Humans Fighting Uncertainty: Crowdsourcing for Top-K Query Processing (Extended abstract)

Eleonora Ciceri, Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi

Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Milan, Italy
first.last@polimi.it

Abstract. Querying uncertain data has become a prominent application due to the proliferation of user-generated content from social media and of data streams from sensors. When data ambiguity cannot be reduced algorithmically, crowdsourcing proves a viable approach, which consists in posting tasks to humans and harnessing their judgment for improving the confidence about data values or relationships. This paper tackles the problem of processing top-K queries over uncertain data with the help of crowdsourcing for quickly converging to the real ordering of relevant results. Several offline and online approaches for addressing questions to a crowd are defined and contrasted on both synthetic and real data sets, with the aim of minimizing the crowd interactions necessary to find the real ordering of the result set.

1 Introduction

Both social media and sensing infrastructures are producing an unprecedented mass of data characterized by their uncertain nature, due to either the noise inherent in sensors or the imprecision of human contributions. Therefore query processing over uncertain data has become an active research field. In the well-known class of applications commonly referred to as "top-K queries", the objective is to find the best K objects matching the user's information need, formulated as a scoring function over the objects' attribute values. If both the data and the scoring function are deterministic, the best K objects can be univocally determined and totally ordered so as to produce a single ranked result set (as long as ties are broken by some deterministic rule). However, in application scenarios involving uncertain data and fuzzy information needs, this does not hold: when either the attribute values or the scoring function are nondeterministic, there may be no consensus on a single ordering, but rather a space of possible orderings. To determine the correct ordering, one needs to acquire additional information so as to reduce the amount of uncertainty associated with the queried data and consequently the number of orderings in such a space. An emerging trend in data processing is *crowdsourcing*, defined as the systematic engagement of humans in the resolution of tasks through online distributed work.

In this paper, we present a synthesis of the results described in [2]. Our approach combines human and automatic computation in order to solve complex problems: when data ambiguity can be resolved by human judgment, crowdsourcing becomes a viable tool for converging towards a unique or at least less uncertain query result. The goal of this paper is to define and compare task selection policies for uncertainty reduction via crowdsourcing, with emphasis on the case of top-K queries.

Problem formulation: Given a data set with uncertain values and an allowed budget of posable questions, determine the set of questions (to be posed to a crowd) that minimizes the expected residual uncertainty of the result, possibly leading to a unique ordering of the top K results.

2 Background

We consider the problem of answering a top-K query over a relational database table T containing N tuples. The relevance of a tuple to the query is modeled as a score. Let $t_i \in T$ be a tuple in the database, and $s(t_i)$ be the score of tuple t_i, computed by applying a *scoring function* over t_i's attribute values. When the score $s(t_i)$ is known for each tuple t_i, the tuples in T can be totally ordered in descending order of $s(t_i)$ by breaking ties deterministically. Instead, if the score $s(t_i)$ is uncertain and thus modeled as a random variable (with a probability density function (pdf) f_i), such an uncertainty induces a partial order over the tuples. Indeed, when the pdf's of two tuples overlap, their relative order is undefined (see, e.g., the pfd's shown in Figure 1(a) for three tuples t_1, t_2, t_3). Therefore, we define the *space of possible orderings* as the set of all the total orderings compatible with the given score probability functions. This space can be represented by means of a *tree of possible orderings* $\mathcal{T}$ (henceforth: TPO) [27], in which each node (except the root) represents a tuple t_i, and an edge from t_i to t_j indicates that t_i is ranked higher than t_j (denoted $t_i \prec t_j$). Each path $\omega = t_1 \prec t_2 \prec \ldots \prec t_N$ represents a possible ordering of the underlying set of tuples T, and is associated with a probability $\Pr(\omega)$ [13]. Figure 1(b) shows the TPO corresponding to the pdf's of the tuples shown in Figure 1(a). Since processing a top-K query over uncertain data requires computing the orderings of the first K tuples, in order to answer such a query it suffices to build the subtree $\mathcal{T}_K$ of possible orderings up to depth K, compatibly with the pdf's of the tuple scores, since no other tuple may be relevant to answer the query. Building the complete tree $\mathcal{T}$ of depth N is thus unneeded, as the probabilities $\Pr(\omega_K)$ for each $\omega_K \in \mathcal{T}_K$ can be computed without knowing $\mathcal{T}$ and its probabilities, and thus much more efficiently. Indeed, $|\mathcal{T}|$ can be shown to increase exponentially with the number of tuples N and the score probability distribution spread δ, while $|\mathcal{T}_K|$ is typically slightly larger than K. Figure 2(a) shows an example TPO with 4 tuples; Figure 2(b) shows the same TPO when only the first $K = 2$ levels are considered.

Given a TPO $\mathcal{T}_K$, we propose four measures to quantify its level of uncertainty. These measures are based on the idea that the larger the number of orderings in $\mathcal{T}_K$ and the more similar their probabilities, the higher its uncertainty.

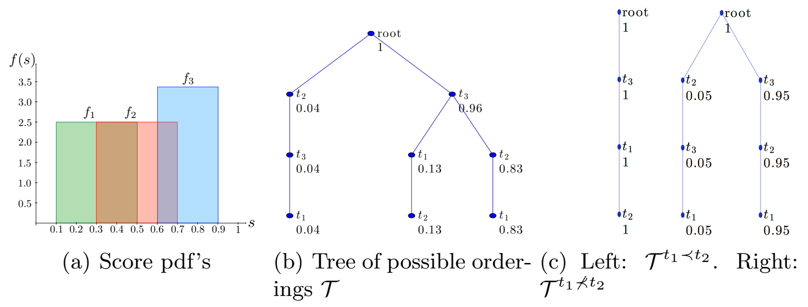

(a) Score pdf's (b) Tree of possible orderings $\mathcal{T}$ (c) Left: $\mathcal{T}^{t_1 \prec t_2}$. Right: $\mathcal{T}^{t_1 \not\prec t_2}$

Fig. 1. (a) Score pdf's for tuples t_1, t_2 and t_3; (b) their TPO $\mathcal{T}$; (c) sub-trees corresponding to the possible relative orders of t_1 and t_2. Each node is labeled with the probability of the corresponding prefix.

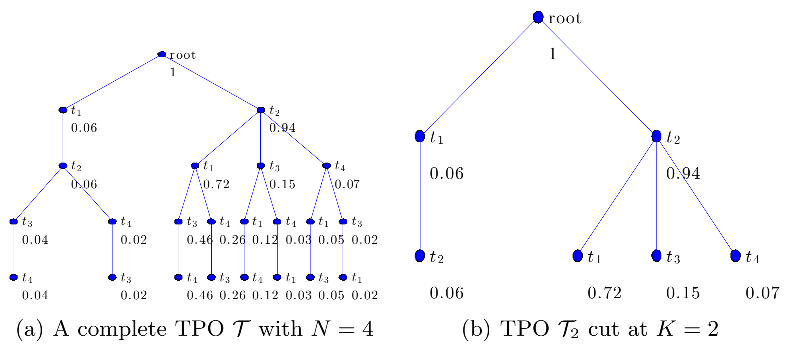

(a) A complete TPO $\mathcal{T}$ with $N = 4$ (b) TPO $\mathcal{T}_2$ cut at $K = 2$

Fig. 2. (a) Tree of possible orderings $\mathcal{T}$; (b) its cut at depth $K = 2$.

- **Entropy.** $U_H(\mathcal{T}_K)$ measures $\mathcal{T}_K$'s uncertainty via Shannon's entropy, based *only* on the probabilities of its leaves.
- **Weighted entropy.** $U_H(\mathcal{T}_K)$ is a weighted combination of entropy values at the first K levels of the TPO.
- **ORA.** $U_{\mathrm{ORA}}(\mathcal{T}_K)$ is based on the idea of comparing all the orderings in $\mathcal{T}_K$ with the Optimal Rank Aggregation (ORA) [26], which is a sort of median ordering in $\mathcal{T}_K$.
- **MPO.** $U_{\mathrm{MPO}}(\mathcal{T}_K)$ refers to another representative ordering, i.e., the Most Probable Ordering (MPO) [26].

We have observed that measures of uncertainty that take into account the structure of the tree in addition to ordering probabilities (i.e., U_{MPO}, U_{H_w} and U_{ORA}) are better suited than state-of-the-art measures (i.e., U_H) when used for our purposes, i.e., finding the best possible criteria to prune irrelevant branches from the tree, thus making it less uncertain so as to converge to a linear ordering.

3 Humans fighting uncertainty

The number of possible orderings in a TPO (i.e., paths from the root to a leaf in $\mathcal{T}$) depends on the number of tuples N and on the overlaps of their pdf's f_i, $i = 1, \ldots, N$, and can be very large even for small values of N.

We have two main ways of reducing uncertainty in $\mathcal{T}$ to quickly converge to the correct ordering: *i)* building only the first K levels of the TPO, thereby focusing on $\mathcal{T}_K$ instead of $\mathcal{T}$, and *ii)* defining crowd tasks for disambiguating the relative order of tuples in order to prune the TPO.

We consider crowd tasks expressed as questions of the form $q = t_i \overset{?}{\prec} t_j$, which compare t_i and t_j to determine which one ranks higher. Given a crowd worker's answer (i.e., either $t_i \prec t_j$ or $t_i \not\prec t_j$), we can prune from $\mathcal{T}_K$ all the paths disagreeing with the answer. As an example, Figure 1(c) shows two sub-trees derived from the tree in Figure 1(b) when either $t_1 \prec t_2$ or $t_1 \not\prec t_2$ (which are the possible answers to the question $t_1 \overset{?}{\prec} t_2$). Note that the leaf probabilities are always normalized so that they sum up to 1, i.e., each probability $\Pr(\omega)$ in a sub-tree $\mathcal{T}' \in \{\mathcal{T}^{t_i \prec t_j}, \mathcal{T}^{t_i \not\prec t_j}\}$ is recomputed as $\frac{\Pr(\omega)}{\sum_{\omega' \in \mathcal{T}'} \Pr(\omega')}$. As soon as the relative order of two objects t_i and t_j is known, the sub-tree that agrees with it becomes the new TPO.

The goal of the *Uncertainty Reduction* (UR) problem is to find a sequence of questions the answers to which allow pruning the TPO $\mathcal{T}_K$ so that a single ordering (the "real" ordering ω_r) remains. A UR algorithm is *optimal* if the sequence it finds is always minimal.

Theorem 1. *No deterministic UR algorithm is optimal.*

We consider two practical classes of algorithms: *i)* offline algorithms, which determine the questions a priori, before obtaining any answer from the crowd, and *ii)* online algorithms, whereby the questions are determined incrementally as the answers to previous questions arrive. These classes reflect two common situations in crowdsourcing markets: one where the interaction is limited to the publication of a batch of tasks, which is evaluated for acceptance as a whole; and one where the employer can inspect the outcome of crowd work as it becomes available and incrementally publish further tasks.

Since optimality is unattainable, we simply aim at maximizing the *expected* uncertainty reduction after pruning $\mathcal{T}_K$. The set $\mathcal{Q}_K$ of relevant questions to ask consists of those that compare tuples with an uncertain relative ordering, i.e., whose pdf's overlap. Let B denote the maximum number of questions (budget) that can be asked to the crowd workers. Our goal is then to select the best sequence of questions $\mathcal{Q}^* = \langle q_1^*, \ldots, q_B^* \rangle$ from $\mathcal{Q}_K$ that causes the largest amount of expected uncertainty reduction (with consequent reduction of the number of orderings in $\mathcal{T}_K$). An offline UR algorithm that always does so is said to be *offline-optimal.*

3.1 Offline question selection strategies

We present an offline-optimal algorithm, and two sub-optimal but faster alternatives.

Best-first search offline algorithm ($\texttt{A}^*\texttt{-off}$). This algorithm adapts the well-known A* search algorithm to exhaustively explore the space of question sets.

Theorem 2. $\texttt{A}^*\texttt{-off}$ *is offline-optimal.*

Top-B offline algorithm ($\texttt{TB-off}$). For each question $q \in \mathcal{Q}_K$, we compute the expected residual uncertainty $\mathcal{R}_q(\mathcal{T}_K)$ (i.e., the uncertainty of the TPO obtained by pruning $\mathcal{T}_K$ according to the answer collected after q is asked). Then, $\mathcal{Q}^*$ is defined as the set of B questions with the highest $\mathcal{R}_q(\mathcal{T}_K)$.

Conditional offline algorithm ($\texttt{C-off}$). This method iteratively selects one question at a time based on the previous selections. Let $\{q_1^*, \ldots, q_i^*\}$ be the first i selected questions ($\emptyset$ when $i = 0$). The $(i+1)$-th question q_{i+1}^* is selected by $\texttt{C-off}$ from $\mathcal{Q}_K \setminus \{q_1^*, \ldots, q_i^*\}$ so as to minimize $\mathcal{R}_{\langle q_1^*, \ldots, q_i^*, q_{i+1}^* \rangle}(\mathcal{T}_K)$, i.e., the residual uncertainty conditioned by the choice of the previously selected questions $q_1^*, \ldots, q_i^*$. The final output is thus $\mathcal{Q}^* = \{q_1^*, \ldots, q_B^*\}$.

3.2 Online question selection strategies

An online algorithm can determine the i-th question based on the answers to all the previously asked $i-1$ questions.

Best-first search online algorithm ($\texttt{A}^*\texttt{-on}$). An online UR algorithm that iteratively applies $\texttt{A}^*\texttt{-off}$ B times.

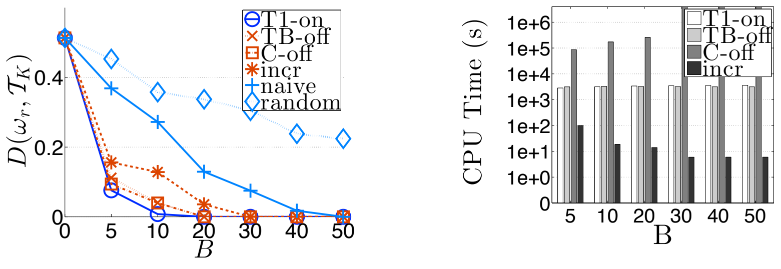
Top-1 online algorithm ($\texttt{T1-on}$). The $\texttt{T1-on}$ algorithm builds the sequence of questions $\mathcal{Q}^*$ iteratively: at each iteration, the algorithm selects the question that minimizes the expected residual uncertainty with budget $B = 1$, appends it to $\mathcal{Q}^*$ and asks it to the crowd. The TPO $\mathcal{T}_K$ is then updated to the sub-tree that agrees with the received answer. Early termination may occur if all uncertainty is removed with $|\mathcal{Q}^*| < B$.

3.3 Handling noisy workers

In a crowdsourcing scenario, the collected answers might be noisy. When a crowd worker's accuracy (i.e., the probability that his/her answer is correct) is less than 1, no pruning of $\mathcal{T}_K$ takes place, but the probabilities of the possible orderings are appropriately adjusted so as to reflect the collected answers. Let $\Pr(\omega)$ and $\Pr(\omega | ans_q = t_i \prec t_j)$ denote, respectively, the probability of the same ordering ω before and after the answer $t_i \prec t_j$ is received, respectively. Then, by Bayes' theorem

$$\Pr(\omega | ans_q = t_i \prec t_j) = \frac{\Pr(ans_q = t_i \prec t_j | \omega) \Pr(\omega)}{p \Pr(t_i \prec t_j) + (1-p) \Pr(t_i \not\prec t_j)},$$

where $\Pr(ans_q = t_i \prec t_j | \omega) = p$, if $t_i \prec t_j$ in ω; otherwise, $1-p$; similarly for $t_i \not\prec t_j$.



(a) Distance to real ordering ω_r (b) CPU time

Fig. 3. Performance of faster algorithms as budget B varies.

3.4 Incremental algorithm

The `incr` algorithm, builds the TPO $\mathcal{T}_K$ incrementally, one level at a time, by alternating tree construction with a round of n questions and tree pruning, thereby reducing the time needed to materialize trees containing a large number of orderings. The number n of questions posed at each round is between 1 and B, thus `incr` can be considered a hybrid between an online and an offline algorithm. Each TPO $\mathcal{T}_k$, $1 \leq k \leq K$, is built by adding one level to $\mathcal{T}_{k-1}$. We only build new levels if there are not enough questions to ask. Then, we select the best questions, pose them to the crowd, collect the answers and apply the pruning accordingly, until either the budget B is exhausted or the TPO is entirely built.

4 Experimental evaluation

The proposed algorithms have been evaluated experimentally against baselines that select questions either randomly or focusing on tuples with an ambiguous order: *i)* the `Random` algorithm returns a sequence of B questions chosen at random among all possible tuple comparisons in $\mathcal{T}_K$; *ii)* the `Naive` algorithm avoids irrelevant questions by returning a sequence of B questions chosen randomly from $\mathcal{Q}_K$. We observed that measures of uncertainty that take into account the structure of the tree in addition to ordering probabilities (i.e., U_{MPO}, U_{H_w}, U_{ORA}) perform better than state-of-the-art measures (i.e., U_H). The experiments show that the `T1−on` and `C−off` algorithms offer a good tradeoff between performance (Figure 3(a)) and costs (Figure 3(b)), with significant reductions of the number of questions wrt. the baselines, and nearly as good as with the A*-based algorithms, but at a fraction of the cost. The proposed algorithms have been shown to work also with non-uniform tuple score distributions and with noisy crowds [2]. Much lower CPU times are possible with the `incr` algorithm, with slightly lower quality (which makes `incr` suited for large, highly uncertain datasets).

5 Related work

Many works in the crowdsourcing area have studied how to exploit a crowd to obtain reliable results in uncertain scenarios, both online and offline [20, 21, 16, 7, 22]. However, none of the mentioned works applies to the top-K setting.

The problem of ranking tuples in the presence of uncertainty has been addressed in several works [27, 14, 13, 18], although not regarding the intervention of a crowd.

Recent works on uncertain top-K scenarios where questions comparing tuples in a set are asked to a crowd include [5, 17, 23, 4, 11], but none of these can be compared to our approach because they either assume no prior knowledge on the tuples or offer no guarantee that the extracted objects are the top K tuples.

Finally, we mention that several works [6, 4, 5, 12] use majority voting as a tool for aggregating multiple noisy answers and computing trusted labels, or other approaches [8, 25, 15, 19, 28, 9, 10, 3, 24, 29, 1].

6 Conclusions and future work

In this paper we have introduced Uncertainty Reduction (UR), which is the problem of identifying the minimal set of questions to be submitted to a crowd in order to reduce the uncertainty in the ordering of top-K query results. Since UR does not admit deterministic optimal algorithms, we have introduced two families of heuristics (offline and online, plus a hybrid thereof) capable of reducing the expected residual uncertainty of the result set. The proposed algorithms have been evaluated experimentally on both synthetic and real data sets, against baselines that select questions either randomly or focusing on tuples with an ambiguous order. The experiments show that offline and online best-first search algorithms achieve the best performance, but are computationally impractical. These trends are further validated on the real datasets. Future work will focus on generalizing the UR problem and heuristics to other uncertain data and queries, for example in skill-based expert search, where queries are desired skills and results contain sequences of people sorted based on their topical expertise and skills can be endorsed by community peers.

Acknowledgments. Support is acknowledged from the EC's FP7 "Smart H2O" project (`http://smarth2o-fp7.eu/`) and the "Proactive" project (`http://www.proactiveproject.eu/`), funded by the EU and Regione Lombardia.

References

1. A. Anagnostopoulos et al. The importance of being expert: Efficient max-finding in crowdsourcing. In *SIGMOD*, 2015.
2. E. Ciceri et al. Crowdsourcing for top-k query processing over uncertain data. *IEEE Trans. Knowl. Data Eng.*, 28(1):41–53, 2016.
3. N. N. Dalvi et al. Aggregating crowdsourced binary ratings. In *WWW*, pages 285–294, 2013.

4. A. Das Sarma et al. Crowd-powered find algorithms. In *ICDE*, pages 964–975. IEEE, 2014.
5. S. B. Davidson et al. Top-k and clustering with noisy comparisons. *ACM Trans. Database Syst.*, 39(4):35:1–35:39, 2014.
6. C. Gokhale et al. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*, pages 601–612, 2014.
7. S. Guo et al. So who won?: Dynamic max discovery with the crowd. In *SIGMOD '12*, pages 385–396, 2012.
8. F. C. Heilbron and J. C. Niebles. Collecting and annotating human activities in web videos. In *ICMR*, page 377, 2014.
9. P. G. Ipeirotis et al. Quality management on amazon mechanical turk. In *SIGKDD workshop on human computation*, pages 64–67. ACM, 2010.
10. M. Joglekar et al. Comprehensive and reliable crowd assessment algorithms. *ICDE*, 2015.
11. H. Kaplan, I. Lotosh, T. Milo, and S. Novgorodov. Answering planning queries with the crowd. *PVLDB*, 6(9):697–708, 2013.
12. S. K. Kondreddi et al. Combining information extraction and human computing for crowdsourced knowledge acquisition. In *ICDE*, pages 988–999, 2014.
13. J. Li and A. Deshpande. Ranking continuous probabilistic datasets. *PVLDB*, 3(1-2):638–649, 2010.
14. J. Li et al. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1):502–513, 2009.
15. X. Liu et al. CDAS: A crowdsourcing data analytics system. *PVLDB*, 5(10):1040–1051, 2012.
16. A. Marcus et al. Crowdsourced databases: Query processing with people. In *CIDR '11*, pages 211–214, 2011.
17. A. Marcus et al. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, Sept. 2011.
18. L. Mo et al. Cleaning uncertain data for top-k queries. In *ICDE, 2013*, pages 134–145. IEEE, 2013.
19. Q. V. H. Nguyen et al. Minimizing Efforts in Validating Crowd Answers. In *The 2015 ACM SIGMOD/PODS Conference*, 2015.
20. A. Parameswaran et al. Human-assisted graph search: It's okay to ask questions. *PVLDB*, 4(5):267–278, 2011.
21. A. Parameswaran et al. Crowdscreen: Algorithms for filtering data with humans. In *SIGMOD '12*, pages 361–372, 2012.
22. A. Parameswaran and N. Polyzotis. Answering queries using humans, algorithms and databases. In *CIDR '11*.
23. V. Polychronopoulos et al. Human-powered top-k lists. In *WebDB*, pages 25–30, 2013.
24. V. C. Raykar and S. Yu. Ranking annotators for crowdsourced labeling tasks. In *NIPS*, pages 1809–1817, 2011.
25. J. Redi and I. Povoa. Crowdsourcing for rating image aesthetic appeal: Better a paid or a volunteer crowd? In *ACM MM*, CrowdMM '14, pages 25–30. ACM, 2014.
26. M. Soliman et al. Ranking with uncertain scoring functions: semantics and sensitivity measures. In *SIGMOD '11*, pages 805–816, 2011.
27. M. Soliman and I. Ilyas. Ranking with uncertain scores. In *ICDE '09.*, pages 317 –328, 2009.
28. P. Venetis and H. Garcia-Molina. Quality control for comparison microtasks. In *Int. Workshop on Crowdsourcing and Data Mining*, pages 15–21. ACM, 2012.
29. C. J. Zhang et al. Reducing uncertainty of schema matching via crowdsourcing. *PVLDB*, 6(9):757–768, July 2013.