C_u):



$M_u = max(M_v, C_v * M_w)$

if $(M_u == M_v)$ then $top_u = top_v$
else $top_u = top_w$

$C_u = C_v * C_w$

v is a leaf node
corresponding to tuple t_i

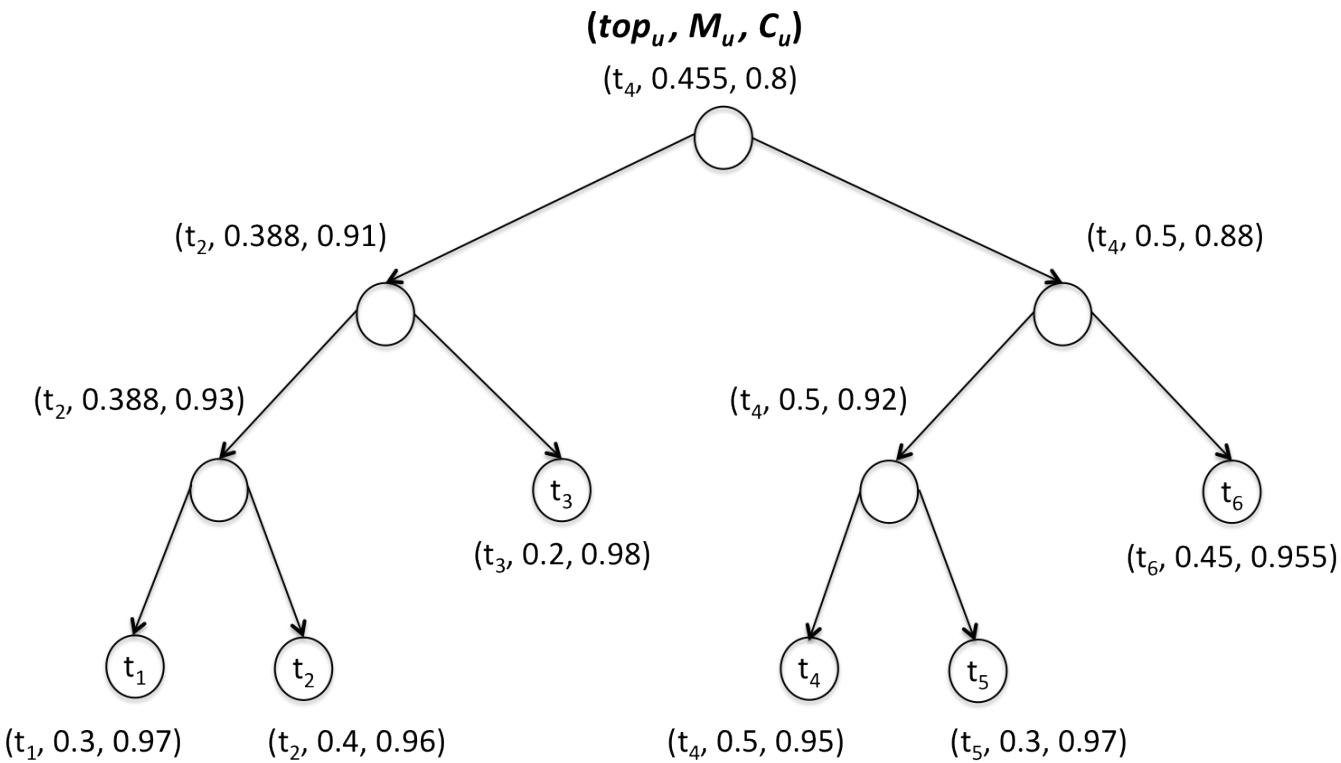
$top_v = t_i, M_v = m_i$
$C_v = c_i$

Our Data Structure

- The data structure Δ maintains a collection of tuples such that top-1 tuple, $t^1 = top_{root}$ and $\Upsilon(t^1) = M_{root}$.

Data Structure for Traffic Monitoring database ($\alpha = 0.9$)

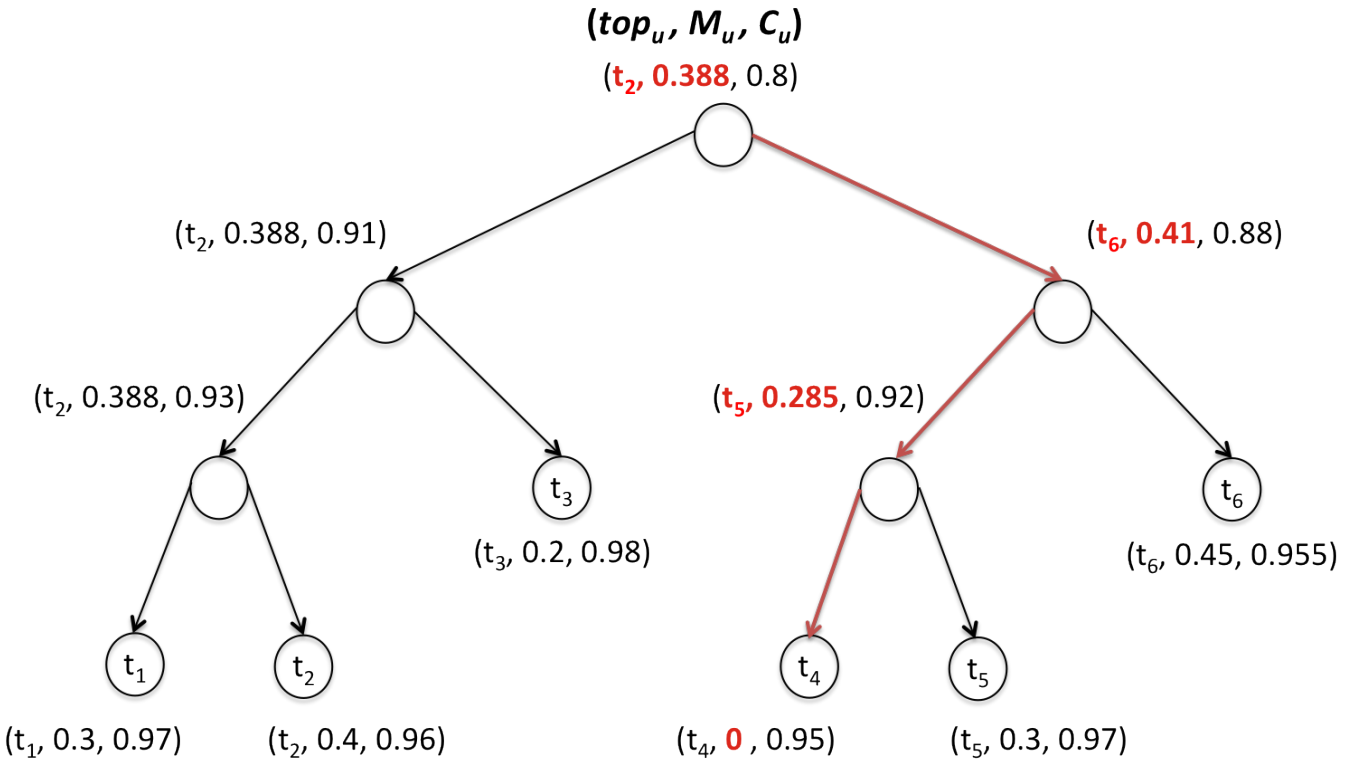
T = $t_1(0.3), t_2(0.4), t_3(0.2), t_4(0.5), t_5(0.3), t_6(0.45)$



Retrieving top-k tuples: $O(k \log n)$

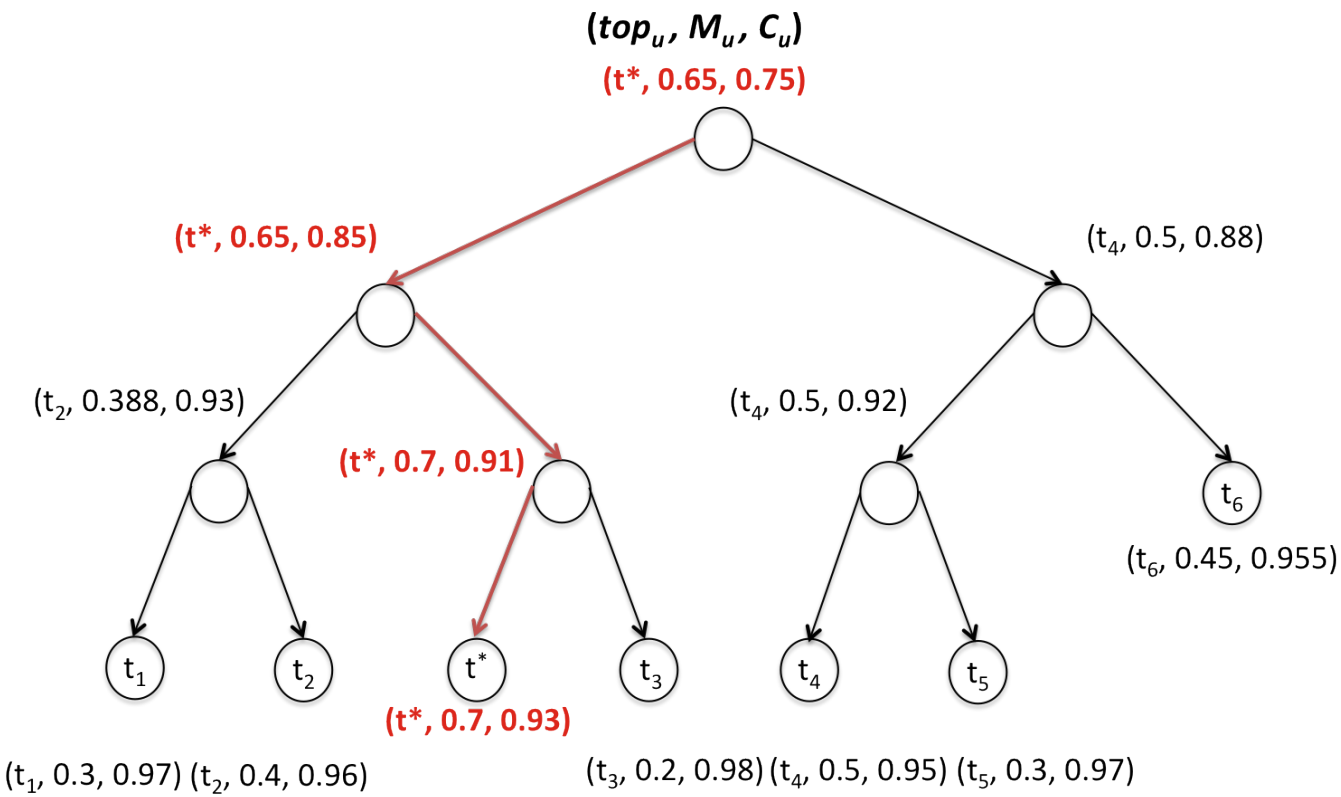
- Retrieve the top-2 tuple t^2: After retrieving t^1, set $M_u = \Upsilon(t^1) = 0$ in the leaf node corresponding to tuple t^1 (node u), as a result tuple with next highest Υ value will be propagated as top_{root} instead of t^1

Revert back the changes done in data structure for answering top-k query by restoring the M values for k retrieved tuples



Inserting a new tuple: $O(\log n)$

- To insert a tuple $t^*(0.7)$ into relation T
  - Insert a new leaf node corresponding to tuple t^*, and propagate the changes upwards
  - If tuple t^* is independent of all other tuple in T then at this point all nodes in the tree Δ have correct values for C and M



Computing $\Upsilon(t_i)$ - Supporting Correlation's

Theorem

For an uncertain relation T, $\Upsilon(t_i)$ can be computed as,

$$\Upsilon(t_i) = m_i \prod_{j<i} c_j$$

where $m_i = \frac{p_i}{(1-(1-\alpha)\hat{p}_i)}$, $c_i = \frac{1-(1-\alpha)(\hat{p}_i+p_i)}{1-(1-\alpha)\hat{p}_i}$ and $\hat{p}_i = \sum t_r$, where t_i and t_r are mutually exclusive and $r < i$.

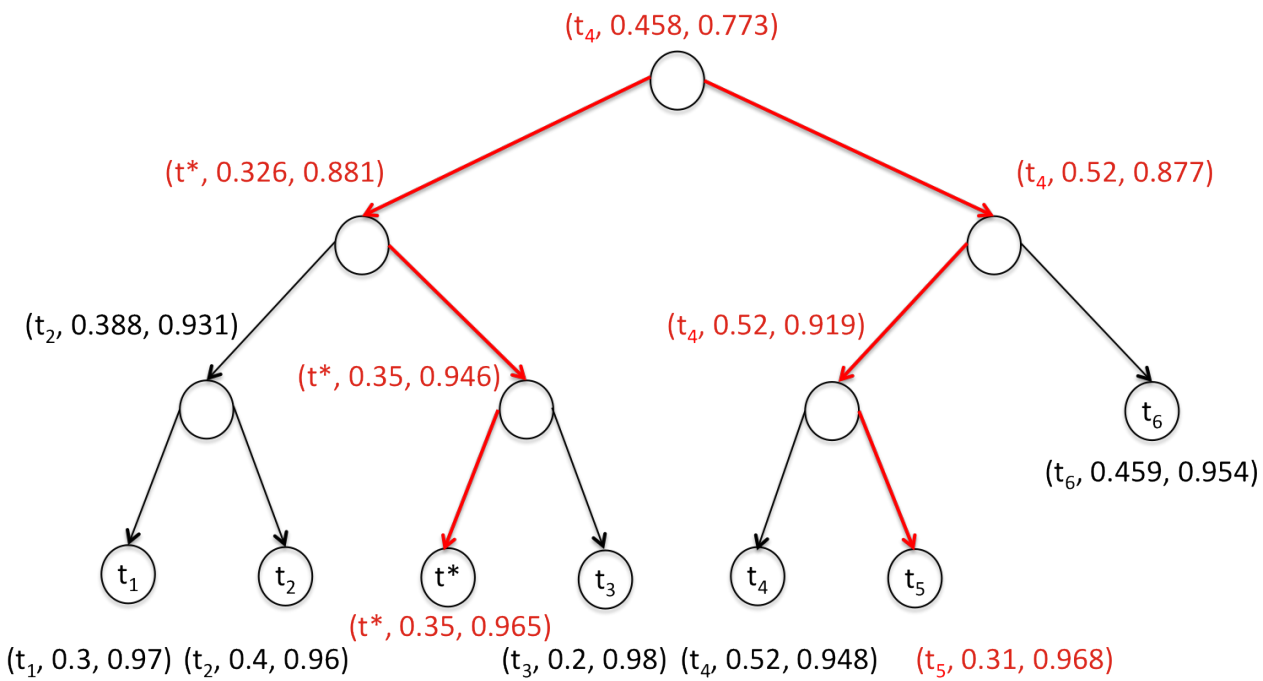
- Above theorem is applicable for relation T with dependent as well as independent tuples
- Tuple t_i contributes m_i for the computation of its own Υ value
- The contribution c_i of a tuple t_i to the Υ value of a tuple t_j is the same for all $j > i$. Hence, relative ordering is preserved making it possible to use divide and conquer approach

Inserting a new tuple: $O(\log n)$

- To insert a tuple $t^*(0.35)$ into relation T
  - If t^* is not independent, then update $M_u = m_j$ and $C_u = c_j$ values for all leaf nodes corresponding to dependent tuples t_j with score lower than that of t^*
  - If an x-tuple is restricted to have constant number of tuples, tuple insertion can be handled in $O(\log N)$ time

T = $t_1(0.3), \{t_2(0.4), t_4(0.5)\}, \{t_3(0.2), t_6(0.45)\}, t_5(0.3)$ $(\alpha = 0.9)$

Assuming tuple t^* is mutually exclusive with t_5



Our Result

Theorem

A collection of uncertain data (N tuples) can be maintained using a linear size dynamic data structure, which can retrieve top-k tuples (based on ranking function $PRF^e(\alpha)$ in $O(k \log N)$ time, and can support insertion or deletion of a tuple t in $O(d \log N)$ time, where d is the number of tuples which are related to t.

Experimental Setup: Datasets[2]

- Synthetic dataset:
  - Score of a each tuple is chosen uniformly at random from [0, 100000]
  - Tuple probability is uniformly distributed in $(0.5 \times 10^{-5}, 1.5 \times 10^{-5})$
  - The number of tuples involved in each x-tuple follows the uniform distribution over range (2,10)

- International Ice Patrol (IIP) Iceberg Sighting Database[1]:
  - We use the attribute "*number of days drifted*" as ranking score
  - "*Confidence-level*" attribute (according to the source of sighting) is converted into probabilities as follows:

radar and visual: 0.8	low earth orbit satellite: 0.5
visual only: 0.7	medium earth orbit satellite: 0.4
radar only: 0.6	high earth orbit satellite: 0.3
estimated: 0.4	

  - All tuples are assumed to be independent

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

[2] Both datasets have 1,00,000 tuples each

Experimental Results

All experiments were conducted on 2.4 GHz Intel Core 2 Duo machine with 2GB memory running MAC OS 10.6.4

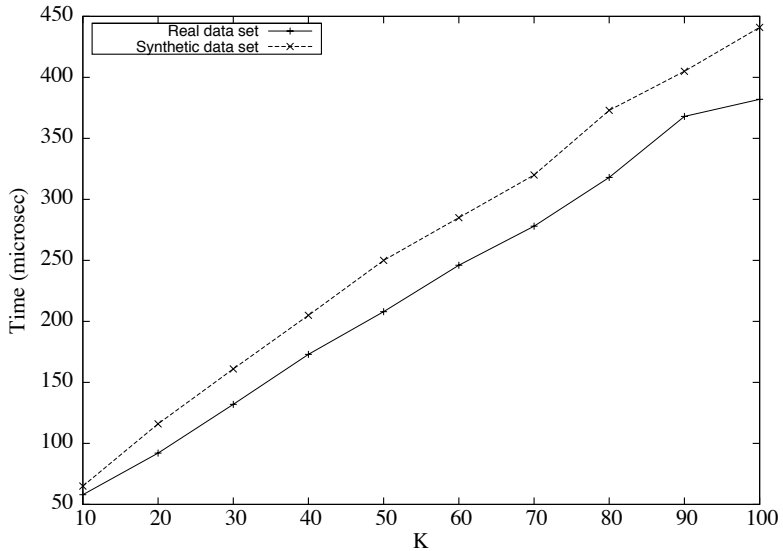


Figure: Top-k query performance on IIP and synthetic data

- Linear dependance of query time on k as obtained in the time bounds
- Correlations among tuples does not affect the query time of our data structure

Experimental Results

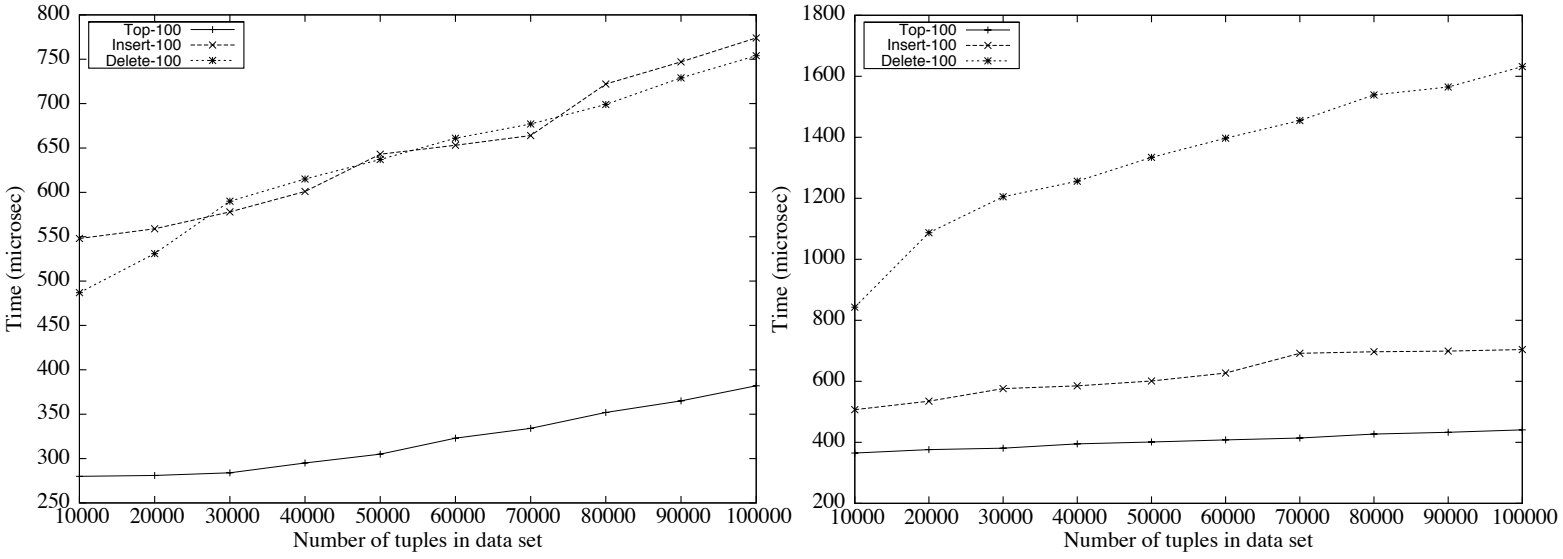


Figure: Processing (insert, delete, top-k) cost on (a) IIP dataset (b) synthetic dataset

- Average insertion/deletion time of a tuple is less for IIP dataset (all tuples are independent) than in case of synthetic data having correlations
- Synthetic dataset:
  - Position of a new tuple to be inserted in score-sorted ordering of tuples is selected at random whereas tuple to be deleted is always the highest scored tuple in the victim x-tuple
  - More number of Update-leaf operations per tuple deleted than for tuple inserted

Questions?

Thank You!