

URank: Formulation and Efficient Evaluation of Top-k Queries in Uncertain Databases

Mohamed A. Soliman
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
m2ali@cs.uwaterloo.ca

Ihab F. Ilyas
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
ilyas@uwaterloo.ca

Kevin Chen-Chuan Chang
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL, USA
kcchang@cs.uiuc.edu

ABSTRACT

Top- k processing in uncertain databases is semantically and computationally different from traditional top- k processing. The interplay between query scores and data uncertainty makes traditional techniques inapplicable. We introduce URank, a system that processes new probabilistic formulations of top- k queries in uncertain databases. The new formulations are based on marriage of traditional top- k semantics with possible worlds semantics. URank encapsulates a new processing framework that leverages existing query processing capabilities, and implements efficient search strategies that integrate ranking on scores with ranking on probabilities, to obtain meaningful answers for top- k queries.

Categories and Subject Descriptors

H.2.4 [Database Management]: Systems

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Query Processing, Probabilistic data, Uncertain data, Topk, Ranking

1. INTRODUCTION

Efficient processing of uncertain (probabilistic) data is a crucial requirement in different domains including sensor networks, moving objects tracking, and data cleaning. Current query processing techniques for uncertain data focus on Boolean queries. However, uncertainty usually arises in data exploration, decision making, and data cleaning scenarios which all involve aggregation and ranking. Top- k queries are dominant type of queries in such applications. This demo presents the URank system, which is one of the first systems to process top- k queries in an uncertain database.

A traditional top- k query returns the k objects with the maximum scores based on some scoring function. In the uncertain world, such a clean definition does not exist. The interplay between query scores and data uncertainty presents interesting issues in both query semantics and processing. For example, there are numerous possible interpretations for uncertain top- k queries. Moreover, reporting a tuple in a top- k answer does not depend only on its

	Time	Radar Loc	Car Model	Plate No	Speed	Conf	World	Prob.
t1	11:45	L1	Honda	X-123	130	0.4	PW ¹ = $\{t1, t2, t6, t4\}$	0.112
t2	11:50	L2	Toyota	Y-245	120	0.7	PW ² = $\{t1, t2, t5, t6\}$	0.168
t3	11:35	L3	Toyota	Y-245	80	0.3	PW ³ = $\{t1, t6, t4, t3\}$	0.048
t4	12:10	L4	Mazda	W-541	90	0.4	PW ⁴ = $\{t1, t5, t6, t3\}$	0.072
t5	12:25	L5	Mazda	W-541	110	0.6	PW ⁵ = $\{t2, t6, t4\}$	0.168
t6	12:15	L6	Nissan	L-105	105	1.0	PW ⁶ = $\{t2, t5, t6\}$	0.252
							PW ⁷ = $\{t6, t4, t3\}$	0.072
							PW ⁸ = $\{t5, t6, t3\}$	0.108

Rules: (t2 $\oplus$ t3), (t4 $\oplus$ t5)

(a)
(b)

Figure 1: Uncertain Relation and its Possible Worlds Space

score, but also on its probability, and the scores and probabilities of other tuples. Tuple scores and uncertainty information impose two different ranking dimensions that need to be integrated to obtain meaningful answers for top- k queries.

In particular, we focus on the problem of such ranking queries in “possible worlds” semantics. Many uncertain data models adopt *possible worlds* semantics, where an uncertain relation is viewed as a set of possible instances (worlds). The structure of these worlds is governed by underlying *generation rules*, e.g., mutual exclusion of tuples that represent the same real-world entity. Such rules could arise naturally with unclear data or could be customized to enforce application requirements, reflect domain semantics, or maintain lineage and data inter-dependencies. To illustrate, consider the following example.

EXAMPLE 1. Consider a radar-controlled traffic, where cars’ speed readings are stored in a database. Radar units detect speed automatically, while car identification, e.g., by plate number, is usually performed by a human operator. In this database, multiple sources of uncertainty exist. For example, radar readings can be interfered with by high voltage lines, close by cars cannot be precisely distinguished, or human operators might make identification mistakes. Figure 1(a) is a snapshot of a relation recording radar readings over the past hour. Each reading is associated with a confidence field “conf” indicating its probability. Radar locations imply that the same car cannot be detected at two different locations within 1 hour interval. This constraint is reflected by the shown exclusiveness rules.

In Example 1, two interesting top- k queries are to report:

- the top- k speeding cars in the last hour.
- a ranking over the models of the top- k speeding cars (e.g., for insurance purposes).

While ranking queries are clearly important for scenarios that involve data uncertainty, as Example 1 illustrates, their processing is challenging. The processing involves both ranking and aggregation over a possible worlds space whose size is exponential in the database size (Figure 1(b) shows the possible worlds of the depicted relation ranked on *speed*). A naïve approach to obtain answers to the above queries is to materialize the whole possible worlds space, find the top- k answer in each world, and aggregate the probabilities of identical answers. The URank system aims at efficiently processing such querying.

New Techniques: For efficient processing of top- k queries over uncertain data, our techniques have two pillars:

- *Search Space Model:* We model uncertain top- k processing as a *state space search* problem. We introduce several space navigation algorithms with optimality guarantees, on the number of accessed tuples and the number of *visited* search states, to find the most probable top- k answers.
- *Processing Framework:* We construct a framework integrating space navigation algorithms and data access methods leveraging existing RDBMS technologies.

Our Contributions: To the best of our knowledge, URank is the first system that fully integrates ranking with uncertainty. The URank system handles new *probabilistic formulations* of top- k queries based on an efficient *processing framework*—both of which, we believe, will set the foundation for the intermarriage of ranking and uncertainty.

2. SYSTEM OVERVIEW

URank is based on a novel processing framework, as depicted in Figure 2, that leverages RDBMS storage, indexing and query processing techniques to compute the most probable top- k answers in an uncertain database. The processing framework consists of two main layers:

- *The Storage Layer:* Tuple retrieval, indexing and traditional query processing (including score-based ranking) are the main functionalities provided by the storage layer. Different data access methods are provided to allow the upper processing layer to retrieve tuples along with their probabilistic information. Generation rules and tuple correlation/dependency information are stored in a rule store, e.g., a Bayesian Network, to allow for probability computation.
- *The Processing Layer:* The processing layer retrieves uncertain tuples from the underlying storage layer, and efficiently navigates the space of possible worlds to compute the most probable top- k answers, while materializing only the needed-to-see parts in the space.

We formulate our problem as searching the space of states that represent all possible top- k answers. Definition 1 gives a formal definition of the space state.

DEFINITION 1. Top- l State: A top- l state s_l is a tuple vector of length l that appears as the top- l answer in one or more valid possible worlds based on some scoring function. $\square$

A top- l state s_l is *complete* if $l = k$. *Complete* states represent possible top- k answers. The *probability of state s_l* is the summation of the probabilities of all possible worlds, where s_l is the top- l answer.

We introduce in [3] two different formulations for uncertain top- k queries based on the above space definition. In the first formulation, U-Top k , we rank the *necessary-to-materialize* states based on probability, with an early-stopping criteria. A U-Top k query reports a top- k vector with the highest aggregated probability across all worlds. In the second formulation, U- k Ranks, we group and aggregate states on the level of individual ranks. A U- k Ranks query reports a set of tuples that might not form together the most probable top- k vector. However, each tuple is a clear winner at its rank over all worlds.

The main components of the processing layer are depicted in Figure 2. The *Rule Engine* is responsible for retrieving and caching tuple dependency information that is required to compute state probabilities. The details of the *Rule Engine* can be an intelligent business process, a sophisticated model, e.g., Bayesian network, or a lineage log as in TRIO [1]. The formulation of states is performed by the *State Formulation* module. The *Space Navigation* module implements navigation algorithms that partially materialize the possible worlds space and prioritize (partial) top- k answers based on their probabilities.

In URank, we assume an *iterator* interface to retrieve tuples from the *storage layer* in sequential access retrieval. In [3], we proved that *sorted score access* minimizes the number of needed-to-see tuples, under our settings, to report uncertain top- k query answers.

We illustrate the interaction of framework components in the following. Figure 3 depicts the components interaction to obtain uncertain top- k answers for the database in Example 1 using *speed* as the tuple score. Tuples are produced by a top- k query plan and are submitted to the Space Navigation module ordered on their scores. Candidate top- l states are materialized by the Space Navigation module. In order to compute the probability of some state, the State Formulation module creates a corresponding combination of tuple events. For example, the tuple event combination corresponding to the top-2 state (t_1, t_5) , is $(t_1, \neg t_2, t_5)$. The formulated combination is submitted to the Rule Engine, which consults the underlying data and rule store, to compute the probability.

Since the number of possible top- l states is exponential in the number of retrieved tuples, our primary *cost metric* is the number of accessed tuples from the storage layer. Moreover, the space navigation module implements efficient search strategies to rank and prune the materialized states in order to avoid space explosion.

The URank system also includes efficient algorithms that cut down the computation significantly under tuple independence, e.g., a key-join query with no dependencies among base tuples. The number of maintained states in this case is linear in k by exploiting a state comparability criterion for more aggressive pruning. The reader is referred to [3] for details of the algorithms and techniques used in each of the system modules.

3. DEMONSTRATION SCENARIO

We demonstrate the URank system by illustrating the materialized parts of the huge possible worlds space, memory and I/O costs, and query execution time. For this demonstration, we implement a Rule Engine prototype that manages simple dependencies over tuple pairs in *base* relations such as exclusiveness, implication and equivalence. Exclusiveness rules means that tuples never co-exist in the same world, implication rules mean that the existence of one tuple in some world leads to the existence of the other tuple in the same world, while equivalence rules mean that both tuples always

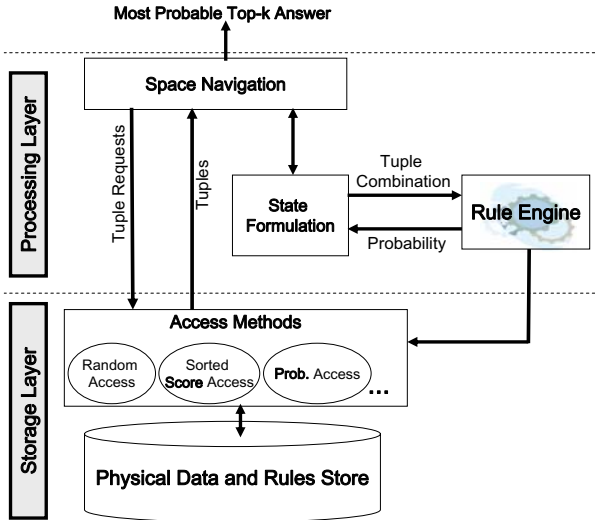


Figure 2: Uncertain top- k Processing Framework

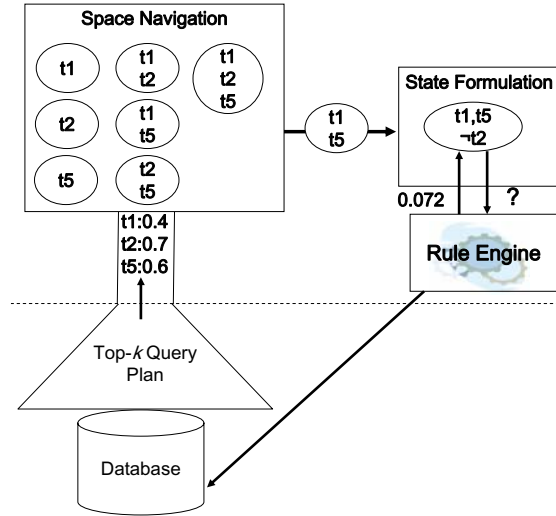


Figure 3: Interaction of Framework Components



Figure 4: URank Graphical User Interface

exist together. In addition, the Rule Engine manages dependencies among *intermediate* joined tuples, e.g., $(r1 \bowtie s1)$ vs. $(r1 \bowtie s2)$. The rule engine extracts the relevant dependencies for each state under consideration, and applies inference techniques to compute state probability. A full-fledged rule engine can be plugged into our framework to extend its support to different query types involving other types of dependencies.

We use datasets with different score and probability distributions to illustrate the robustness of our techniques in different settings. We show that our techniques scale up with practical values of k , and that materialized space is accommodated in main memory in most cases. Figure 4 shows our demonstration GUI. The demonstration scenario is as follows:

1. The user enters an uncertain top- k query in window (1), Figure 4, using a simple SQL-like language, specifying the required top- k query semantics (U-Top k or U- k Ranks), k , tuple source (a SQL query), and the names of scoring and confidence attributes. We implemented query rewriting methods

to transform this query into a traditional SQL query that is compiled into a rank-aware query plan using the RankSQL system [2] to pipeline score-ranked tuples to our processing layer when needed. The user receives query results in the results pane along with some statistics on query execution time, consumed tuples, and memory consumption.

2. The user can inspect the rules that apply to base tuples and edit them using the rule editor shown in window (2), Figure 4. The user can also filter the rules by rule type, or tuple identifier. By manipulating the underlying rules and re-executing the queries, the user can understand the effect of tuple dependencies and rule complexity on the system performance.
3. The materialized states are visually shown in the tracing mode shown in window (3), Figure 4. The user observes how the system generates and ranks states when processing each tuple, and how the space is lazily explored to avoid the combinatorial explosion that is inherent to the nature of our problem. The left pane shows a snapshot of the state queue of our UTop- k algorithm [3] at some point during its execution. Each state is illustrated as a vector of score-ordered tuples, and is associated with its probability, as computed by the rule engine. The right-top pane is a compact view of the state queue showing all states along with their lengths and probabilities. The right-bottom pane shows the relevant rules that apply to the highlighted state. The system, in the tracing mode, animates space navigation and queue maintenance to allow the user to interact with the state space and understand how query answers are computed.

4. REFERENCES

- [1] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB*, 2006.
- [2] C. Li, K. C.-C. Chang, I. F. Ilyas, and S. Song. RankSQL: Query algebra and optimization for relational top- k queries. In *SIGMOD*, 2005.
- [3] M. A. Soliman, I. F. Ilyas, and K. C.-C. Chang. Top- k query processing in uncertain databases. In *ICDE*, 2007.