Crowdsourced Data Management: Overview and Challenges

Guoliang Li*, Yudian Zheng#, Ju Fan†, Jiannan Wang‡, Reynold Cheng#
*Department of Computer Science, Tsinghua University, China
#Department of Computer Science, The University of Hong Kong, Hong Kong, China
†DEKE Lab & School of Information, Renmin University of China, China
‡School of Computing Science, Simon Fraser University, Canada
liguoliang@tsinghua.edu.cn, ydzheng2@cs.hku.hk, fanj@ruc.edu.cn, jnwang@sfu.ca, ckcheng@cs.hku.hk

ABSTRACT

Many important data management and analytics tasks cannot be completely addressed by automated processes. Crowdsourcing is an effective way to harness human cognitive abilities to process these computer-hard tasks, such as entity resolution, sentiment analysis, and image recognition. Crowdsourced data management has been extensively studied in research and industry recently. In this tutorial, we will survey and synthesize a wide spectrum of existing studies on crowdsourced data management. We first give an overview of crowdsourcing, and then summarize the fundamental techniques, including quality control, cost control, and latency control, which must be considered in crowdsourced data management. Next we review crowdsourced operators, including selection, collection, join, top-k, sort, categorize, aggregation, skyline, planning, schema matching, mining and spatial crowdsourcing. We also discuss crowdsourcing optimization techniques and systems. Finally, we provide the emerging challenges.

Keywords

Crowdsourcing; Data Management; Crowdsourcing Optimization

1. INTRODUCTION

There are many computer-hard tasks, such as entity resolution [93, 95, 91, 96], sentiment analysis [113, 61, 69], and image recognition [98, 82, 103], which cannot be effectively solved by existing machine algorithms. Benefited from using human cognitive ability, crowdsourcing has been emerged as an effective paradigm to address such tasks by utilizing hundreds of thousands of ordinary workers (i.e., the crowd) [92, 25, 47, 100, 60, 8, 11]. Thanks to public crowdsourcing platforms, such as Amazon Mechanical Turk (AMT) [1], CrowdFlower [2], the access to crowdsourcing resources has become easier. As reported in [3], more than 500K workers from 190 countries have performed tasks on AMT [1].

Over the past few years, crowdsourcing has become an active area in the data management community from both research and industry (see a survey [57] and a book [67]). There are several important problems in crowdsourced data management as shown in Figure 1. Typically, in a crowdsourcing platform (e.g., AMT [1]), there are two types of users, called *"workers"* and *"requesters"*. Requesters publish tasks on a crowdsourcing platform, while workers perform tasks and return the results. Suppose a requester has an entity resolution problem to solve, which aims to find the same entity from 10K products. The requester needs to first perform *"task design"*, e.g., designing the user interface of a task (e.g., showing a pair of products and asking the crowd to choose between "the same entity" and "different entities"), and set up some properties of the tasks (e.g., the price of a task, the number of workers to answer a task, the time duration to answer a task). Then the requester publishes the tasks to the platform, and collects the answers from the crowd. In this process, three core crowdsourced data management techniques must be considered: *"quality control"*, *"cost control"*, and *"latency control"*. Quality control aims to generate high-quality answers from workers' (possibly noisy) answers, by characterizing a worker's quality and aggregating workers' answers [39, 61, 105, 16, 110, 53, 14, 52, 100, 50, 49, 13, 47, 100, 87, 79, 71, 27, 109, 108, 43, 44, 43, 44]. Cost control focuses on how to reduce human costs while still keeping good result quality [93, 95, 22, 96, 99, 91, 48, 99, 89, 36, 69, 39, 19, 77, 26, 104, 54, 75, 112, 111]. Latency control exploits how to reduce the latency by modeling workers' latency and estimating workers' arrival rates [35, 31, 41]. Note there are trade-offs among quality, cost, and latency, and existing studies focus on how to balance them, e.g., optimizing the quality given a fixed cost, reducing the latency given a fixed cost, minimizing the cost under latency and quality constraints, etc.

In addition, it is rather inconvenient for the requester to interact with the crowdsourcing platforms, because the platforms require requesters to set parameters and even write codes. To encapsulate the process of interacting with the crowd, we need to design crowdsourced operators and systems. For example, entity resolution can use a crowdsourced join to find objects referring to the same entity. In data extraction, we need to use crowdsourced selection to select relevant data. In subjective comparison scenarios, we need to use crowdsourced sort to rank the results. All database operators have been studied in crowdsourcing, e.g., Selection [73, 72, 81, 103], Collection [86, 76], Join [93, 95, 91, 96, 36, 99], Topk/Sort [39, 19, 77, 26, 39, 21, 104, 54], Categorize [75], Aggregation [39, 88, 66, 42, 21], Skyline [62, 63, 37], Planning [51, 64, 106, 83, 84], Schema Matching [105, 70, 28], Mining [11, 12, 9, 10], and Spatial Crowdsourcing [85]. In addition, several crowdsourcing database systems (e.g., CrowdDB [33], Deco [74], Qurk [68]) have been developed for query plan generation and optimization.

• **Tutorial Structure.** The 3 hours' tutorial is split into 2 sections.

In the first section (1.5 hours), we first give an overview of crowdsourcing (0.5 hour), including motivation of crowdsourcing (see Section 2), basic crowdsourcing concepts (e.g., workers, requesters), crowdsourcing platforms (e.g., AMT [1], CrowdFlower [2]), crowdsourcing workflow, and crowdsourcing applications. Then we talk

SIGMOD'17, May 14-19, 2017, Chicago, IL, USA

DOI: http://dx.doi.org/10.1145/3035918.3054776

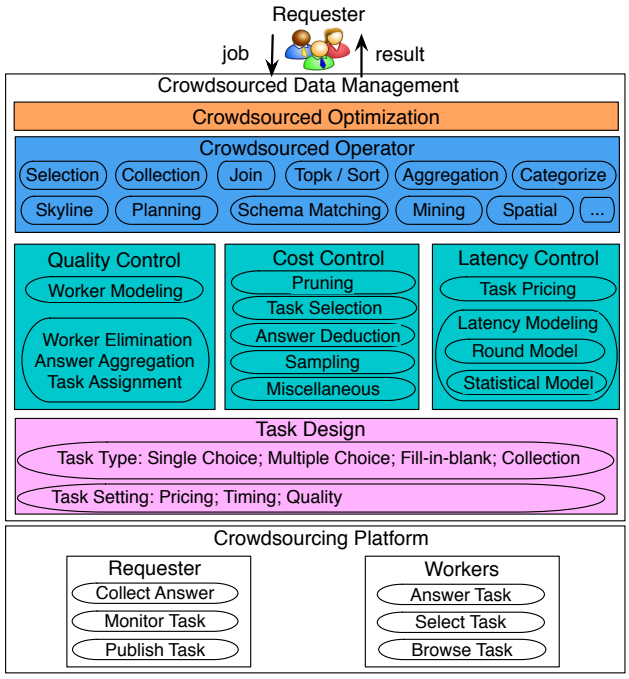

Figure 1: Overview of Crowdsourced Data Management.

about three fundamental techniques in crowdsourcing (1 hour), i.e., quality control, cost control and latency control (see Section 3). Specifically, we summarize the factors considered in each case and discuss the pros and cons of different factors in the techniques.

In the second section (1.5 hours), we first discuss optimization techniques in crowdsourced data management by balancing the trade-offs among the three fundamental factors (20 min) and then introduce crowdsourced database systems (20 min), e.g., CrowdDB [33], Deco [74], Qurk [68], including their design, query plan generations and optimizations (see Section 4). Then we discuss how existing studies address crowdsourced operators (20 min), e.g., *selection, collection, join, topk, sort, categorize, aggregation, skyline, planning, schema matching, mining* and *spatial crowdsourcing*, in order to optimize cost, quality or latency (see Section 4.2). Finally we provide emerging challenges (15 min) in Section 5. We leave 15 min for Q&A to interact with the tutorial audience.

Tutorial Audience. The intended audience include all SIGMOD attendees from both research and industry communities. We will not require any prior background knowledge and a basic understanding of database (e.g., selection, join) will be helpful.

Differences from Existing Tutorials. Although there are existing crowdsourcing tutorials (e.g., in VLDB'16 [7], VLDB'15 [34], ICDE'15 [18], VLDB'12 [24]), most of them focus on a specific part of crowdsourcing. VLDB'16 [7] investigates human factors involved in task assignment and completion. VLDB'15 [34] focuses on truth inference in quality control. ICDE'15 [18] reviews individual crowdsourcing operators, crowdsourced data mining and social applications. VLDB'12 [24] introduces crowdsourcing platforms and discusses design principles for crowdsourced data management. Compared with these tutorials, we will summarize an overview of a wide spectrum of work on crowdsourced data management, with a special focus on the fundamental techniques for controlling quality, cost and latency. We will also introduce crowdsourcing systems and operators, including the works published very recently [27, 37, 40, 41, 50, 65, 71, 89, 90, 96, 108, 110, 111, 112, 113], which can give the audience an update of the crowdsourcing techniques. Moreover, we will also provide emerging challenges.

2. CROWDSOURCING OVERVIEW

Crowdsourcing Workflow. Suppose a requester (e.g., Microsoft product team) has a set of computer-hard tasks (e.g., entity resolution tasks that find the objects referring to the same entity). As traditional entity-resolution algorithms are still far from perfect [93], the requester wants to utilize crowdsourcing to achieve high quality. To this end, the requester first designs the tasks (e.g., a task for every pair of objects that asks workers to check whether the two objects refer to the same entity) and sets the price of each task. Then the requester publishes their tasks on a crowdsourcing platform e.g., AMT [1]. Workers who are willing to perform such tasks accept the tasks, answer them and submit the answers back to the platform. The platform collects the answers and reports them to the requester. If a worker has accomplished a task, the requester who publishes the task can approve or disapprove the worker's answers. The approved workers will get paid from the requester. As the crowd has contextual knowledge and cognitive ability, crowdsourced entity resolution can improve the quality [93, 95, 96].

Applications. There are many successful applications that utilize crowdsourcing to solve computer-hard tasks. For example, Von Ahn et al. digitized newspapers for The New York Times by getting Internet users to transcribe words from scanned texts [92]. Their method achieved accuracy exceeding 99% and has transcribed over 440 million words. As another example, Eiben et al. utilized a game-driven crowdsourcing method to enhance a computationally designed enzyme [25]. Crowdsourcing can also benefit data management applications, such as data cleaning [94, 76], data integration [47, 100, 60], and knowledge construction [8, 11].

Task Design. There are several important task types that are widely used in real-world crowdsourcing platforms. *(1) Single-Choice Task.* Workers select a single answer from multiple options. For example, in sentiment analysis, given a review, it asks workers to assess the sentiment of the review (options: Positive, Neutral, Negative). *(2) Multiple-Choice Task.* Workers select multiple answers from multiple options. For example, given a picture, workers select the objects that appear in the picture (Options: Monkey, Tree, Banana, Beach, Sun). *(3) Fill-in-blank Task.* Workers need to fill-in-blank for an object. For example, given a professor, workers are asked to fill the university of the professor. *(4) Collection Task.* Workers need to collect information, e.g., collecting 100 US universities. Single/multiple choice tasks are closed-world tasks and the workers only need to select from given options, while fill/collection tasks are open-world tasks and the workers can provide any results.

Task Setting. The requester also needs to determine some task settings based on his/her requirements. *(1) Pricing.* The requester needs to price each task, usually varying from a few cents to several dollars. Note that pricing is a complex game-theoretic problem. Usually, high prices can attract more workers, thereby reducing the latency; but paying more does not always improve answer quality [31]. *(2) Timing.* The requester can set time constraints for a task. For each task, the requester can set the time bound (e.g., 10 minutes) to answer it, and the worker must answer it within this time bound. *(3) Quality Control.* The requester can select the quality-control techniques provided by the crowdsourcing platform, or design his/her own methods (see Section 3).

Crowdsourcing Platforms. We also introduce existing crowdsourcing platforms, e.g., AMT [1], CrowdFlower [2], ChinaCrowd [4], and discuss their features, differences, and functionalities (including whether the requester can control task assignment, whether the requester can select specific workers).

3. FUNDAMENTAL TECHNIQUES

We review three fundamental techniques, i.e., quality control, cost control and latency control, and discuss their trade-offs.

3.1 Quality Control

Crowdsourcing may yield relatively low-quality results, e.g., a malicious worker may intentionally give wrong answers, and a worker may have different levels of expertise. To achieve high quality, we need to tolerate errors and infer high-quality results from noisy answers. There are various factors to consider in quality control.

The first is to characterize a worker's quality (called *"worker modeling"*) [39, 61, 32, 105, 16, 110, 53, 14, 52, 100, 50, 49, 13, 47, 100, 87, 79, 71, 27, 109, 108, 43, 44, 43, 44]. Some initial works [22, 101, 52, 58] model each worker as a single value $\in [0, 1]$ (called "*worker probability*"), capturing the probability that the worker will answer tasks correctly. An extension of worker probability model is to introduce the *confidence interval* into the probability, which to some extent models the variance of a worker's answering behavior. There are also works [13, 60] that model each worker's quality as a *confusion matrix*, which represents the worker's ability in answering different labels, e.g., an optimistic worker tends to answer "*positive*" to a sentiment analysis task even if it holds "*neutral*" sentiment. Some recent works [97, 27, 111, 65] model the *diverse skills* of a worker, which captures the worker's answering abilities for different domains. For example, a *sports* fan while paying no attention to *politics* might answer tasks related to *sports* more correctly compared with those related to *politics*.

Then in order to infer worker's quality, there are three ways. The first is to adopt qualification test, and when a worker comes, he/she is required to answer qualification test (containing tasks with known ground truth) before the worker can answer real tasks. Then based on the estimated quality of workers, we can eliminate/block the low-quality workers (called *"worker elimination"*) to answer tasks [78, 47, 66]. The second is to use golden tasks, which mix tasks with ground truth into the tasks assigned to workers. Different from qualification test, workers do not know which are golden tasks, and they do not perform a test at their first come. The two approaches both require the ground truth of a subset of tasks to be known in advance. The third computes each worker's quality in an unsupervised manner, i.e., without requiring ground truth to be known [16, 56, 61, 113, 13, 47, 110, 101, 65, 97]. The basic principle is two fold: (1) the workers who have answered tasks correctly tend to have high qualities; (2) the answers of tasks given by high quality workers tend to be the true answers. Following these two intuitive principles, existing works [13, 47, 101, 60, 65, 97] often regard workers' qualities and tasks' truth as two sets of parameters, and follow an iterative approach to update them until convergence. They are often called "*Truth Inference*" methods, since not only workers' qualities are computed, but also each task's truth is obtained. We summarize different factors in existing truth inference methods in Table 1. We list the task models (e.g., modeling the difficulty of tasks, the domains in tasks), the worker models as discussed above, and the techniques used. Majority voting directly computes the truth by the answers of majority workers, and other methods adopt an iterative approach. Some of them design an optimization function with desired goals [59, 58, 114]; others adopt the probabilistic graphical model [13, 65, 47, 101, 97], where the most classical framework is Expectation-Maximization [13, 47, 23].

There are also some works that actively assign tasks to appropriate workers (called *"task assignment"*) [109, 108, 110, 16, 61, 15, 113, 27]. Their basic ideas are to gradually and judiciously assign the tasks to the workers that have high probability to correctly answer the tasks. Following this way, the quality can be improved within a fixed amount of budget.

3.2 Cost Control

The crowd is not free, and a large number of tasks would result in high costs. For example, in entity resolution, if there are 10K objects, there will be about 50M pairs. Even if the price per pair is \$0.01, it still takes lots of money. Therefore, a big challenge for crowdsourced data management is cost control. That is, how to reduce human cost while still keeping good result quality.

There are several cost-control techniques. The first is *"pruning"*, which first uses computer algorithms to remove unnecessary tasks and then utilizes the crowd to answer only the useful ones [93, 95, 22, 96, 99, 91]. The second is *"task selection"*, which prioritizes tasks with high benefits for crowdsourcing [48, 99, 89, 36, 69, 39, 19, 77, 26, 104, 54, 75]. The third is *"answer deduction"*, which crowdsources a subset of tasks and based on the answers collected from the crowd, deduces the results of other tasks [95, 91, 96, 38, 51, 106, 9]. The fourth is *"sampling"*, which samples a subset of tasks to crowdsource [66, 42, 94]. There are also *"miscellaneous"* ways [66, 86], which try to leverage well-designed task interfaces and pay-as-you-go approach to reduce costs. These cost-control techniques can also be used together. For example, we can first prune many tasks and then utilize the task-selection idea to select tasks. There are some specialized cost control techniques designed to optimize cost for a particular operator. For example, Marcus et al. [66] proposed a count-based user interface to reduce the number of tasks required for crowdsourced count.

Table 1: Comparisons of Truth Inference Methods.

Method	Task Modeling	Worker Modeling	Techniques
Majority Voting	No Model	No Model	Direct Computation
ZenCrowd [22]	No Model	Worker Probability	Graphical Model
GLAD [101]	Task Difficulty	Worker Probability	Graphical Model
D&S [13]	No Model	Confusion Matrix	Graphical Model
Minimax [114]	No Model	Diverse Skills	Optimization
BCC [55]	No Model	Confusion Matrix	Graphical Model
CBCC [87]	No Model	Confusion Matrix	Graphical Model
LFC [80]	No Model	Confusion Matrix	Graphical Model
CATD [58]	No Model	Confidence Interval	Optimization
CRH [59]	No Model	Worker Probability	Optimization
Multi [97]	Latent Topics	Diverse Skills	Graphical Model
KOS [52]	No Model	Worker Probability	Graphical Model
Mean Field [60]	No Model	Confusion Matrix	Graphical Model

3.3 Latency Control

Crowd answers may incur excessive latency for several reasons. For example, workers may be distracted or unavailable, the tasks may not be appealing to enough workers, or the tasks might be difficult for most workers. If the requester has a time constraint, it is important to control latency. There are several strategies for latency control. The first is pricing [35, 31]. Usually a higher price attracts more workers and can reduce the latency. The second is latency modeling [90, 81]. There are mainly two latency models: the round model [81, 90] and the statistical model [103, 31]. The round model leverages the idea that tasks can be published in multiple rounds. If there are enough active workers on the crowdsourcing platform, the latency of answering tasks in each round can be regarded as constant time. Thus the overall latency is modeled as the number of rounds. The statistical model is also used to model latency, which leverages the collected statistics from previous crowdsourcing tasks to build statistical models that can capture the workers' arrival time, the completion time, etc. These derived models can then be used to predict and perhaps adjust for expected latency.

3.4 Trade-Off

There is a tradeoff among cost, quality, and latency. Firstly, the cost-control techniques may sacrifice the quality. For example, the answer deduction may reduce the quality if the crowd makes an error in their answers, and pruning can decrease the quality if some important tasks are pruned as discussed above. Thus, some studies study how to balance quality and cost [93, 95, 19]. Secondly, there

is also a trade-off between latency and cost. For example, in order to reduce cost, some cost-control techniques (e.g., answer detection) have to publish tasks in multiple rounds. However, increasing the number of rounds will lead to long latency. Thirdly, the similar trade-off also exists between latency and quality. For example, to increase quality, task assignment assigns hard tasks to more workers and easy tasks to fewer workers. To achieve this goal, it needs to select tasks in multiple rounds to better understand the tasks. Thus, a large number of rounds can improve the quality but reduce the latency. To balance the trade-off among quality, cost, and latency, existing studies focus on different problem settings, e.g., optimizing the quality given a fixed cost, minimizing the cost with a little sacrifice of quality, reducing the latency given a fixed cost, etc.

4. CROWDSOURCING SYSTEMS

Several crowdsourcing database systems [33, 68, 74, 30] have been proposed to encapsulate the process of interacting with the crowdsourcing platforms. The basic workflow of query processing consists of query parser, query plan generation, optimization, and execution. Given a query, a parser is first applied and multiple plans can be generated. Then the query optimization selects the best query plan, and finally executes the plan. Existing crowdsourcing database systems focus on query model, query operators, and query optimization techniques. Next we discuss different existing crowdsourced optimization techniques and operators.

4.1 Crowdsouring Optimization

CrowdDB [33] extends SQL and defines a new query language, called CrowdSQL, to define which table or attribute should be crowdsourced. In query processing, CrowdDB introduces three crowd operators: CrowdProbe (collect missing information of attributes or new tuples), CrowdJoin (a nested-loop join over two tables), and CrowdCompare (compare between two elements). CrowdDB proposes rule-based optimization techniques for processing queries with multiple operators.

Qurk [68] uses an SQL-based query language with user-defined functions (UDFs) to enable crowdsourced data management. Qurk focuses on implementing join and sort. It has two important components for cost optimization: task cache and task model. Task cache maintains the crowdsourced answers from previous tasks, while task model trains a model to predict the results for the tasks based on the data that are already collected from the crowd. It uses cost-based optimization to select a good query plan.

Deco [74] focuses on crowdsourcing missing values or new tuples based on the defined fetch rules. In query optimization, to find the best query plan with the minimum cost, it considers both cost estimation and optimal query generation. For cost estimation, it proposes an iterative approach to estimate the cost for a query plan. For optimal query plan generation, it enumerates all possible query plans and selects the best query plan with the least estimated cost.

4.2 Crowdsouring Operators

There are many crowdsourced operators proposed to enable real-world applications, e.g., Selection [73, 72, 81, 103], Collection [86, 76, 29], Join [93, 95, 17, 91, 96, 36, 99], Topk/Sort [39, 19, 77, 26, 39, 21, 104, 54, 107], Categorize [75], Aggregation [39, 88, 66, 42, 21], Skyline [62, 63, 37], Planning [51, 64, 106, 83, 84], Schema Matching [105, 70, 28], Mining [11, 12, 9, 10], and Spatial Crowdsourcing [85, 45, 46]. Various techniques are adopted to optimize the operator's trade-off among three factors: cost, quality and latency. To obtain high-quality results, different applications require to use different crowdsourced operators, which have operator-specific optimization goals over three factors: cost, quality and latency. Specifically, for each operator, we discuss optimization goals (cost, quality, or latency), and the techniques used to achieve the desired optimization goal. For example, in the selection operator, a type of query is filtering [73, 72], which filters items based on several properties (e.g., pictures with humans on them). They usually perform quality and cost trade-off, e.g., given a fixed budget, the quality of the query should be optimized. To improve quality, truth inference and task assignment are often adopted, which infer the truth correctly and judiciously assign tasks to appropriate workers; while to improve cost, more advanced task selection methods (e.g., based on predicate cardinality estimation) are often preferred.

5. CROWDSOURCING CHALLENGES

Query Optimization. An SQL query often corresponds to multiple query plans and it relies on a query optimizer to select the best plan. Traditionally, the way a query optimizer works is to estimate the computation cost of each query plan and choose the one with the minimum estimated cost. However, this process turns to be quite challenging in a crowdsourcing environment because (1) there are three optimization objectives (result quality, monetary cost, and latency) that need to be considered and (2) humans are much more unpredictable than machines.

Benchmark. A large variety of TPC benchmarks (e.g., TPC-H for analytic workloads, TPC-DI for data integration) standardize performance comparisons for database systems and promote the development of database research. Although there are some open datasets [5], there is still lack of standardized benchmarks available. In order to better explore the research topic, it is important to study how to develop evaluation methodologies and benchmarks for crowdsourced data management systems.

Big Data. In the big data era, data volumes are increasing very fast. Compared to machines, humans are much more expensive, and thus it would be increasingly more costly to apply crowdsourcing to emerging big data scenarios. There are existing works that aim to address this problem, but they only work for some certain data processing tasks, such as data cleaning [94], data labeling [69]. Therefore, it is important to continue this study and to develop new techniques that work for all kinds of data processing tasks.

Macro-Tasks. Most of existing studies focus on micro-tasks, which can be easily assigned to workers and instantly answered by workers. However, many real applications need to use macro-tasks, such as writing a paper. Macro-tasks are hard to be split and accomplished by multiple workers, because they will loose the context information if they are split [40]. Workers are not interested in answering a whole macro-task as each macro-task will take a long time. Thus it is rather challenging to support macro-tasks, including automatically splitting a macro-task, assigning tasks to crowd or machines, and automatically aggregating the answers.

Privacy. There are several types of privacy issues in crowdsourcing. First, the requester wants to protect the privacy of their tasks [102]. The tasks may contain sensitive attributes and could cause privacy leakage. Malicious workers could link them with other public datasets to reveal individual private information. Although the requester can publish anonymity data to the workers using existing privacy techniques, e.g., K-Anonymity, it may lower down the quality as the workers cannot get the precise data. Thus it is challenging to trade-off the accuracy and privacy for requesters. Second, the workers have privacy-preserving requirement. Personal information of workers can be inferred from the answers provided by the workers, such as their locations, professions, hobbies. On the other hand, the requester wants to assign their tasks to appropriate workers that are skilled at their tasks (or close to the tasks).

Mobile Crowdsourcing. With the growing popularity of smartphones, there are emerging mobile crowdsourcing platforms, e.g.,

gMission [20], Waze [6], ChinaCrowd [4]. These mobile platforms pose new challenges for crowdsourced data management. First, more factors (e.g., spatial distance, mobile user interface) will affect workers' latency and quality. It is more challenging to control quality, latency and cost for mobile platforms. Second, traditional crowdsourcing platforms adopt worker selection model to assign tasks; however mobile crowdsourcing requires to support server assignment model, which calls for new task assignment techniques.

Acknowledgment

Guoliang Li was supported by 973 Program of China (2015CB358700), NSF of China (61373024, 61632016, 61422205, 61472198), and FDCT/007/2016/AFJ. Ju Fan was supported by the NSF of China (61602488) and CCF-Tencent Open Research Fund (CCF-Tencent RAGR20160108). Jiannan Wang was supported in part by an NSERC Discovery Grant (No. 19579) and an SFU President's Research Start-up Grant (NO. 877335). Reynold Cheng and Yudian Zheng were supported by the Research Grants Council of Hong Kong (RGC Projects HKU 17229116 and 17205115) and the University of Hong Kong (Projects 102009508 and 104004129).

6. REFERENCES

[1] https://www.mturk.com/.
[2] http://www.crowdflower.com.
[3] https://docs.aws.amazon.com/AWSMechTurk/latest/RequesterUI/amt-ui.pdf.
[4] http://www.chinacrowds.com.
[5] http://dbgroup.cs.tsinghua.edu.cn/ligl/crowddata.
[6] https://www.waze.com.
[7] S. Amer-Yahia and S. B. Roy. Human factors in crowdsourcing. *Proceedings of the VLDB Endowment*, 9(13):1615–1618, 2016.
[8] Y. Amsterdamer, S. Davidson, A. Kukliansky, T. Milo, S. Novgorodov, and A. Somech. Managing general and individual knowledge in crowd mining applications. In *CIDR*, 2015.
[9] Y. Amsterdamer, S. B. Davidson, T. Milo, S. Novgorodov, and A. Somech. Oassis: query driven crowd mining. In *SIGMOD*, pages 589–600. ACM, 2014.
[10] Y. Amsterdamer, S. B. Davidson, T. Milo, S. Novgorodov, and A. Somech. Ontology assisted crowd mining. *PVLDB*, 7(13):1597–1600, 2014.
[11] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart. Crowd mining. In *SIGMOD*, pages 241–252. ACM, 2013.
[12] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart. Crowdminer: Mining association rules from the crowd. *PVLDB*, 6(12):1250–1253, 2013.
[13] A.P.Dawid and A.M.Skene. Maximum likelihood estimation of observer error-rates using em algorithm. *Appl.Statist.*, 28(1):20–28, 1979.
[14] B. I. Aydin, Y. S. Yilmaz, Y. Li, Q. Li, J. Gao, and M. Demirbas. Crowdsourcing for multiple-choice question answering. In *AAAI*, pages 2946–2953, 2014.
[15] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W. C. Tan. Asking the right questions in crowd data sourcing. In *ICDE*, 2012.
[16] C. C. Cao, J. She, Y. Tong, and L. Chen. Whom to ask? jury selection for decision making tasks on micro-blog services. *PVLDB*, 5(11):1495–1506, 2012.
[17] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. Cost-effective crowdsourced entity resolution: A partial-order approach. In *SIGMOD*, pages 969–984, 2016.
[18] L. Chen, D. Lee, and T. Milo. Data-driven crowdsourcing: Management, mining, and applications. In *ICDE*, pages 1527–1529. IEEE, 2015.
[19] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *WSDM*, pages 193–202, 2013.
[20] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, Y. Tong, and C. J. Zhang. gmission: a general spatial crowdsourcing platform. *PVLDB*, 7(13):1629–1632, 2014.
[21] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries. In *ICDT*, pages 225–236, 2013.
[22] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW*, pages 469–478, 2012.
[23] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J.R.Statist.Soc.B*, 30(1):1–38, 1977.
[24] A. Doan, M. J. Franklin, D. Kossmann, and T. Kraska. Crowdsourcing applications and platforms: A data management perspective. *Proceedings of the VLDB Endowment*, 4(12):1508–1509, 2011.
[25] C. B. Eiben, J. B. Siegel, J. B. Bale, S. Cooper, F. Khatib, B. W. Shen, F. Players, B. L. Stoddard, Z. Popovic, and D. Baker. Increased diels-alderase activity through backbone remodeling guided by foldit players. *Nature biotechnology*, 30(2):190–192, 2012.
[26] B. Eriksson. Learning to top-k search using pairwise comparisons. In *AISTATS*, pages 265–273, 2013.
[27] J. Fan, G. Li, B. C. Ooi, K. Tan, and J. Feng. icrowd: An adaptive crowdsourcing framework. In *SIGMOD*, pages 1015–1030, 2015.
[28] J. Fan, M. Lu, B. C. Ooi, W.-C. Tan, and M. Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *ICDE*, pages 976–987, 2014.
[29] J. Fan, Z. Wei, D. Zhang, J. Yang, and X. Du. Distribution-aware crowdsourced entity collection. *IEEE Trans. Knowl. Data Eng.*, 2017.
[30] J. Fan, M. Zhang, S. Kok, M. Lu, and B. C. Ooi. Crowdop: Query optimization for declarative crowdsourcing systems. *IEEE Trans. Knowl. Data Eng.*, 27(8):2078–2092, 2015.
[31] S. Faradani, B. Hartmann, and P. G. Ipeirotis. What's the right price? pricing tasks for finishing on time. In *AAAI Workshop*, 2011.
[32] J. Feng, G. Li, H. Wang, and J. Feng. Incremental quality inference in crowdsourcing. In *Database Systems for Advanced Applications - 19th International Conference, DASFAA 2014, Bali, Indonesia, April 21-24, 2014. Proceedings, Part II*, pages 453–467, 2014.
[33] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD*, pages 61–72, 2011.
[34] J. Gao, Q. Li, B. Zhao, W. Fan, and J. Han. Truth discovery and crowdsourcing aggregation: A unified perspective. *Proceedings of the VLDB Endowment*, 8(12):2048–2049, 2015.
[35] Y. Gao and A. G. Parameswaran. Finish them!: Pricing algorithms for human computation. *PVLDB*, 7(14):1965–1976, 2014.
[36] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. W. Shavlik, and X. Zhu. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*, pages 601–612, 2014.
[37] B. Groz and T. Milo. Skyline queries with noisy comparisons. In *PODS*, pages 185–198, 2015.
[38] A. Gruenheid, D. Kossmann, S. Ramesh, and F. Widmer. Crowdsourcing entity resolution: When is A=B? Technical report, ETH Zürich.
[39] S. Guo, A. G. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD*, pages 385–396, 2012.
[40] D. Haas, J. Ansel, L. Gu, and A. Marcus. Argonaut: Macrotask crowdsourcing for complex data processing. *PVLDB*, 8(12):1642–1653, 2015.
[41] D. Haas, J. Wang, E. Wu, and M. J. Franklin. Clamshell: Speeding up crowds for low-latency data labeling. *PVLDB*, 9(4):372–383, 2015.
[42] H. Heikinheimo and A. Ukkonen. The crowd-median algorithm. In *HCOMP*, 2013.
[43] C.-J. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *ICML*, pages 534–542, 2013.
[44] C.-J. Ho and J. W. Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, 2012.
[45] H. Hu, G. Li, Z. Bao, and J. Feng. Crowdsourcing-based real-time urban traffic speed estimation: From speed to trend. In *ICDE*, 2016.
[46] H. Hu, Y. Zheng, Z. Bao, G. Li, and J. Feng. Crowdsourced poi labelling: Location-aware result inference and task assignment. In *ICDE*, 2016.
[47] P. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *SIGKDD Workshop*, pages 64–67, 2010.
[48] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy. Pay-as-you-go user feedback for dataspace systems. In *SIGMOD*, pages 847–860, 2008.
[49] M. Joglekar, H. Garcia-Molina, and A. G. Parameswaran. Evaluating the crowd with confidence. In *SIGKDD*, pages 686–694, 2013.
[50] M. Joglekar, H. Garcia-Molina, and A. G. Parameswaran. Comprehensive and reliable crowd assessment algorithms. In *ICDE*, pages 195–206, 2015.
[51] H. Kaplan, I. Lotosh, T. Milo, and S. Novgorodov. Answering planning queries with the crowd. *PVLDB*, 6(9):697–708, 2013.
[52] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *NIPS*, pages 1953–1961, 2011.
[53] L. Kazemi, C. Shahabi, and L. Chen. Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In *SIGSPATIAL*, pages 304–313, 2013.
[54] A. R. Khan and H. Garcia-Molina. Hybrid strategies for finding the max with the crowd. Technical report, 2014.
[55] H.-C. Kim and Z. Ghahramani. Bayesian classifier combination. In *AISTATS*, pages 619–627, 2012.
[56] L. I. Kuncheva, C. J. Whitaker, and C. A. Shipp. Limits on the majority vote accuracy in classifier fusion. *Pattern Anal. Appl.*, 6(1):22–31, 2003.
[57] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. Crowdsourced data management: A survey. *TKDE*, 28(9):2296–2319, 2016.
[58] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han. A confidence-aware approach for truth discovery on long-tail data. *PVLDB*, 8(4):425–436, 2014.
[59] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD*, pages 1187–1198, 2014.
[60] Q. Liu, J. Peng, and A. T. Ihler. Variational inference for crowdsourcing. In *NIPS*, pages 701–709, 2012.

[61] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. CDAS: A crowdsourcing data analytics system. *PVLDB*, 5(10):1040–1051, 2012.
[62] C. Lofi, K. E. Maarry, and W. Balke. Skyline queries in crowd-enabled databases. In *EDBT*, pages 465–476, 2013.
[63] C. Lofi, K. E. Maarry, and W. Balke. Skyline queries over incomplete data - error models for focused crowd-sourcing. In *ER*, pages 298–312, 2013.
[64] I. Lotosh, T. Milo, and S. Novgorodov. Crowdplanr: Planning made easy with crowd. In *ICDE*, pages 1344–1347. IEEE, 2013.
[65] F. Ma, Y. Li, Q. Li, M. Qiu, J. Gao, S. Zhi, L. Su, B. Zhao, H. Ji, and J. Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *KDD*, pages 745–754. ACM, 2015.
[66] A. Marcus, D. R. Karger, S. Madden, R. Miller, and S. Oh. Counting with the crowd. *PVLDB*, 6(2):109–120, 2012.
[67] A. Marcus and A. Parameswaran. Crowdsourced data management industry and academic perspectives. *Foundations and Trends in Databases*, 6(1-2):1–161, 2015.
[68] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, pages 211–214, 2011.
[69] B. Mozafari, P. Sarkar, M. Franklin, M. Jordan, and S. Madden. Scaling up crowd-sourcing to very large datasets: a case for active learning. *PVLDB*, 8(2):125–136, 2014.
[70] Q. V. H. Nguyen, T. T. Nguyen, Z. Miklós, K. Aberer, A. Gal, and M. Weidlich. Pay-as-you-go reconciliation in schema matching networks. In *ICDE*, pages 220–231. IEEE, 2014.
[71] W. R. Ouyang, L. M. Kaplan, P. Martin, A. Toniolo, M. B. Srivastava, and T. J. Norman. Debiasing crowdsourced quantitative characteristics in local businesses and services. In *IPSN*, pages 190–201, 2015.
[72] A. G. Parameswaran, S. Boyd, H. Garcia-Molina, A. Gupta, N. Polyzotis, and J. Widom. Optimal crowd-powered rating and filtering algorithms. *PVLDB*, 7(9):685–696, 2014.
[73] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD*, pages 361–372, 2012.
[74] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *CIKM*, pages 1203–1212. ACM, 2012.
[75] 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.
[76] H. Park and J. Widom. Crowdfill: collecting structured data from the crowd. In *SIGMOD*, pages 577–588, 2014.
[77] T. Pfeiffer, X. A. Gao, Y. Chen, A. Mao, and D. G. Rand. Adaptive polling for information aggregation. In *AAAI*, 2012.
[78] V. C. Raykar and S. Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research*, 13:491–518, 2012.
[79] V. C. Raykar, S. Yu, L. H. Zhao, A. K. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *ICML*, pages 889–896, 2009.
[80] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *JMLR*, 11(Apr):1297–1322, 2010.
[81] A. D. Sarma, A. G. Parameswaran, H. Garcia-Molina, and A. Y. Halevy. Crowd-powered find algorithms. In *ICDE*, pages 964–975, 2014.
[82] P. Smyth, U. M. Fayyad, M. C. Burl, P. Perona, and P. Baldi. Inferring ground truth from subjective labelling of venus images. In *NIPS*, pages 1085–1092, 1994.
[83] H. Su, K. Zheng, J. Huang, H. Jeung, L. Chen, and X. Zhou. Crowdplanner: A crowd-based route recommendation system. In *ICDE*, pages 1144–1155. IEEE, 2014.
[84] H. Su, K. Zheng, J. Huang, T. Liu, H. Wang, and X. Zhou. A crowd-based route recommendation system-crowdplanner. In *ICDE*, pages 1178–1181, 2014.
[85] H. To, G. Ghinita, and C. Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *PVLDB*, 7(10):919–930, 2014.
[86] B. Trushkowsky, T. Kraska, M. J. Franklin, and P. Sarkar. Crowdsourced enumeration queries. In *ICDE*, pages 673–684, 2013.
[87] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *WWW*, pages 155–164, 2014.
[88] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis. Max algorithms in crowdsourcing environments. In *WWW*, pages 989–998, 2012.
[89] V. Verroios and H. Garcia-Molina. Entity resolution with crowd errors. In *ICDE*, pages 219–230, 2015.
[90] V. Verroios, P. Lofgren, and H. Garcia-Molina. tdp: An optimal-latency budget allocation strategy for crowdsourced MAXIMUM operations. In *SIGMOD*, pages 1047–1062, 2015.
[91] N. Vesdapunt, K. Bellare, and N. N. Dalvi. Crowdsourcing algorithms for entity resolution. *PVLDB*, 7(12):1071–1082, 2014.
[92] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.
[93] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. CrowdER: crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.
[94] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *SIGMOD*, pages 469–480, 2014.
[95] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD*, 2013.
[96] S. Wang, X. Xiao, and C. Lee. Crowd-based deduplication: An adaptive approach. In *SIGMOD*, pages 1263–1277, 2015.
[97] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. The multidimensional wisdom of crowds. In *NIPS*, pages 2424–2432, 2010.
[98] P. Welinder and P. Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *CVPR Workshop (ACVHL)*, pages 25–32. IEEE, 2010.
[99] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. *PVLDB*, 6(6):349–360, 2013.
[100] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. R. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043, 2009.
[101] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043, 2009.
[102] S. Wu, X. Wang, S. Wang, Z. Zhang, and A. K. H. Tung. K-anonymity for crowdsourcing database. *TKDE*, 26(9):2207–2221, 2014.
[103] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *MobiSys*, pages 77–90, 2010.
[104] P. Ye, U. EDU, and D. Doermann. Combining preference and absolute judgements in a crowd-sourced setting. In *ICML Workshop*, 2013.
[105] C. J. Zhang, L. Chen, H. V. Jagadish, and C. C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *PVLDB*, 6(9):757–768, 2013.
[106] C. J. Zhang, Y. Tong, and L. Chen. Where to: Crowd-aided path selection. *PVLDB*, 7(14):2005–2016, 2014.
[107] X. Zhang, G. Li, and J. Feng. Crowdsourced top-k algorithms: An experimental evaluation. *PVLDB*, 9(4):372–383, 2015.
[108] Z. Zhao, F. Wei, M. Zhou, W. Chen, and W. Ng. Crowd-selection query processing in crowdsourcing databases: A task-driven approach. In *EDBT*, pages 397–408, 2015.
[109] Z. Zhao, D. Yan, W. Ng, and S. Gao. A transfer learning based framework of crowd-selection on twitter. In *SIGKDD*, pages 1514–1517, 2013.
[110] Y. Zheng, R. Cheng, S. Maniu, and L. Mo. On optimality of jury selection in crowdsourcing. In *EDBT*, pages 193–204, 2015.
[111] Y. Zheng, G. Li, and R. Cheng. Docs: Domain-aware crowdsourcing system. *PVLDB*, 10(4):361–372, 2016.
[112] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng. Truth inference in crowdsourcing: Is the problem solved? *PVLDB*, 10(5):541–552, 2017.
[113] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*, pages 1031–1046, 2015.
[114] D. Zhou, S. Basu, Y. Mao, and J. C. Platt. Learning from the wisdom of crowds by minimax entropy. In *NIPS*, pages 2195–2203, 2012.

Biography

Guoliang Li (liguoliang@tsinghua.edu.cn) is currently working as an associate professor in the Department of Computer Science, Tsinghua University, Beijing, China. His research interests mainly include data cleaning and integration, and crowdsourcing.

Yudian Zheng (ydzheng2@cs.hku.hk) is currently a PhD candidate in the Department of Computer Science, University of Hong Kong. His main research interests include data analysis and data management in crowdsourcing.

Ju Fan (fanj@ruc.edu.cn) is currently working as an associate professor in the Department of Computer Science, Renmin University, Beijing, China. His main research interests include crowdsourcing and knowledge base.

Jiannan Wang (jnwang@sfu.ca) is currently working as an assistant professor in the School of Computing Science at Simon Fraser University, Canada. His main research interests include data cleaning and crowdsourcing.

Reynold Cheng (ckcheng@cs.hku.hk) is currently working as an associate professor in the Department of Computer Science, University of Hong Kong. His main research interests include large-scale data management, knowledge bases and crowdsourcing.