International Journal of Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.06,Issue.26
August-2017,
Pages:5109-5112

www.ijsetr.com

Minimizing Crowd using Top-K Query Processing over Uncertain Data

KAMMA PUJITHA[1], OM PRAKASH SAMANTRAY[2], T. SUBBA REDDY[3]

[1]PG Scholar, Dept of CSE, Narasaraopeta Engineering College, JNTUK, Kakinada, AP, India.
[2]Associate Professor, Dept of CSE, Narasaraopeta Engineering College, JNTUK, Kakinada, AP, India,
Email: Om.prakash02420@gmail.com.
[3]Assistant Professor, Dept of CSE, Narasaraopeta Engineering College, JNTUK, Kakinada, AP, India,
Email: subbareddynec@gmail.com.

Abstract: Top-k processing in uncertain databases is semantically and computationally different from traditional top-k processing. We introduce new probabilistic formulations for top-k queries. When data ambiguity cannot be reduced algorithmically, Crowd sourcing proves a viable approach, which consists in posting tasks to humans and harnessing their judgment for improving the confidence about data values or relationships. We propose efficient algorithms to compute these vectors. We also extend the semantics and algorithms to the scenario of score ties, which is not dealt with in the previous work in the area. 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. Our experiments show the efficiency of our techniques under different data distributions with orders of magnitude improvement over na¨ıve materialization of possible worlds.

Keywords: Top-K, Distribution, Uncertain Data, Typical. User/Machine Systems, Query Processing.

I. INTRODUCTION

It is well-known that crowd sourcing works best when tasks can be broken down into very simple pieces. An entire schema matching may be too large a grain for a crowd each individual may have small quibbles with a proposed matching, so that a simple binary question on the correctness of matching may get mostly negative answers with each user declaring it less than perfect [1]. On the other hand, asking open-ended questions is not recommended for a crowd, because it may be difficult to pull together a schema matching from multiple suggestions [2].In the well-known class of applications commonly referred to as "top-K queries" [3], 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 [4].The goal of this paper is to define and compare task selection policies for uncertainty reduction via crowd sourcing, with emphasis on the case of top-K queries. The expected residual uncertainty of the result, possibly leading to a unique ordering of the top K results. The main contributions of the paper are as follows [5]:

1. We formalize a framework for uncertain top-K query processing, adapt to its existing techniques for computing the possible orderings, and introduce a procedure for removing unsuitable orderings, given new knowledge on the relative order of the objects [6].
2. We define and contrast several measures of uncertainty, either agnostic dependent on the structure of the orderings.
3. We formulate the problem of Uncertainty Resolution (UR) in the context of top-K query processing over uncertain data with crowd support.The UR problem amounts to identifying the shortest sequence of questions that, when submitted to the crowd, ensures the convergence to a unique, or at least more determinate [7],
4. We introduce two families of heuristics for question selection: offline, where all questions are selected prior to interacting with the crowd, and online, where crowd answers and question selection can intermix.
5. We propose an algorithm that avoids the materialization of the entire space of possible orderings to achieve even faster results.
6. We conduct an extensive experimental evaluation of several algorithms on both synthetic and real datasets, and with a real crowd, in order to assess their performance and scalability[8].

II. SYSTEM OVERVIEW

We give an overview of the architecture for task driven crowd-selection. We illustrate the architecture of our task-driven crowd-selection system [9]. The core component of our system is crowd manager. The main functionalities of crowd manager are latent skill inference for workers and choose the right crowd for given crowdsourced tasks. The

Copyright @ 2017 IJSETR. All rights reserved.

crowd model is stored in the crowd databases which support crowd insertion, crowd update and crowd retrieval. The red lines show the process of latent skill inference for workers as well as build latent category space for tasks, which is based on resolved tasks with feedback scores. The crowd databases are then updated. Given a coming crowdsourced task, the crowd manager first projects it into the built latent category space. Next, the crowd manager returns the workers online as the candidate crowd for this task [10]. The crowd manager then ranks the workers who are skilled in this task. The top ranked workers are chosen for solving the task. After that, the task dispatcher distributes this task to the selected workers. Finally, the system keeps collect the answers return by the selected workers. In summary, our task-driven crowd-selection system can automatically ask the right crowd to process the crowdsourced tasks. The system is able to incrementally project the coming tasks to the existing latent category space such that the workers can be chosen in the real time. In the following sections, we present the idea and the methods of implementing this task-driven crowd-selection system [11]. The system overview is depicted in figure 1.

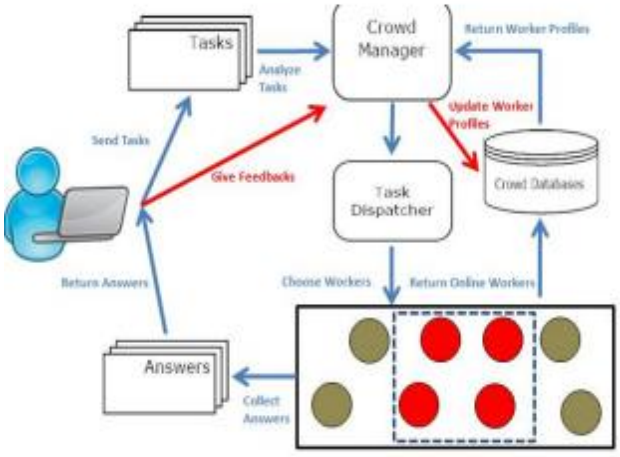


Fig 1. System overview.

III. RELATED WORK

Many works in the crowd sourcing area have studied how to exploit a crowd to obtain reliable results in uncertain scenarios. In [12], binary questions are used to label nodes in a directed acyclic graph, showing that an accurate question selection improves upon a random one. Similarly [12] and [13] aim to reduce the time and budget used for labeling objects in a set by means of an appropriate question selection. Instead, [14] proposes an online question selection approach for finding the next most convenient question so as to identify the highest ranked object in a set. A query language where questions are asked to humans and algorithm is described in [15]; humans are assumed to always answer correctly, and thus each question is asked once. All these works do not apply to a top-K setting and cannot be directly compared to our work. Active learning is a form of supervised machine learning, in which a learning algorithm is able to interact with the workers (or some other information source) to obtain the desired outputs at new data points [16]. In particular, [17, 18] proposed active learning methods specially designed for crowd-sourced databases. Our work is essentially different from active learning in two perspectives: (1) the role of workers in active learning is to improve the learning algorithm in this paper the involvement of workers is to reduce the uncertainty of given matching. (2) The uncertainty of answers are usually assumed to be given before generating any questions; in this paper, the uncertainty of answers has to be considered after the answers are received, since we cannot anticipate which workers would answer our questions. To our best knowledge, there is no algorithm in the field of active learning can be trivially applied to our problem.

IV. TREE OF POSSIBLE ORDERINGS (TPO)

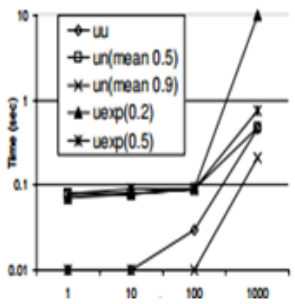
Building the TPO A method for constructing a TPO T was proposed in [19]. Let T be a table containing the tuples with uncertain score {t1,...,tN }. In order to build the tree, a dummy root node is created. Then, the sources (i.e., tuples ti 2 T such that there does not exist any tj 2 T such that lj>ui) are extracted from T and attached as children of the root. Next, each extracted source is used as a root for computing the next level of the tree. The asymptotic time complexity of building the tree up to level K is O(KN2). Finally, the probability Pr(!) of any ordering ! in the tree can be computed, e.g., with the generating functions technique [20] with asymptotic time complexity O(N2), or via Monte Carlo sampling. In particular[21], when considering two topples ti and tj in the full TPO T (i.e., when K = N), each path in T agrees either with ti tj or with ti 6 tj. Thus, T can be partitioned into two sub-trees.

A. Algorithm

Online question selection strategies an online algorithm has the ability to determine the i-th question based on the answers collected for all the previously asked i 1 questions. Differently from the offline case, the output of an online algorithm is treated as a sequence and not as a set, since each received answer may influence the choice of the next question, and thus the order matters. Best-first search online algorithm (A↞on). An online UR algorithm can be obtained by iteratively applying A↞off B times. At the i-th step, 1 □ i □ B, A↞on identifies the i-th question q↞ i in its output and asks it to the worker, thus obtaining the answer ans!(q↞ i). Question q↞ i is simply the first element of the sequence Q↞ i returned by A↞off for the TPO T hans!(q↞ 1),...,ans!(q↞ i1)i K with budget (Bi+1), where ! is the real ordering and q↞ 1 ,...,q↞ i1 are the previously selected questions (initially, A↞off is applied on TK with budget B and the first question in its output is chosen as q↞ 1). Intuitively, each step chooses the most promising question q↞ i within the horizon of the remaining B i + 1 questions to be asked based on the current knowledge of !. Note that, as new answers arrive, the next most promising questions might no longer coincide with the rest of the previously planned sequence Q↞i . Being based on A↞off, A↞on is costly. Thus, we also consider a simpler but more efficient online algorithm [22].

Algorithm 1: Top-1 online algorithm (T1on)
Input: TPO TK, Budget B
Output: Optimal sequence of questions Q←
Environment: Underlying real ordering!
1) Q←:= ;
2) For i := 1 to B
3) If |TK| = 1 then break;
4) q← i := arg minq2QK\Q←Rhqi(TK);
5) Q← := Q←hq← i i;
6) Ask q← i to the crowd and collect the answer ans!(q← i)
7) TK := T ans!(q← i) K ;
8) return Q←;

The time and scan depth of Indep U-Topk, respectively, while the time and scan depth of IndepU-kRanks, respectively with k values up to 1000. The best case for algorithm is to find highly probable tuples frequently in the scoreranked stream. The counter scenario applies to uexp(0.2) whose mean value forces confidence to decay relatively fast leading to small number of highly probable tuples. IndepU-Topk execution time is under 10 seconds for all data distributions, and it consumes a maximum of 15,000 tuples for k=1000 under exponentially-skewed distribution. The maximum scan depth of IndepU-kRanks is 4800 tuple, however the execution time is generally larger. This can be attributed to the design of algorithm where bookkeeping and candidate maintenance operations are more extensive in IndepU-kRanks.



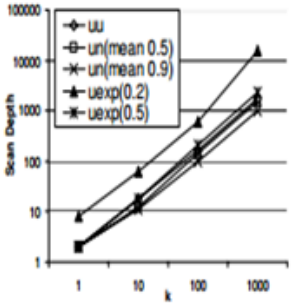


Fig 2. IndepU-Topk time **Figure 3. IndepU-Topk depth.**

We generated bivariate gaussian data over score and confidence, and controlled correlation coefficient by adjusting bivariate covariance matrix. Positive correlations result in large savings since in this case high scored tuples are attributed with high confidence, which allows reducing the number of needed-to-see tuples to answer uncertain top–k queries.

V. CONCLUSIONS

We have introduced Uncertainty Resolution (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. We formulated the problem as a state space search, Our processing framework leverages existing storage and query processing techniques and can be easily integrated with existing DBMSs. The proposed algorithms have been shown to work also with non-uniform tuple score distributions and with noisy crowds. Much lower CPU times are possible with the online algorithm, with slightly lower quality. These trends are further validated on the real datasets. We then develop variation algorithm that transforms the probabilistic inference into a standard optimization problem, which can be solved efficiently. We also devise an online crowd-selection algorithm that projects the coming tasks into the existing latent category space and choose the highly skilled workers for the tasks. We validate the performance of our algorithm based on the data collected from three well-known crowd sourcing applications: Quora, Yahoo ! Answer and Stack Overflow.

VI. FUTURE WORK

We will try to fold this into a more realistic and more complete model of worker error rates. We have also ignored more complex worker incentives, including boredom and thoughtless answer selection, as being standard across crowd sourcing platforms. We propose to provide the score distribution of top-k vectors and c-Typical-Topk answers to applications and devise efficient algorithms to cope with the computational challenges. We also extend the work to score ties. Experimental results verify our motivation and our approaches

VII. REFERENCES

[1] M. Allahbakhsh et al. Quality control in crowdsourcing systems: Issues and directions. IEEE Internet Comp., 17(2): 76– 81, 2013.
[2] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. Commun. ACM, 54(4):86–96, 2011.
[3] X. L. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. VLDB J., 18(2):469–500, 2009.
[4] D. Barbara, H. Garcia-Molina, and D. Porter. The management of probabilistic data. IEEE Transactions on Knowledge and Data Engineering, 4(5):487–502, 1992.
[5] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. Uldbs: Databases with uncertainty and lineage. In VLDB, 2006.
[6] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In SIGMOD, 2003
[7] J. Guo, S. Xu, S. Bao, and Y. Yu. Tapping on the potential of q&a community by recommending answer providers. In CIKM, pages 921–930. ACM, 2008.
[8] S. Guo, A. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In Proceedings of SIGMOD, pages 385–396. ACM, 2012
[9] T. Hofmann. Probabilistic latent semantic indexing. In SIGIR, pages 50–57. ACM, 1999.
[10] H. Kaplan, I. Lotosh, T. Milo, and S. Novgorodov. Answering planning queries with the crowd.
[11] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. Cdas: a crowdsourcing data analytics system. Proceedings of PVLDB, 5(10):1040–1051, 2012.
[12] A. Parameswaran et al. Human-assisted graph search: It's okay to ask questions. PVLDB, 4(5):267–278, 2011.
[13] A. Parameswaran et al. Crowdscreen: Algorithms for filtering data with humans. In SIGMOD '12, pages 361–372, 2012.
[14] A. Parameswaran and N. Polyzotis. Answering queries using humans, algorithms and databases. In CIDR '11.

[15] V. Polychronopoulos et al. Human-powered top-k lists. In WebDB, pages 25–30, 2013.
[16] A. G. Parameswaran and N. Polyzotis. Answering queries using humans, algorithms and databases. In CIDR, pages 160–166, 2011.
[17] A. G. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. PVLDB, 4(5):267–278, 2011.
[18] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernandez, and ´ R. Fagin. Translating web data. In VLDB, pages 598–609, 2002.
[19] M. Soliman and I. Ilyas. Ranking with uncertain scores. In ICDE '09., pages 317 –328, 2009.
[20] P. Venetis et al. Max algorithms in crowdsourcing environments. In WWW, pages 989–998, 2012.
[21] P. Venetis and H. Garcia-Molina. Quality control for comparison microtasks. In International Workshop on Crowdsourcing and Data Mining, pages 15–21. ACM, 2012.
[22] S. E. Whang et al. Question selection for crowd entity resolution. VLDB, 2013.
[23] H. Yu et al. Enabling ad-hoc ranking for data retrieval. ICDE, 2005.

Author's Profile:

K.Pujitha studying M.Tech (CSE) in Narasaraopeta Engineering College, NRT. Completed B.Tech (CSE) in Narasaraopeta Engineering College, NRT.

Om Prakash Samantray is presently working as Associate Professor in dept. of CSE in Narasaraopeta Engineering College. Email id:Om.prakash02420@gmail.com.

Subba Reddy is presently working as Assistant Professor in dept. of CSE in Narasaraopeta Engineering College, Email id:subbareddynec@gmail.com.