Cleaning Uncertain Data With Crowdsourcing - A General Model With Diverse Accuracy Rates

Chen Zhang, *Member, IEEE*, Haodi Zhang, Weiteng Xie, Nan Liu, Qifan Li, Di Jiang, Peiguang Lin, Kaishun Wu, *Member, IEEE*, and Lei Chen, *Fellow, IEEE*

Abstract—Since inaccuracies commonly exist in many applications, data uncertainty has become an important problem in database systems. To deal with data uncertainty, probabilistic databases can be used to store uncertain data, and querying facilities are provided to yield answers with confidence. However, the results from a query or mining process may not be reliable when the uncertainty propagates in the systems. In this paper, we leverage the power of crowdsourcing by designing a set of Human Intelligence Tasks, or HITs in short, to ask a crowd to improve the quality of uncertain data. In particular, we consider crowds consists of workers with diverse accuracy rates when answering the HITs. We design solutions to maximize the data quality with minimal number of HITs. There are two obstacles for this non-trivial optimization, which lead to very high computational cost for selecting the optimal set of HITs. First, members of a crowd may return incorrect answers with different probabilities. Second, the HITs decomposed from uncertain data are often correlated. We have addressed these challenges in this paper by designing an effective approximation algorithm and an efficient heuristic solution, especially for crowds with diverse individual accuracy rates. To further improve the efficiency, we derive tight lower and upper bounds for effective filtering and estimation. Extensive experiments on both a simulated crowd and a real crowdsourcing platform are conducted to evaluate our solutions.

Index Terms—Crowdsourcing, cleaning uncertain data, approximation algorithm, heuristic algorithm

1 Introduction

Since inaccuracies commonly exist in many applications, data uncertainty has become an important problem in database systems. In the field of information extraction, some learning-based model (e.g., Conditional Random Field) usually produces a ranked list of extractions, each of which is associated with a probability of correctness [1]. In entity resolution and schema matching, automatic tools may generate multiple results, each of which is associated with a confidence value indicating the matching quality [2], [3]. When data from multiple data sources are integrated, conflicting information may be handled by keeping conflicted data associated with probabilities, a.k.a uncertain/probabilistic data.

As possible worlds exponential increases due to different combinations of real instances of uncertain data, mining uncertain data is often computationally expensive. Besides, the returned results from queries may not be useful, since the uncertainty in the data may propagate. Thus, improvement of the data quality by reducing the uncertainty is needed for many applications. In quite a few cases, data becomes uncertain because computers have difficulties to recognize human-intuitive semantics via the given data representation (e.g., images, natural language) or handle human-intrinsic tasks (e.g., extracting the correct information or language translation). Therefore, intuitively, human perception should be very effective in cleaning uncertain data. In the existing works of uncertain data cleaning [4], [5], it is assumed that there is a cleaning agent (e.g., an expert) available, which may access and return the ‘ground truth’ of the uncertain data. However, in many circumstances, such availability does not hold. Along with the emerging of uncertain data, crowdsourcing has become a hot research topic due to the success of several crowdsourcing platforms such as Amazon Mechanical Turk, CrowdFlower and CrowdOTA [6]. These platforms provide human computational power, which is not only always available, but is at least fairly affordable. In this paper, we leverage the power of crowdsourcing for cleaning uncertain data. Compared with the conference version of this work [7], we adopt a more realistic setting and generalize the model, in which the crowd consists of workers with diverse individual accuracy rates, instead of the same rate when answering the HITs.

In this paper, we consider the *x-relation* model, one of existing uncertainty models. The x-relation model is widely

- *Chen Zhang is with the School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan 250014, China, and also with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong, SAR China. E-mail: czhangad@cse.ust.hk.*
- *Haodi Zhang, Weiteng Xie, Nan Liu, and Qifan Li are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China. E-mail: {hdzhang, wtxie, nliu, qfli}@szu.edu.cn.*
- *Di Jiang is with WeBank AI, Shenzhen 518052, China. E-mail: dijiang@webank.com.*
- *Peiguang Lin is with the School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan 250014, China. E-mail: linpg@sdufe.edu.cn.*
- *Kaishun Wu is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Guangzhou HKUST Fok Ying Tung Research Institute, Nansha District, Guangzhou City, 511458, China. E-mail: wu@szu.edu.cn.*
- *Lei Chen is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, SAR China. E-mail: leichen@cse.ust.hk.*

Manuscript received 15 Nov. 2019; revised 8 July 2020; accepted 13 Sept. 2020. Date of publication 30 Sept. 2020; date of current version 7 July 2022.
(Corresponding author: Haodi Zhang.)
Recommended for acceptance by B. Glavic.
Digital Object Identifier no. 10.1109/TKDE.2020.3027545

TABLE 1
Location Address of T_1: 34A Broadway East San Francisco

Xid	Tid	House no	Area	City	$Pr(t_i)$
T_1	t_1	34	Broadway	San Francisco	0.4
T_1	t_2	34A	Broadway East	San Francisco	0.3
T_1	t_3	34A	Broadway East	Francisco	0.2
T_1	t_4	34	East	Francisco	0.1

TABLE 2
Candidate HITs for Cleaning With Crowd Workers, and Distributions of the Ground Truth and Crowdsourced Answers

ID	HIT content	$Pr(h_c)$	P_{cr}	$Pr(A_c)$
HIT1	Is the House no 34?	y(0.5) n(0.5)	0.90	y(0.5) n(0.5)
HIT2	Is the House no 34A?	y(0.5) n(0.5)	0.65	y(0.5) n(0.5)
HIT3	Is the Area Broadway?	y(0.4) n(0.6)	0.55	y(0.49) n(0.51)
HIT4	Is the Area Broadway East?	y(0.5) n(0.5)	0.70	y(0.5) n(0.5)
HIT5	Is the Area East?	y(0.1) n(0.9)	0.85	y(0.22) n(0.78)
HIT6	Is the City San Francisco?	y(0.7) n(0.3)	0.60	y(0.54) n(0.46)
HIT7	Is the City Francisco?	y(0.3) n(0.7)	0.75	y(0.4) n(0.6)

adopted in the field of uncertain database [4], [5], [8], [9], [10]. In this model, an *x-relation* contains a set of independent *x-tuples*. Each x-tuple includes a set of mutually exclusive tuples (or called *alternatives*), associated with probabilities, which represent a discrete probability distribution of these tuples being correct. An example probabilistic database is illustrated in Table 1. The uncertainty of each entry is characterized by an x-tuple. Each x-tuple represents an address of a facility location. The locations can be provided by some information extraction model such as HMM or CRF. Detailed generation of such probabilistic databases from extraction models is available in [1]). As shown in Table 1, the x-tuple T_1 indicates four possible addresses for a location, t_1 - t_4, for textual data "34A Broadway East San Francisco". The probability of t_1 being correct is 0.4. Such a probabilistic data model well characterizes the correlation among the attributes (labels). In the example of Table 1, the labels 'House no', 'Area' and 'City' are correlated: the marginal probability $Pr(House\ no = 34A) = Pr(t_2) + Pr(t_3) = 0.5$, while conditional probability $Pr(House\ no = 34|Area = East) = 1$ and $Pr(House\ no = 34|Area = Broadway\ East) = 0$.

To measure the data quality of an x-tuple, we follow the proposal in [4], [5], [11] and use the *negative value of the Shannon entropy*, i.e., $\sum_p p \log p$. For instance, the quality of T_1 in Table 1, denoted by $Q(T_1)$, is computed as $Q(T_1) = 0.4 * \log 0.4 + 0.3 * \log 0.3 + 0.2 * \log 0.2 + 0.1 * \log 0.1 = -0.55$. This measure quantifies the ambiguity of an x-tuple: the more uncertain the data, the lower the quality. Our core objective is to improve the quality of x-tuples with the help of crowdsourcing.

1.1 Crowdsourcing-Based Data Cleaning

In this subsection, we introduce the crowd-aided uncertainty cleaning. We first demonstrate in Table 1 an example of uncertain data. The data uncertainty in the example is essential, because data extraction techniques cannot fully interpret the location address written in natural language. Therefore, it is critical to investigate the following question: where can we find the human perceptions concerning the given uncertain data? Fortunately, we have crowdsourcing as a promising solution. With the recent development of crowdsourcing in both academic and industrial communities, there are sophisticated on-line crowdsourcing platforms, for instance, Mechanical Turk, CrowdFlower, etc. They serve affordable sources of human perceptions and are commonly used. The expected information concerning the given uncertain data can be queried via publishing questions, named Human Intelligent Tasks, or HITs in short, each of which would generate a monetary cost. Moreover, the work-flow of publishing HITs can be automated with available APIs provided by above platforms. Thus, we can efficiently clean the data uncertainty and improve the quality by issuing a number of HITs. Thus, the key to the task is how to generate and crowdsource proper HITS.

As to the design of proper HITs, it is commonly accepted that crowdsourcing works best when tasks can be broken down into simple pieces [12], [13], [14]. A complete tuple with many attributes is probably too large for crowdsourcing workers. In reality, if a proposed tuple contains many attributes, most individual workers may have small quibbles with it, so that a simple question on the correctness of tuple may get mostly negative answers, declaring it less than perfect. There is a good chance that such negative answers are misjudgment due to the complexity of the tasks. In other words, determining the correctness of complex tuples with many attributes is too difficult for crowdsourcing workers. On the other hand, asking open-ended questions is not recommended for crowdsourcing workers either, because it may be difficult to integrate multiple suggestions of heterogeneous semantics.

Therefore, each HIT proposed to the workers should be simple and clear enough. For crowdsourcing-based data cleaning, we propose to ask the crowdsourcing workers questions regarding individual cells. Each HIT is a question of form "Is the *A.att v*?" , where v and *A.att* represent a cell value and the corresponding attribute, respectively. Given an example in Table 1, there are seven candidate HITs available to be selected for crowdsourcing, as shown in Table 2. Compared with questions directly about entire tuples, these simpler questions about a single attribute can be easily answered with a yes or no, in most circumstances.

With the running example in Tables 1 and 2, we demonstrate how the crowdsourcing worker is going to clean the uncertain data by answering HITs, and improve the data quality as a consequence. Assuming that "I: City = San Francisco" (i.e., HIT6) is confirmed to be true by a crowdsourcing worker. Then we have

$$\begin{aligned} Pr(t_1|I) &= \frac{Pr(t_1) \cdot Pr(I|t_1)}{Pr(I) = Pr(t_1) + Pr(t_2)} \\ &= \frac{0.4 \times 1}{0.4 + 0.3} = 0.57 > 0.5. \end{aligned} \tag{1}$$

Similarly $Pr(t_2|I) = 0.43$, $Pr(t_3|I) = 0$ and $Pr(t_4|I) = 0$. As a result, the quality $Q(T_1)$ is improved from -0.55 to -0.3.

In the above example, we assume that obtaining information I from crowdsourcing worker with 100 percent confidence, which is unrealistic if we consider the error rate of crowdsourcing worker. As pointed out in paper [15], even with the simple tasks such as labeling, the quality of normal worker (not sloppy worker or spammer) is around 75 percent.

Clearly, this quality varies depending on many factors, including the format and domain of the HIT, the time of publication, the HIT interface etc. Sometimes the quality may be even worse than 75 percent. Nevertheless, such noisy answers may still be conductive as far as the uncertainty is well managed. Continued with the running example, we make a more realistic assumption: "F: City = San Francisco" is answered by a crowdsourcing worker cr with general confidence 60 percent. Assuming the HIT is answered independently by a worker, we have

$$\begin{aligned} & Pr(t_1|Env: F \text{ is obtained from crowd}) \\ &= Pr(t_1)\cdot Pr(Env|t_1)/Pr(Env) \\ &= \frac{Pr(t_1)\cdot Pr(cr)}{Pr(F)\cdot Pr(cr) + (1-Pr(F))\cdot Pr(\neg cr)} \\ &= \frac{0.4\times 0.6}{0.7\times 0.6 + 0.3\times 0.4} = 0.44, \end{aligned} \tag{2}$$

where I is the crowdsourced fact, and in $Pr(cr)$ and $Pr(\neg cr)$, cr represents "worker cr is correct" for simplicity. Similarly, $Pr(t_2|e) = 0.34$, $Pr(t_3|e) = 0.15$ and $Pr(t_4|e) = 0.07$. The quality $Q(T_1)$ is hereby improved after crowdsourcing a few more HITs, the confidence of the correct tuple is increased to be close to 1, while the others approach to 0. Hence, the quality of uncertain data is still improved after cleaning, even when the crowdsourcing worker is imperfect.

1.2 Challenges and Contributions

Although crowdsourcing provides a new approach for uncertain data cleaning, it is not trivial to deploy a crowdsourcing framework for the task. First, each HIT is associated with a cost, we need to design solutions to maximize the data quality by selecting an optimal minimum set of HITs. Second, crowdsourcing workers are not perfect and may return incorrect answers. What makes it worse is that, in reality, different workers have different accuracy rates. Third, the HITs decomposed from an x-tuple are naturally correlated, which makes greedy selection does not actually work. The above facts lead to the NP-hardness for selecting the optimal set of HITs. To address this challenge, we not only design an approximation algorithm with approximation guarantee $(1-1/e)$, but also propose a heuristic solution, which is much more efficient than the approximation one, but has comparable effectiveness.

In our previous conference paper [7], we addressed this problem and assumed that each crowdsourcing worker committed to the same accuracy rate. While in this paper, we consider a more general and realistic situation: 1) each HIT has a probability that it will be answered correctly according to the hardness of HIT, and 2) every crowdsourcing worker has a probability of answering HIT correctly, which indicates the credibility of the worker. Therefore, the model in [7] is a special case of our proposal, in which all crowdsourcing workers answer HIT with the same accuracy rate. Under the two conditions above, we can compute the probability that HIT is answered correctly, and propose the k HITs selected by the model without assuming a unified accuracy rate of crowdsourcing workers. We summarize our contributions as follows:

1) We indicate how to utilize noisy crowdsourced answers (each with an individual accuracy rate) to improve the data quality of an x-tuple, and analyze the functional relations among the accuracy rates of crowdsourcing workers, data quality and a candidate HIT.
2) For a crowd with diverse accuracy rates, we address the core optimization problem, namely, how to select the most profitable questions with a cardinality constraint. We prove that the expected quality improvement by asking these HITs is equivalent to the difference between the 'joint entropy' and 'sum of entropy values of crowdsourcing workers'. This greatly reduces the time complexity of our algorithm. We also prove that finding the exact solution is NP-hard, and an approximation algorithm as well as a heuristic solution are proposed.
3) Compared with the earlier version of the paper [7], we remove the strong assumption that all crowdsourcing workers have the same accuracy rate, and generalize the model. Besides, we derive new lower and upper bounds, which are tighter than the previous ones, for effective filtering and estimation.
4) We have conducted extensive experiments on both synthetic and real data sets, with crowds consisting of workers that have diverse accuracy rates. The experimental results show that the data quality is significantly improved via the power of crowdsourcing, even without the original assumption of uniform accuracy. We also compare our method with automatic cleaning tools such as HoloClean, and propose a natural way to combining the two approaches.

2 Problem Formulation

In this section, we first give formal definition of uncertain data of form x-tuples, and corresponding quality metric, and then define crowd-based cleaning operations, and finally formulate the problem of utilizing crowdsourcing to clean uncertain data.

2.1 Uncertain Data of Form X-Tuples

In this subsection, we deploy the same definition of uncertain data and the metric for data quality as in the conference version. Table 3 gives the meanings of the main symbols used in this paper.

TABLE 3
Meanings of Symbols

Symbol	Meaning
T	an x-tuple
$Q(T)$	data quality of T
$t_1,\ldots,t_{\lvert T\rvert}$	tuples of T
$Pr(t_1),\ldots,Pr(t_{\lvert T\rvert})$	probabilities of tuples
ATT	semantic attributes of T
c_{ij} (or simply c)	value of attribute att_j in tuple t_i
h_c	the cleaning HIT verifying c/c_{ij}
$Pr(h_c)$	probability of c being the correct value of att_j
A_h	the crowdsourced answer of HIT h
$Pr(A_h)$	probability of crowdsourcing workers answering 'yes' to HIT h
P_{cr}	the accuracy of the crowd worker cr
$H(cr)$	the entropy of the crowdsourcing worker cr
$H(x)$	the entropy of a random variable x

Definition 1 (x-tuple). *For a given table under schema Σ, which consists of a set of* semantic attributes $ATT = \{att_1, att_2, \ldots, att_{|ATT|}\}$. *Each tuple t_i consists of three components: a unique identifier id, a confidence $P(t_i)$ that is the probability of t_i being correct, and the set of semantic attributes ATT. An x-tuple T consists of one or more tuples i.e., $T = \{t_1, t_2, \ldots, t_{|T|}\}$. $|ATT|$ and $|T|$ are the size of ATT and T, respectively. Each of the tuples is associated with a probability of being correct. Let $Pr(t_i)$ denote the probability of t_i being correct, then*

$$\sum_{i=1}^{|T|} Pr(t_i) \leq 1. \tag{3}$$

Please note that the *semantic* attributes indicate the attributes describing human-intrinsic information of an entity. Besides the semantic attributes, schema Σ may also consist of other attributes (e.g., IDs). In the example of Table 1, "House no", "Area" and "City" are semantic attributes, while others are not. Intuitively, only data under semantic attributes worth being crowdsourced.

Definition 2. *Given an x-tuple T, the* quality *of T, denoted by $Q(T)$, is the negative value of the Shannon entropy, i.e.,*

$$Q(T) = \sum_{t \in T} Pr(t) \cdot \log Pr(t), \tag{4}$$

where $\sum_{t \in T} Pr(t) = 1$.

Shannon entropy is commonly used for measuring the uncertainty of the data [4], [5]. Shannon entropy quantifies the randomness of a random variable, and low randomness indicates high quality of uncertain data. However, we adopt the metric to evaluate the quality of an individual x-tuple, rather than the answer of a query. By improving the quality of the data of form x-tuples, the quality of any query answers would be naturally improved. Notice that if $\sum_{t \in T} Pr(t) < 1$, we conceptually insert into T a placeholder tuple t_{null}, with null values for all attributes and probability $Pr(t_{null}) = 1 - \sum_{t \in T} Pr(t)$.

2.2 Crowdsourcing Framework for Uncertain Data Cleaning

We follow the framework in the conference version of this paper for cleaning the data uncertainty.

Definition 3. *Given x-tuple T, let c_{ij} be the cell intersecting tuple $t_i \in T$ and attribute att_j. A* cleaning HIT *(or HIT in short) corresponding to c_{ij}, denoted $h_{c_{ij}}$, is a human intelligent task asking the crowdsourcing worker to verify whether c_{ij} is the correct value for att_j in T. Moreover, let $Pr(h_{c_{ij}})$ be the probability that the correct answer of $h_{c_{ij}}$ is 'yes', then*

$$Pr(h_{c_{ij}}) = \sum_{t_x \models c_{ij}} Pr(t_x), \tag{5}$$

where $t_x \models c_{ij}$ denotes c_{ij} is the value of att_j in tuple t_x, i.e., t_x entails c_{ij}. In other words, $t_x \models c_{ij}$ means that if t_x is correct, then c_{ij} is correct.

From Definition 3, we can see that the candidate HITs can be easily constructed for a given x-tuple T. Since different tuples may have the same value for an attribute, the probability of c_{ij} being the correct value of att_j (i.e., the correct answer of $h_{c_{ij}}$ is yes) can be computed by summing up the probabilities of tuples in which c_{ij} is the value of att_j. The total number of HITs constructed for t, denoted by m, is upper bounded by $|ATT| \cdot |T|$, i.e., the number of attributes multiply the number of tuples. To be more specific, $m = \sum_{i=1}^{|ATT|} opts(att_i)$, where $opts(att_i)$ is the number of all options for attribute att_i. Each HIT can be considered as a random variable following a Bernoulli distribution. The distributions of HITs in Table 1 are demonstrated. Please note that the HITs derived from the same x-tuple are correlated in general. In the example of Table 1, given that "the House no is 34A" (HIT2), we can easily infer that "the Area is Broadway East"(HIT4), and the probability of "the City is Francisco"(HIT7) becomes 0.4 (similar to Equation (1)).

Due to differences in skill levels, or the amount of time and effort spent, the answers returned by crowdsourcing workers may be imperfect. We model the potential noisy answers by a general error model for crowdsourcing workers, illustrated as follows.

Definition 4. *For a crowd worker cr, cr's* accuracy rate, *denoted by $P_{cr} \in [0.5, 1]$, is the probability that cr correctly answers each HIT.*

It is worth mentioning that each HIT is assumed to be answered independently. Actually, how to model the crowdsourcing workers is application-specific. While some works assume that the crowdsourced answers are 100 percent accurate [13], [16], [17], [18], we adopt a more general error model, which ensures that the answer returned by the crowdsourcing worker is always correct with a probability no lower than $1/2$, and each crowdsourcing worker with different accuracy rates,as shown in Definition 4. This is a classical crowdsourcing model widely used by a stream of works [19], [20], [21], [22], [23]. A crowdsourcing worker may have different accuracy for different domains. The accuracy for a domain can be easily estimated with a set of sample HITs with ground truth.

Definition 5. *Given a crowdsourcing worker cr and its accuracy P_{cr}, the* entropy of worker *cr is*

$$H(cr) = -P_{cr} \cdot \log P_{cr} - (1 - P_{cr}) \cdot \log (1 - P_{cr}). \tag{6}$$

Given the crowdsourcing worker's accuracy, $H(cr)$ is a positive constant measuring the randomness of the crowd worker's behavior.

Definition 6. *For an x-tuple T, a crowdsourcing worker with accuracy $P_{cr} \in [0.5, 1]$, the answer A_c for h_c provided by the crowdsourcing worker, the* entropy of answer *A_c is*

$$\begin{aligned} H(A_c) = &-Pr(A_c = yes) \cdot \log Pr(A_c = yes) \\ &- Pr(A_c = no) \cdot \log Pr(A_c = no). \end{aligned}$$

For convenience, we regard h_c and A_c that appear in probability functions to be Boolean variables

$$\begin{aligned} &h_c: \text{ the ground truth of HIT } h_c \text{ is yes;} \\ &A_c: \text{ HIT } h_c \text{ is answered yes by the crowdsourcing worker.} \end{aligned} \tag{7}$$

Thus, the entropy of A_c is

$$\begin{aligned} H(A_c) = &- Pr(A_c) \cdot \log Pr(A_c) \\ &- (1 - Pr(A_c)) \cdot \log (1 - Pr(A_c)). \end{aligned} \tag{8}$$

2.3 Problem Statement

Now, we formally define the problem of data cleaning via crowdsourcing as follows. Given an x-tuple T, the budget B, crowdsourcing workers $C = \{cr_1, cr_2, \ldots, cr_k\}$, each with individual accuracy $P_{cr_1}, P_{cr_2}, \ldots, P_{cr_k}$, our goal is to maximize the quality $Q(T)$ by selecting HITs and receiving at most B crowdsourced answers. A crowd is considered as a tool for cleaning uncertain data. However, since a crowd may be noisy and unreliable, the first concern is how to use the noisy crowdsourced answers to improve data quality. In Section 3, we present a series of theoretical analyses, proving that the expected improvement of data quality is always non-negative. Then, in Section 4, we optimize the crowdsourcing process by selecting the most profitable set of HITs. In Section 5 we formally introduce the algorithms, and finally evaluate our solutions with experiments in Section 6.

3 Utilization of Crowdsourced Answers

A crowd can be noisy, hence each crowdsourced answer is inherently uncertain. In this section, we discuss how to use uncertain crowdsourced answers to improve the data quality of x-tuples. Notice that in this paper, we consider a diverse crowd, in which the workers have different error rates.

3.1 Merging Crowdsourced Answers into X-Tuples

Since both crowdsourced answers and x-tuple are essentially uncertain, the effect of crowdsourced answers on data quality can be treated as posterior probabilities conditioning on answers, which is properly addressed with Bayes' theorem.

First, for a given x-tuple T, we clarify the functional relations among the accuracy rates of crowdsourcing workers $P_{cr_1}, P_{cr_2}, \ldots, P_{cr_k}$, data quality $Q(T)$ and a candidate HIT h_c. As mentioned above, please note the difference between h_c and A_c: h_c denotes the 'ground truth' of the HIT, i.e.,

$$\begin{aligned} Pr(h_c) &= Pr(h_c \text{ is 'yes'}) \\ &= Pr(\text{the ground-truth answer for } c \text{ is 'yes'}), \end{aligned} \tag{9}$$

while A_c denotes the 'crowdsourced answer' of the HIT, and

$$\begin{aligned} Pr(A_c) &= Pr(A_c \text{ is 'yes'}) \\ &= Pr(\text{the crowdsourced answer for } c \text{ is 'yes'}). \end{aligned} \tag{10}$$

Because the crowd may make mistakes, they follow different distributions. Since the HITs are assumed to be answered independently, the relation between $Pr(h_c)$ and $Pr(A_c)$ can be expressed as the following equation:

$$Pr(A_c) = Pr(h_c) \cdot P_{cr} + (1 - Pr(h_c)) \cdot (1 - P_{cr}). \tag{11}$$

In Table 2, the values of $Pr(h_c)$ and $Pr(A_c)$ are demonstrated, which reveal the differences discussed above.

We now illustrate how to compute $Q(T|A_c)$ and $Q(T|\neg A_c)$, after an answer is crowdsourced from some worker. With each crowdsourced answer A_c, the probability of each tuple $t \in T$ is updated with the received answer A_c. Intuitively, if t indicates contradicting content as A_c, then $Pr(t)$ should decrease; otherwise $Pr(t)$ should increase since it is further confirmed by the crowdsourcing worker. Explicitly, for each crowdsourced answer A_c, the probability of any $t \in T$ is modified as

$$\begin{aligned} &Pr(t|A_c = yes) = Pr(t) \cdot Pr(A_c = yes|t)/Pr(A_c) \\ &Pr(t|A_c = no) = Pr(t) \cdot Pr(A_c = no|t)/(1 - Pr(A_c)), \end{aligned} \tag{12}$$

where $Pr(A_c|t) = P_{cr}$ when $t \models c$ (i.e., the crowdsourcing worker confirms t), and $Pr(A_c|t) = 1 - P_{cr}$ when $t \models \neg c$ (i.e., the crowdsourcing worker disagrees with t). By recursively applying Equation (12), conflicting crowdsourced answers are integrated into the x-tuple t.

It is easy to perform the algebraic manipulations to show that, for any two answers A_{c_0} and A_{c_1}, we have

$$Pr(t|A_{c_0}, A_{c_1}) = Pr(t|A_{c_1}, A_{c_0}). \tag{13}$$

Equation (13) indicates that the final result of adjustment is independent of the sequence of the crowdsourced answers. In other words, when we have a deterministic set of HITs, it does not matter in what sequence the answers are used for adjusting the distribution in the x-tuple. In contrast, what matters is to determine the set of HITs to be asked, which is the core challenge addressed in Section 4.

3.2 Crowdsourcing Worker's Accuracy versus Data Quality

According to the Equations (11) and (12), we introduce two theorems from [7] to conclude the relation between the crowdsourcing worker's accuracy and the data quality (Theorems *Harmless Random* and *Non-negative Expectation* in [7]).

Theorem 1. *If a crowdsourcing worker randomly answers a HIT, i.e., $P_{cr} = 0.5$, then it does not affect the data quality.*

Now we compute the expectation of data quality after receiving the answer A_c, denoted by $\mathbb{E}Q(T|A_c)$.

Theorem 2. *Given an x-tuple T, if a HIT h_c is answered by a crowdsourcing worker with accuracy P_{cr}, the expected improvement of data quality of T is mathematically equivalent to the difference between 'the entropy of the crowdsourced answer of h_c' and the 'entropy of the crowdsourcing worker (Definition 5)', which is non-negative.*

Notice that the quality of an x-tuple does not necessarily monotonically increase as receiving crowdsourced answers. It is possible for the quality to decrease, when a surprisingly wrong answer is received. For instance, a 'no' is answered by some worker for a HIT which has a quite high probability to be correct, or vice versa. However, one can see that the data quality obviously converges to zero (i.e., the upper bound of data quality), since each HIT has a non-negative expectation on the change of quality. To achieve fast convergence, the key issue is how to select the HITs wisely.

4 Top-k HITs With Diverse Accuracy Rates

The core computational challenge to address is to select a collection of HITs with a cardinality constraint k. For the given budget B, we are interested in asking k HITs at each iteration, so there are totally $\lceil \frac{B}{k} \rceil$ iterations for budget to run out. When $k > 1$, different workers can then pick up these tasks and solve them in parallel, cutting down wall-clock time. Therefore, for a given budget of HITs B, larger k value leads to lower latency. However, we pay for this by having some questions answered that are not at the top of the list - we are issuing k good questions rather than only the very best one. At each iteration, we select k distinct HITs; but the same HIT may be repetitively selected in different iterations. Please note that the selection of HITs depends on the crowdsourced answers received from previous iterations. So the overall selection is an adaptive process, and how many times each HIT is asked during the whole process is determined by the crowdsourced answers and the proposed algorithms discussed in the rest of this section.

For a given x-tuple with m candidate distinct HITs, there are potentially C_m^k possible selections. In this section, we discuss how to select k HITs in order to maximize the expected quality improvement.

4.1 Deriving Objective Function

As stated, in our early version of the paper [7], we assumed that all crowdsourcing works have the same accuracy, which is a very strong assumption. In this paper, we extend the model to allow diverse accuracy rates, and each worker cr_i has his/her own accuracy rate P_{cr_i}. Obviously the new model is more general and realistic. Surprisingly, after deleting the strong assumption of uniform accuracy rate, our algorithm is still optimal.

For a given set $S_h = \{h^i_{c_1}, h^i_{c_2}, \ldots, h^i_{c_k}\}$ of k HITs, each of which is assigned to one crowdsourcing work, we derive the expectation of quality improvement caused by the aggregation of crowdsourced answers of these HITs. Let A_{c_1}, A_{c_2},...,A_{c_k} denote the answers of the k HITs with accuracy P_{cr_1}, P_{cr_2},...,P_{cr_k}, then A_{S_h} is a discrete random variable with 2^k possible outcomes, since each h_{c_i} is either answered 'yes' or 'no'. Let D_A and p_A be the domain and probability distribution of A_{S_h}, respectively, i.e.,

$$\begin{aligned} &D_A = \{a_i | a_i \subseteq 2^{\{h_{c_1}, h_{c_2}, \ldots, h_{c_k}\}} \ and \\ &\forall h_{c_j} \in a_i \ , h_{c_j} \ is\ answered\ yes\ by\ crowd; \\ &\ and\ \forall h_{c_j} \notin a_i \ , h_{c_j} \ is\ answered\ no\ by\ crowd\} \\ &p_A = (Pr(a_1), Pr(a_2), \ldots, Pr(a_{2^k})), \end{aligned} \tag{14}$$

where $\sum_{i=1}^{2^k} Pr(a_k) = 1$. Thus the entropy of A_{s_h} is

$$H(A_{s_h}) = -\sum_{a_i \in D_A} Pr(a_i) \cdot \log Pr(a_i). \tag{15}$$

As we know, each c_i has a probability to show its ground truth, i.e., with P_{cr_i} to be true before crowdsourcing worker answer h_{c_n}. We view $U = \{c_n; n = 1, \ldots, k\}$ as a set of random variables which follow Bernoulli distribution and take value true/false. Let D_U and p_U be the domain and probability distribution of $\{c_n; n = 1, \ldots, k\}$ respectively. Each element of D_U is a sequence of true and false with a corresponding probability in p_U. Thus we have

$$\begin{aligned} &D_U = \{u_i | u_i = \{h^i_{c_1}, h^i_{c_2}, \ldots, h^i_{c_k}\}, h^i_{c_n} = ture\ or\ false\} \\ &p_U = (Pr(u_1), Pr(u_2), \ldots, Pr(u_{2^k})), \end{aligned} \tag{16}$$

where $|D_U| = 2^k$. we have

$$Pr(u_i) = \sum_{\substack{t_j \in T \\ n=1,\ldots,k \\ \forall c^i_n = ture \\ [t \models c_i}} Pr(t_j). \tag{17}$$

We aim to find a set of k HITs such that their answers would lead to the maximal (expected) improvement of data quality. As shown in Theorem 4.1, under the assumption that crowdsourcing workers give correct answers with diverse accuracy probability, we prove that the uncertainty reduction by a set of HITs is equivalent to their joint entropy (denoted by $\mathrm{H}(A_{S_h})$) minus sum of entropy values of crowdsourcing workers. So we investigate expected data quality by receiving the crowdsourced answer A_{S_h}. Given fixed k HITs with accuracy P_{cr_1}, P_{cr_2},..., P_{cr_k}, $H(A_{S_h}|T)$ is equivalent sum of entropy values of crowdsourcing workers, i.e., a constant. So, we have

$$\begin{aligned} \Delta Q_{S_h}(T) &= -\sum_{a_i \in D_A} Pr(a_i) \cdot \log Pr(a_i) - \sum_{i=1}^{k} H(cr_i) \\ &= H(A_{S_h}) - \sum_{i=1}^{k} H(cr_i). \end{aligned} \tag{18}$$

We highlight the conclusion of Equation (18) with the following theorem.

Theorem 3 (Objective Function). *Given an x-tuple, a set of k cleaning HITs S_h and diverse accuracy rates of crowdsourcing workers, the expected quality improvement by asking these HITs is equivalent to difference between the 'joint entropy' and 'sum of entropy values of crowdsourcing workers'.*

Since $\sum_{i=1}^{k} H(cr_i)$ is a constant, we only need to select HITs to maximize $H(A_{S_h})$. Formally, we have the following optimization goal:

$$S_h := \arg\max_{S_h} \ H(A_{S_h}). \tag{19}$$

4.2 Lower and Upper Bounds

According to Theorem 3, we expect to select a set of k HITs, with highest $H(A_{S_h})$. For a given set HITs S_h, computing the exact value of $H(A_{S_h})$ encounters exponential complexity, due to the correlation among HITs and the crowd imperfection.

Note $H(A_{S_h}) = -\sum_{a_i \in D_A} Pr(a_i) \cdot \log Pr(a_i)$, the value of $Pr(a_i)$ should be computed first to further compute $H(A_{S_h})$. For k HITs, the answers in a_i are denoted by $A^i_{h_{c_1}}$, $A^i_{h_{c_2}}$,..., $A^i_{h_{c_k}}$. We present the formula for the computation of $Pr(a_i)$

$$Pr(a_i) = \sum_{j=1}^{2^k} Pr(u_j) \cdot Q_{ij}, \tag{20}$$

where

$$Q_{ij} = \prod_{\substack{n=1,..,k \\ c_n^j=ture, A_{c_n^i}=yes}} P_{cr_n} \cdot \prod_{\substack{n=1,..,k \\ c_n^j=ture, A_{c_n^i}=no}} (1-P_{cr_n}) \cdot \prod_{\substack{n=1,..,k \\ c_n^j=false, A_{c_n^i}=yes}} P_{cr_n} \cdot \prod_{\substack{n=1,..,k \\ c_n^j=false, A_{c_n^i}=no}} (1-P_{cr_n}). \tag{21}$$

Clearly, the exact computation yields exponential complexity w.r.t k, which is inefficient when k is large. Thus, we are interested in reducing or avoiding the exact computation pertaining to $H(A_{S_h})$. By Equations (17) and (20), we know that the number of elements with positive probability in D_U is less than or equal to $|T|$ and the time complexity of computing all $Pr(u_i)$ is bounded by $O(k|R|)$, but the time complexity of computing all $Pr(a_i)$ will be $O(2^k)$. So, we hope to bound $H(A_{S_h})$ by $H(S_h)$, in which $H(S_h)$ denotes the joint entropy of HITs in S_h.

In this subsection, we demonstrate how to bound $H(A_{S_h})$ by $H(S_h)$ through mathematical formula as follows. Meanwhile, we propose the upper and lower bounds that are tighter than the ones in [7]. The bounds can be computed in linear time and used to enable effective filtering (in Section 5.1) and heuristic estimation (in Section 5.2).

Upper Bound. please note the difference between $H(S_h)$ and $H(A_{S_h})$ - $H(S_h)$ measures the randomness of the HITs, or more precisely, 'the ground truth of the HITs'; while $H(A_{S_h})$ measures the randomness of the 'crowdsourced answer of the HITs'. Intuitively, this difference is caused by the fact that the crowdsourcing worker makes mistakes. Then by the chain rule of Shannon entropy, we have $H(S_h|A_{S_h}) = H(A_{S_h}, S_h) - H(A_{S_h})$, therefore

$$\begin{aligned} H(A_{S_h}) &= H(A_{S_h}, S_h) - H(S_h|A_{S_h}) \\ &= H(A_{S_h}|S_h) + H(S_h) - H(S_h|A_{S_h}). \end{aligned}$$

We have an upper bound of $H(A_{S_h})$

$$\begin{aligned} H(A_{S_h}) &= H(S_h) + \sum_{i=1}^{k} H(cr_i) - H(S_h|A_{S_h}) \\ &\le H(S_h) + \sum_{i=1}^{k} H(cr_i). \end{aligned} \tag{22}$$

By Equations (15) and (20), we have

$$H(A_{S_h}) = \sum_{i=1}^{2^k}\sum_{j=1}^{2^k} Pr(u_j)Q_{ij}\log Pr(u_j)Q_{ij}. \tag{23}$$

Note that $\sum_{j=1}^{2^k} Pr(u_j) = 1$ and $P_{cr_n} \in [0.5,1]$, we have

$$\sum_{j=1}^{2^k} Pr(u_j)Q_{ij} \ge \min_j Q_{ij} = \prod_{n=1}^{k}(1-P_{cr_n}).$$

Therefore

$$\begin{aligned} H(A_{S_h}) &\le -\sum_{i=1}^{2^k}\sum_{j=1}^{2^k} Pr(u_j)Q_{ij}\log\prod_{n=1}^{k}(1-P_{cr_n}) \\ &= -\sum_{t=1}^{k}\log(1-P_{cr_t}). \end{aligned} \tag{24}$$

Finally, by Equations (22) and (24), we have the following upper bound

$$H(A_{S_h}) = \min\Big(H(S_h) + \sum_{i=1}^{k} H(cr_i), -\sum_{j=1}^{k}\log(1-P_{cr_j})\Big). \tag{25}$$

Lower Bound. we have

$$\begin{aligned} H(A_{S_h}) =& H(S_h) + \sum_{i=1}^{k} H(cr_i) - H(S_h|A_{S_h}) \\ \ge& H(S_h) + \sum_{i=1}^{k} H(cr_i) + \prod_{i=1}^{k} P_{cr_i}\log\prod_{i=1}^{k} P_{cr_i} \\ &+ (1-\prod_{i=1}^{k} P_{cr_i})\log(1-\prod_{i=1}^{k} P_{cr_i}) \\ &- (1-\prod_{i=1}^{k} P_{cr_i})\log(2^k-1). \end{aligned} \tag{26}$$

Similar with the upper bound, we have

$$\begin{aligned} H(A_{S_h}) &\ge -\sum_{i=1}^{2^k}\sum_{j=1}^{2^k} Pr(u_j)\cdot Q_{ij}\cdot\log\prod_{t=1}^{k} P_{cr_t} \\ &= -\sum_{i=1}^{k}\log P_{cr_i}. \end{aligned} \tag{27}$$

Finally, by Equations (26) and (27), we have the following lower bound

$$\begin{aligned} H(A_{S_h}) = \max\Big(& H(S_h) + \sum_{i=1}^{k} H(cr_i) \\ &+ \prod_{i=1}^{k} P_{cr_i}\cdot\log\prod_{i=1}^{k} P_{cr_i} - (1-\prod_{i=1}^{k} P_{cr_i})\cdot\log(2^k-1) \\ &+ (1-\prod_{i=1}^{k} P_{cr_i})\cdot\log(1-\prod_{i=1}^{k} P_{cr_i}), \\ &-\sum_{i=1}^{k}\log P_{cr_i}\Big). \end{aligned} \tag{28}$$

4.3 Computing the Bounds: X-Partition Algorithm

From formula (22) and (27), one can see that we only need to compute the joint entropy $H(S_h)$, since the other components are all constant for a given x-tuple and a set of HITs. A naive way of computing the joint entropy is to compute all the marginal probabilities, hence yields to $O(2^k)$ complexity. However, with creating a small $O(|T|)$ in-memory index, we can achieve overall linear complexity - $O(k|T|)$.

We propose a novel algorithm named "X-partition". The intuition of the algorithm is, for each HIT h_c, the x-tuple T can be divided in to two parts, with $t_i \models c$ and $t_j \models \neg c$ respectively. As a consequence, k HITs can partition T into at most 2^k parts, and the aggregated probability of each part (i.e., the sum of probabilities of entries within a part) constitutes the marginal distribution of S_h. Note each part includes at least one tuple, so the total number of parts is at most $|T|$. Now we illustrate the algorithm in detail as following steps.

Step 1: Remove a HIT h_c from S_h.

Step 2: Partition the table into two parts T_0 and T_1, where $\forall t \in T_0, t \models c$, and $\forall t \in T_1, t \models \neg c$.

Step 3: Remove a HIT $h_{c'}$ from S_h

Step 4: For each of the current parts T_l, further divide it into two parts, T_{l0} and T_{l1}.

Step 5: repeat Step 3 and Step 4 until all k HITs are removed, then return $-\sum p \log p$, where p is the sum of probabilities of tuples within each part.

Correctness and Complexity. The correctness of the algorithm is straightforward. Each part T_l corresponds to a point of the marginal distribution of the S_h, so the entropy of these parts are equivalent to the joint entropy. In addition, the order of HITs would not affect the final partitioning result. At each iteration, for a given HIT, partitioning requires all the tuples, hence takes $O(|T|)$ time. As a result, the overall complexity becomes $O(k|T|)$, which is essentially linear w.r.t the size of input.

Running Example. Given the example in Tables 1 and 2, we consider a set of HITs $S_h = \{h_2, h_3\}$ are answered by two workers with P_{cr_2} = 0.8 and P_{cr_3} = 0.6. Then as shown in Table 2, we have - $Pr(h_2) = 0.5$,$Pr(h_3) = 0.4$, i.e., the probability of h_2 (h_3) has ground truth 'yes' are 0.5 (0.4); moreover, the probability of h_2 (h_3) being answered 'yes' by a crowdsourcing worker is 0.5 (0.45). The distributions of S_h and A_{S_h} is presented in the following table:

outcomes	$Pr(S_h)$	$Pr(A_{S_h})$
$h_2 \wedge h_3$	0	0.216
$h_2 \wedge \neg h_3$	0.5	0.284
$\neg h_2 \wedge h_3$	0.4	0.264
$\neg h_2 \wedge \neg h_3$	0.1	0.236

Note the distribution of S_h indicates the ground truth of the HITs, which is derived from the x-tuple in Table 1. Clearly, this distribution is irrelevant with the accuracy of the crowdsourcing worker. On the other hand, A_{S_h} denotes the distribution of the crowdsourced answer, which is computed according to formula (15). Having these two distributions, we can easily compute $H(S_h) = 1.36$, and $H(A_{S_h}) = 2.0$. We also have $\sum_{n=2}^{3} H(cr_n) = H(cr_2) + H(cr_3) = 0.72 + 0.97 = 1.69$ and -$\sum_2^3 \log(1 - P(cr_n)) = 3.64$, then we have $H(S_h) + \sum_{n=2}^{3} H(cr_n) = 1.36 + 1.69 = 3.05 < 3.64$, by Equation (25), we can get the upper bound is $3.05 > 2.0$. Similarly, by the Equation (28), it is easy to get a lower bound is 1.23, and $1.23 < 2.0$.

5 Hardness and Algorithms

One can see that the optimization problem derived in formula (19) is non-linear, due to the nature of Shannon entropy. In fact, the optimization problem is to find k HITs from the m candidate HITs, so the searching space is of complexity $O(m!)$. However, $H(A_{S_h})$ is the entropy of the crowdsourced answer of a set of HITs, and it is known that entropy is a submodular function [24]. Our optimization problem is essentially a sub-problem of the maximization of a general submodular function. We prove that finding the optimal solution for our optimization problem is NP-hard.

Theorem 4 (NP-hardness). *It is NP-hard to find a set of k HITs such that the expected improvement of data quality is maximized.*

PROOF. Please see appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2020.3027545.

5.1 Approximation Algorithm

Although finding the optimal solution is intractable, it is known that the maximization of submodular functions can be approximated with a performance guarantee of $(1 - 1/e)$, by a greedy algorithm [25] - we iteratively select the best one HIT, given the ones selected so far.

Formally, we have the optimization function at the kth iteration

$$x := \arg\max_x H(A_{(S_{k-1} \cup \{h_x\})}), \tag{29}$$

where S_{k-1} is the set of HITs selected from previous iterations.

At each iteration, we need to determine one HIT h_x out of m candidates, so $H(A_{(S_{k-1} \cup \{h_x\})})$ would be computed at most m times to find the maximum. Therefore, this approximation algorithm entails time complexity $O(2^k m)$, i.e., linear w.r.t the size of the x-tuple, but exponential w.r.t k. In order to further improve the efficiency, we propose two pruning methods, illustrated as follows.

Instance-Level Pruning.

We first adopt the lower upper bounds derived from Section 4.2. For a pair of HITs h_0 and h_1, if we have

$$H(A_{(S_{k-1} \cup \{h_0\})}).lb \geq H(A_{(S_{k-1} \cup \{h_1\})}).ub,$$

then h_1 can be safely pruned. Please note that $A_{(S_{k-1} \cup \{h_0\})}.lb$ and $A_{(S_{k-1} \cup \{h_1\})}.ub$ denote the lower bound of $A_{(S_{k-1} \cup \{h_0\})}$ and the upper bound of $A_{(S_{k-1} \cup \{h_1\})}$, respectively. Both of them can be efficiently computed within $O(k)$ time by the X-partition algorithm in Section 4.3.

Algorithm-Level Pruning.

Submodularity can be exploited algorithmically to implement an accelerated variant of the greedy algorithm. In each of the k iterations, the greedy algorithm must identify the HIT with maximum marginal gain, given the HITs selected in the previous iterations. The key insight is that, as a consequence of submodularity of the objective function, the marginal benefit of any HIT is monotonically non-increasing during the iterations of the algorithm. In other words, for all h and k, the submodularity guarantees that $H(A_{(S_k \cup \{h\})}) \leq H(A_{(S_{k-1} \cup \{h\})})$. Therefore, we maintain a list of upper bounds (i.e., $A_{(S_{k-1} \cup \{h\})}$) on the marginal gains sorted in decreasing order. In each iteration, the algorithm extracts the HIT h_0 with the highest value from the ordered list. As a result, for all h_1, if we have

$$A_{(S_k \cup \{h_0\})} \geq A_{(S_{k-1} \cup \{h_1\})}, \tag{30}$$

then, h_1 can be safely pruned.

5.2 Heuristic Algorithm

In the approximation algorithm above, after applying the pruning methods, we may still have to compute the exact value of $H(A_{(S_{k-1}\cup\{h_x\})})$ for several times, each of which entails exponential complexity w.r.t k. This can be inefficient when k is large. In this subsection, we use the average of upper bound and lower bound to estimate the value of $H(A_{(S_{k-1}\cup\{h_x\})})$ for HIT selection. Explicitly, we apply the following estimator:

$$H(A_{(S_{k-1}\cup\{h_x\})}) \approx \frac{A_{(S_{k-1}\cup\{h_x\})}.lb + A_{(S_{k-1}\cup\{h_x\})}.ub}{2}. \quad (31)$$

We compute the estimator with formula (31), and select the HIT with the highest estimated value. Therefore, considering the entire heuristic algorithm, we do not need to compute any exact value of $H(A_{(S_{k-1}\cup\{h_x\})})$. In the experiments, we show that the effectiveness of this heuristic algorithm is comparable with the approximation algorithm.

The heuristic algorithm is particular useful when k is large - it entails very short (linear w.r.t k) execution time for the program to select HITs. On the other hand, recall that larger k would lead to less overall time cost of crowdsourcing (i.e., the time waiting for the crowd to return answers), since multiple workers can take HITs in parallel. So the heuristic algorithm is recommended over the approximation algorithm when the user expects short overall running time (program execution time + crowdsourcing). In other words, when the budget is the main constraint, small k and the approximation algorithm should be adopted, whereas the heuristic algorithm with a large k value is suggested if the time-efficiency is the primary constraint.

6 Experimental Evaluation

The goal of our experiments contains following aspects, 1) we study the effect of different parameters for quality improvement with respect to our approximation algorithm and heuristic algorithm, especially the effect of considering different accuracy rates of the crowd; 2) we compare the two proposed algorithms with baseline algorithms in term of hardness reduction; 3) we compare our crowdsourcing based algorithm with automatic cleaning tools such as HoloClean, and discuss the combination of the two approaches.

The rest of this section includes following content.

- First, we present in Section 6.1 the necessity of considering different accuracy rates. As in our previous conference paper [7], the workers in the crowd are considered to have the accuracy rate, the performance is obviously worse compared with our method in this paper.
- Second, in Section 6.2, we compare our approximation algorithm and heuristic algorithm with two baselines, with different numbers of selected HITs per iteration. The first baseline just selects HITs randomly, and the other one is a greedy strategy which commits to the largest reduction of uncertainty each iteration.
- Third, in Section 6.3, we approximate the algorithm facilitated by the instance-level pruning and algorithm-level pruning, and then discuss the efficiency.
- Fourth, we test our methods in Section 6.4 with Amazon Mechanical Turk, which is a well-known public crowdsourcing platform. Due to page limits, we omit corresponding experiment results with the uniform accuracy rate in our previous conference paper.
- Finally, we compare our crowdsourcing based algorithm with automatic cleaning tools such as HoloClean [26], and discuss the combination of the two approaches in Section 6.5.

6.1 Diverse Accuracy versus Uniform Accuracy

We first present the necessity of introducing diverse accuracy rates into the framework. In our previous conference paper [7], the workers in the crowd are considered to have accuracy rates. Such an assumption is acceptable when the crowdsourcing workers in reality have very similar quality in term of answer accuracy. However, when there is obvious accuracy diversity in the crowd, which is very common in real applications, the selection of the best HITs will be inaccurate due to the insufficient estimation of the hardness reduction.

Consider such a crowd of workers, in which a half of the population have quite low accuracy rates, say around P_{cr}^{l}, and the other half has high accuracy rates, around P_{cr}^{h}. Now we mixed up these two subgroup of crowdsourcing workers and test the performance on a real dataset *Hospital* in [26], which contains 1,000 x-tuples and 5,000 tuples. As the algorithm in [7] can not distinguish workers with different accuracy rates, we use the average accuracy rate of the crowd as input. Our algorithm with diverse accuracy rates shows much better performance. With ($P_{cr}^{l} = 0.1, P_{cr}^{h} = 0.9$) and ($P_{cr}^{l} = 0.5, P_{cr}^{h} = 1.0$), the performance of our method is significantly better than the algorithm that assumes the same accuracy rate for all users. Particularly in Fig. 2a, with $P_{cr}^{l} = 0.1$ and $P_{cr}^{h} = 0.9$, the previous version of the algorithm in [7] can barely handle the problem. Under the budget limit, the uncertainty of the data is not reduced at all, which is consistent with Theorem 1.

6.2 Performance Comparison on Synthetic Data

In this subsection, we experiment on synthetic data with a simulated crowd. We generate 10,000 x-tuples in total, each of which contains 50 tuples and 20 attributes, and report the average performance. For each published HIT, we randomly generate an accuracy rate $P_{cr} \in [0.5, 1]$ with uniform distribution. Third, given a HIT, we generate the correct yes-no answer with probability P_{cr} (i.e., generate the wrong answer with probability 1 - P_{cr}, and then return the answer and P_{cr} as the inputs for adjustment (Section 3.1). Since each x-tuple usually contains a small number of tuples (less than 15 in the real datasets used in Section 6.4), the synthetic data is fairly large enough to test on the scalability of the proposed algorithms. We set the total budget of HITs $B = 60$ for a given x-tuple, and at each iteration we select k HITs, so there are totally $\lceil 60/k \rceil$ iterations (there are maybe less than k HITs in the last iteration). Then we demonstrate the performances of the approximation algorithm (*appr*), the heuristic algorithm (*heur*), the greedy algorithm (*greed*) and a naive algorithm (*rand*), which selects HITs randomly. Please note that the proposed algorithms are both *linear* w.r.t the size of x-tuples. The main computational challenge is that the time cost of *appr* is exponential w.r.t k. It'll be thoroughly evaluated in the experiments.

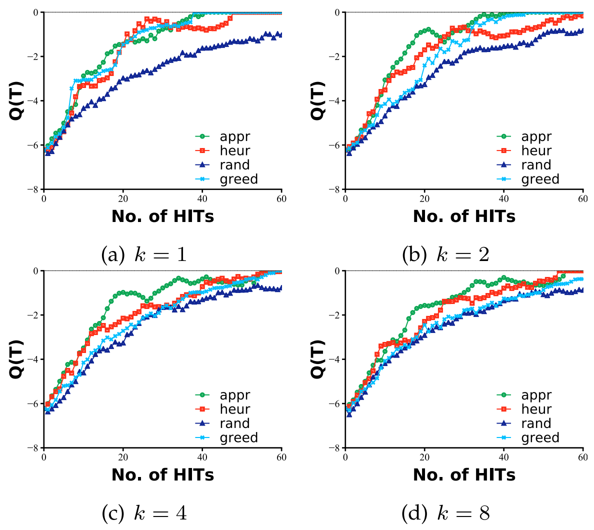


Fig. 1. P_{cr_i} uniformly distributed in [0.5,1] and $k = 1, 2, 4, 8$.

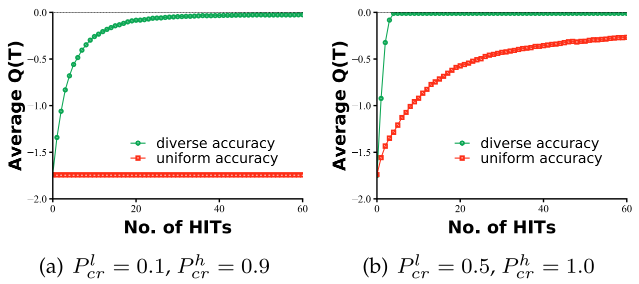


Fig. 2. Diverse accuracy versus uniform accuracy.

We first present the effectiveness of the algorithm with varying k, which is the number of questions selected for each iteration. For a given budget $B = 60$, larger k leads to less iterations. With more than one HITs on the crowd ($k > 1$), different workers can take them on parallel. We set k to 1,2,4,8, and test the performances of the algorithms above. As shown in Fig. 1, smaller k tends to be more effective on quality improvement. In fact, the larger k is, the less advantage *appr* and *heur* have comparing to a random selection or a greedy one. This is because each HIT is selected based on crowdsourced answers of HITs selected in previous iterations. Recall that we select k out of m candidates at each iteration, so when $k = m$, *appr* and *heur* are the same as random selection, i.e., select all of the HITs we have.

We now present the performances of the approximation algorithm (*appr*), the heuristic algorithm (*heur*), the greedy algorithm (*greed*) and a naive algorithm (*rand*). As indicated in Fig. 1, *heur*, *appr* and *greed* significantly outperform *rand* in all cases. It's worth mentioning that *appr*, *heur* and *greed* have almost the same performance when $k = 1$, because at each iteration, the three algorithms would always select the HIT with probability closest to 0.5 to achieve their respective local optimal. When $k > 1$, *heur* and *appr* notably outperform than *greed*, since the greedy algorithm is always looking for the local optimal solution. It is difficult to greedily get the global optimal solution when multiple HITs are given at the same time. We can also observe that *heur* and *appr* occasionally perform similarly; whereas *appr* performs slightly better than *heur* and *greed* in most cases in terms of the number of HITs.

6.3 Approximation and Efficiency

In Sections 5.1 and 5.2, the approximation algorithm (*appr*) and the heuristic algorithm (*heur*) are proposed as solutions for k-HIT selection. Recall that *appr* generates near-optimal solutions with approximation guarantee, but it encounters high computational cost when k is large. Therefore, two filtering techniques are proposed to improve the efficiency. In the following, we demonstrate the computational cost of the proposed algorithms in Fig. 3. In particular, the curves labeled with *appr_i_pruning* and *appr_a_pruning* represent the approximation algorithm facilitated by the instance-level pruning and algorithm-level pruning, respectively. In addition, the curve *appr_both_prunings* is the result with the both pruning methods used. The pruning techniques turn out to significantly improve the efficiency of the algorithm *appr*, and the time cost of *heur* is linear w.r.t k. We can observe that the instance-level pruning is more effective than the algorithm-level one. Moreover, combining the pruning methods is better than any single one of them. We also run the exact solution of k-HIT selection, which enumerates all the possible size-k sets of HITs. However, due to the NP-hardness of the exact solution, we cut off the experiment when $k > 7$, which can not terminate in acceptable time. As mentioned previously, we derive new lower and upper bounds for effective filtering and estimation in this paper. Both of the new bounds are tighter than the ones in our previous conference paper [7], and the tighter bounds benefit on the efficiency of all algorithms, compared with Fig. 2 in [7].

6.4 Performance Testing on Amazon Mechanical Turk

We implement our two approaches on Amazon Mechanical Turk (AMT), which is a widely used crowdsourcing marketplace. We tested a relatively large dataset from the real-world raw data *Hospital* [26]. The tested data from *Hospital* includes 100 x-tuples, each of which has 19 attributes and around 5 tuples. Thus there are in total more than 1,250 tuples and 6,000 HITs. For each x-tuple in the dataset, we use the label in the raw data as the ground truth. Each worker is required to take an unpaid *Qualification Test* containing 10 sample questions. A worker is accepted only if he/she correctly answers at least 6 questions in the qualification test. Each worker is only allowed to answer one HIT from each x-tuple, but they may take multiple HITs from different x-tuples. In order to estimate the crowd's accuracies, we first publish 30 testing HITs for each dataset. Then we estimate the crowd's accuracies with these testing HITs as well as the questions answered in the qualification tests. After the 30 testing HITs being finished, the crowd's accuracies are between 0.64 to 0.86. We randomly sample a precision between these two values to set up the worker's accuracy rate for each crowdsourcing round.

We set the budget $B = 60$, and each HIT is awarded 0.04 USD. We compare *appr*, *heur* and *rand* with $k = 1, 5, 10$. Please note a budget is set up for each x-tuple, so users are allowed to set different budgets for different data instances. For the scalability concerning the number of x-tuples, if the same budget is assumed to be set for all x-tuples, the total number of HITs is simply equal to the 'number of x-tuples' multiplies the 'budget', i.e., linear w.r.t the number of x-tuples. We test the competing algorithms with three groups

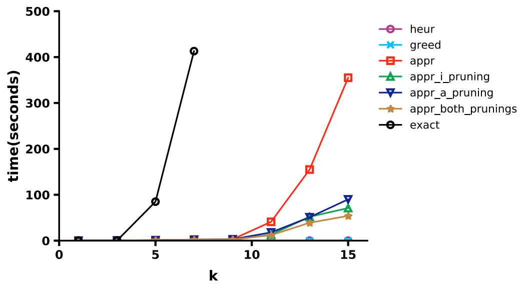


Fig. 3. Efficiency of k-HIT selection algorithms (with new upper and lower bounds for a diverse crowd).

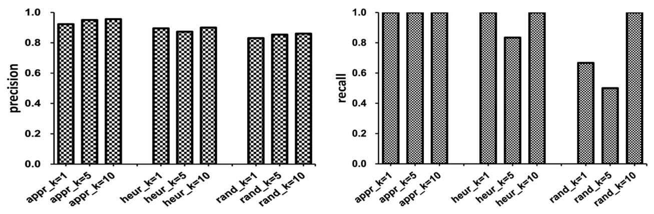


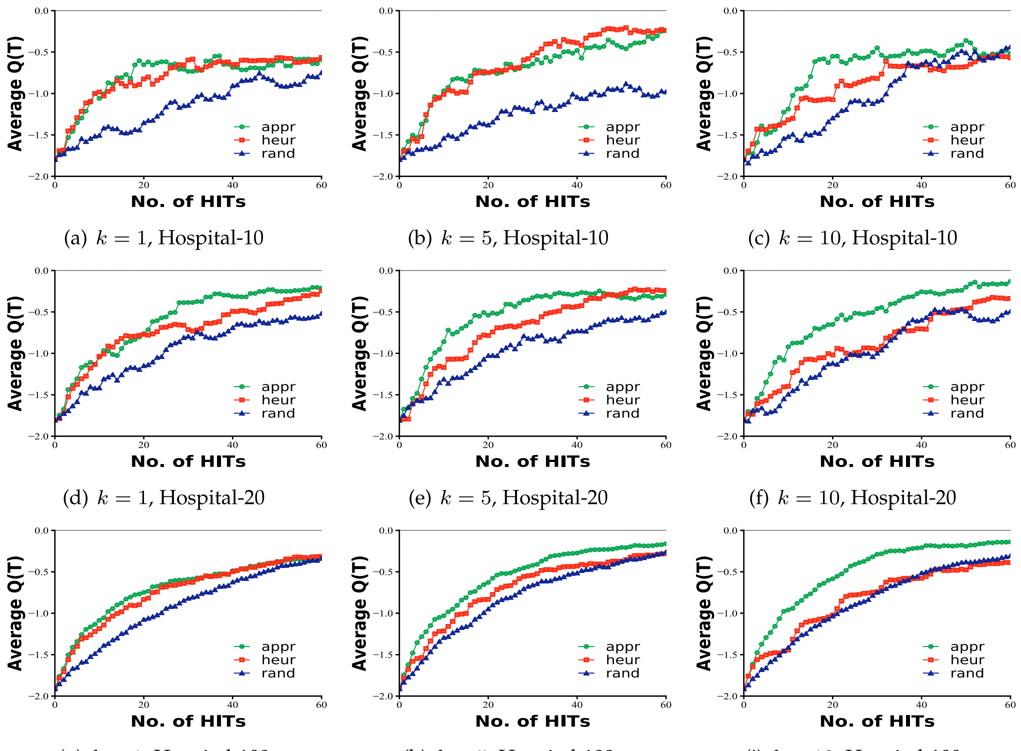
Fig. 5. Data Quality with budget constraint - precision and recall.

of experiments, respectively on 10, 20 and 100 x-tuples with k=1, 5 and 10, and then plot the average Q(T). The average performances are demonstrated in Fig. 4. In terms of the improvement of data quality, one can see that the performance is basically consistent with the simulation - *appr* and *heur* outperform *rand* in all cases in terms of the number of HITs. Moreover, with larger amount of x-tuples, the updated data quality shows smoother improvement during crowdsourcing. In the comparison with HoloClean in Section 6.5, We include the dataset *Hospital* and another large dataset *Flights*, and set the accuracy given to the algorithms to be a random value between 0.64 and 0.86 (lower and upper bound from the qualification tests for the crowd). We will show that, with the very large amount of HITs, the average Q(T) is almost monotonic during crowdsourcing. We conclude that for the two large real-world datasets, our algorithms perform quite well.

Lastly, we verify the correctness of our approaches, by evaluating the accuracy, precision and recall of the best tuple in each x-tuple, i.e., the tuple with the highest possibility after cleaning. The accuracy is the ratio between the number of best tuples that are the same as the ground truth and the total number of x-tuples. Precision is computed as the ratio of the correctly reported correspondences out of all reported correspondences. Recall is computed as the ratio of correctly reported correspondences out of all correspondences in the ground truth. The data accuracy improvement is illustrated in Fig. 6. Please note that "Raw_Data" denotes the data accuracy before cleaning. One can see that the data accuracy is significantly improved after cleaning. Moreover, for *appr* and *heur*, the lower value of k tend to have higher accuracy. This is because smaller k indicates better HIT selections, but longer latency. Fig. 5 illustrates the quality of the best tuples after uncertainty reduction with budget B = 60. From the experimental results, we conclude that *appr* and *heur* significantly outperform than *rand* in all cases, *appr* and *heur* occasionally perform similarly. This is basically consistent with the experimental results in Fig. 6.

6.5 Comparison and Combination With Automatic Cleaning Method

We compare our crowdsourcing based approach with some automatic data cleaning methods such as HoloClean [26]. HoloClean is a framework for holistic data repairing driven by probabilistic inference. The framework repairs dirty data



(a) $k = 1$, Hospital-10 (b) $k = 5$, Hospital-10 (c) $k = 10$, Hospital-10

(d) $k = 1$, Hospital-20 (e) $k = 5$, Hospital-20 (f) $k = 10$, Hospital-20

(g) $k = 1$, Hospital-100 (h) $k = 5$, Hospital-100 (i) $k = 10$, Hospital-100

Fig. 4. Testing on Amazon mechanical turk.

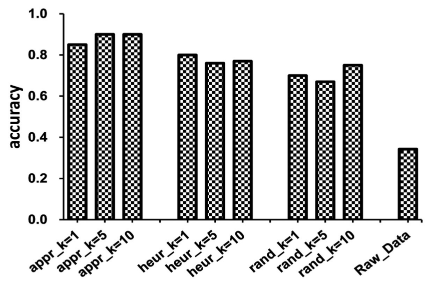


Fig. 6. Data accuracy improvement.

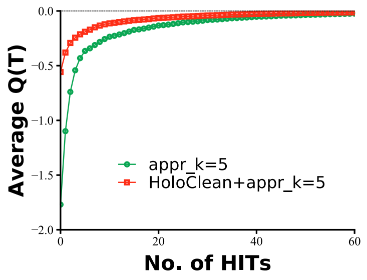


Fig. 8. Combining crowd-aided algorithm with HoloClean on *Hospital*.

by using some explicit rules or constraints from human experts, some additional external information and quantitative statistics. Compared to our crowd-aided method, the human expertise is embedded in the explicit rules or constraints, instead of the crowdsourced answers. The repair of the data by HoloClean consists of two steps, namely noisy cell recognition and data correction. In the former step, the system finds all possible noisy cell by either some predefined first-order logical rules, or some external data information, or some detection techniques for data abnormality. Then in the second step, HoloClean proposes probability inference model and repair the data in the noisy cells, under the weak supervision by the above detection results. After repairing, the system outputs the distribution of possible values of each noisy cells.

To compare our crowd-aided algorithm with the automatic cleaning approach, we formulate the datasets used in [26] as x-tuples. Each item in the raw data is regarded as an x-tuple, which is split into several candidate tuples, according to the candidate values of the attributes, and the distribution of the tuples is initialized with the statistical distribution. These x-tuples are then given to approximation algorithm to get a updated distribution of the candidate tuples for each x-tuple. Finally, with the updated distributions of the candidate tuples (which are joint probability distributions), we compute the marginal probability distributions on the candidate values of the noisy cells. We use the labels in dataset as the ground truth to evaluate the output. Thus, we can compare the methods on two large real datasets, *Flights* and *Hospital*, in term of the precision, the recall, and F1 score. The dataset *Flights* contains 57,246 x-tuples and 286,230 tuples, and *Hospital* contains 1,000 x-tuples and 5,000 tuples. Note that the performance of HoloClean is effected by a pruning threshold τ, which cuts off some minor noises and allows users to tradeoff the scalability and the quality of repairs obtained by it. Higher values of τ lead to smaller domains for random variables, thus, smaller size factor graphs and more efficient inference. For both of the datasets, we compare our method with the best performance of HoloClean, namely *Flights* with $\tau = 0.3$ and *Hospital* with $\tau = 0.5$. As shown in Fig. 7, our method performs better than HoloClean on both datasets.

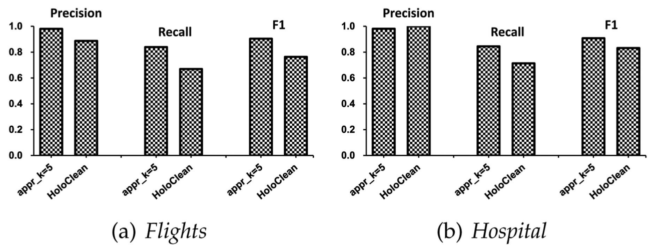


Fig. 7. Comparison with HoloClean on *Flights* and *Hospital*.

It worth mentioning that the crowd-aided method and the automatic cleaning can be combined. The output of HoloClean can be used to initialize the distributions of the candidate tuples before crowdsourcing. We compare the performances of the approximation algorithm with two different initial distributions, one of which uses the joint distributions computed according to the output of HoloClean, and the other one uses the joint distributions computed according to statistical distributions in the raw data. The average $Q(T)$ of 1,000 x-tuples from *Hospital* is plotted in Fig. 8. The result shows that 1) the output of HoloClean is further improved by crowdsourced answers, and 2) the distributions from HoloClean as initial input can also accelerate the convergence of our algorithm.

7 Related Works

7.1 Cleaning Uncertain Data

Data cleaning techniques have been developed for decades by the database community [27], [28]. In particular, there is a number of works studying the issues of cleaning uncertain data [5], [29]. Given that a limited amount of resources to clean the database, [4] describes a technique for choosing the set of uncertain objects to be cleaned, in order to achieve the best improvement in the quality of query answers. They develop a quality metric, namely PWS-quality, for a probabilistic database, and they investigate how such a metric can be used for data cleaning purposes. Please note PWS-quality is essentially the negative value of Shannon entropy, which is the same metric we adopt in this paper. In [5], the authors extend the approach of [4] to support top-k queries. [29] provides an analysis on the sensitivity of a probabilistic query answer to the input data. [30] applies user's feedback to improve the quality of an uncertain database integrated from different sources.

This paper distinguishes itself from the existing works in two perspectives as follows. First, we consider the crowdsourcing workers as a special resource for cleaning uncertain data. The speciality includes 1) the crowdsourcing workers works best when the tasks are broken down into small pieces; and 2) crowdsourcing workers may provide wrong information. Second, we focus on cleaning individual data instances (i.e., x-tuples), not query answers. By improving the quality of data instances, any related queries can be benefited.

7.2 Crowdsourcing

The recent development of crowdsourcing brings us a new opportunity to engage human intelligence into the process of answering queries [31], [32], [33]. Crowdsourcing provides a new problem-solving paradigm [34], [35], which has been blended into several research communities, including database and data mining [36]. In the practical viewpoint, [37] proposed and develop a query processing system using microtask-based crowdsourcing to answer queries. In [38], crowdsourcing is used for top-k query processing over uncertain data and [39] studied knowledge base semantic integration using crowdsourcing. Moreover, in [40], a declarative query model is proposed to cooperate with standard relational database operators. In addition, in the viewpoint of theoretical study, many fundamental queries have been extensively studied, including filtering [22], max [19], sorting [15], join [16], etc. Besides, crowdsourcing-based solutions of many complex algorithms are developed, such as categorization based on graph search [13], clustering [41], entity resolution [17], [18], tagging in social networks [42], trip planning [43], topic discovery [44] etc. In the recent work [45], the authors use crowdsourcing techniques to improve the results of information extraction systems. Our work is essentially different from [45] discussed as follows. First, during the selection of HITs, [45] uses the 'token entropy' and 'mutual information' to select HITs, but does not consider the error rate of the crowd, which is one of the core challenges addressed in our model. Second, [45] only focuses on data from information extraction systems, and their methods cannot be used when the uncertainty is caused by other reasons.

8 Conclusion

In this paper, we propose utilizing the power of crowdsourcing on cleaning uncertain data, in particular, with the crowd containing workers that have diverse accuracy rates. Uncertainty is inherited in many modern applications, such as information extraction, entity resolution, schema matching, and truth discovery. We believe that adopting crowdsourcing as a new component of a DBMS would be extremely conductive for the improving the quality of uncertain data, hence effectively improve the overall performance. Our work represents an initial solution towards cleaning uncertain data with crowdsourcing.

Acknowledgment

This work was partially supported by the Hong Kong RGC GRF Project 16202218, CRF Project C6030-18G, C1031-18G, C5026-18G, AOE Project AoE/E-603/18, China NSFC No. 61729201 and No. 61806132, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, Didi-HKUST joint research lab project, the Priming Research Fund in Shenzhen University, Tencent Rhino-Bird Young Faculty Open Fund, and Wechat and Webank Research Grants. Chen Zhang and Haodi Zhang contributed equally to this work.

References

[1] R. Gupta and S. Sarawagi, "Creating probabilistic databases from information extraction models," in *Proc. 32nd Int. Conf. Very Large Data Bases*, 2006, pp. 965–976.
[2] P. Singla and P. Domingos, "Entity resolution with Markov logic," in *Proc. 6th Int. Conf. Data Mining*, 2006, pp. 572–582.
[3] A. Gal, "Managing uncertainty in schema matching with top-K schema mappings," in *Journal on Data Semantics VI*. Berlin, Germany: Springer, 2006, pp. 90–114.
[4] R. Cheng, J. Chen, and X. Xie, "Cleaning uncertain data with quality guarantees," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 722–735, 2008.
[5] L. Mo, R. Cheng, X. Li, D. W. Cheung, and X. S. Yang, "Cleaning uncertain data for top-k queries," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2013, pp. 134–145.
[6] X. Yu, G. Li, Y. Zheng, Y. Huang, S. Zhang, and F. Chen, "CrowdOTA: An online task assignment system in crowdsourcing," in *Proc. IEEE 34th Int. Conf. Data Eng.*, 2018, pp. 1629–1632.
[7] C. J. Zhang, L. Chen, Y. Tong, and Z. Liu, "Cleaning uncertain data with a noisy crowd," in *Proc. 31st IEEE Int. Conf. Data Eng.*, 2015, pp. 6–17.
[8] P. Agrawal *et al.*, "Trio: A system for data, uncertainty, and lineage," in *Proc. 32nd Int. Conf. Very Large Data Bases*, 2006, pp. 1151–1154.
[9] Y. Tong, L. Chen, and B. Ding, "Discovering threshold-based frequent closed itemsets over probabilistic data," in *Proc. IEEE 28th Int. Conf. Data Eng.*, 2012, pp. 270–281.
[10] Y. Tong, L. Chen, Y. Cheng, and P. S. Yu, "Mining frequent itemsets over uncertain databases," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1650–1661, 2012.
[11] R. Cheng, "Querying and cleaning uncertain data," in *Proc. Int. Workshop Qual. Context*, 2009, pp. 41–52. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04559-2_4
[12] H. Park and J. Widom, "Query optimization over crowdsourced data," *Proc. VLDB Endowment*, vol. 6, no. 10, pp. 781–792, 2013.
[13] A. G. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom, "Human-assisted graph search: It's okay to ask questions," *Proc. VLDB Endowment*, vol. 4, no. 5, pp. 267–278, 2011.
[14] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut, "CrowdForge: Crowdsourcing complex work," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 43–52.
[15] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller, "Human-powered sorts and joins," *Proc. VLDB Endowment*, vol. 5, no. 1, pp. 13–24, 2011.
[16] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, "Leveraging transitive relations for crowdsourced joins," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2013, pp. 229–240.
[17] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "CrowdER: Crowdsourcing entity resolution," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1483–1494, 2012.
[18] S. E. Whang, P. Lofgren, and H. Garcia-Molina, "Question selection for crowd entity resolution," *Proc. VLDB Endowment*, vol. 6, no. 6, pp. 349–360, 2013.
[19] S. Guo, A. G. Parameswaran, and H. Garcia-Molina, "So who won?: Dynamic max discovery with the crowd," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2012, pp. 385–396.
[20] S. B. Davidson, S. Khanna, T. Milo, and S. Roy, "Using the crowd for top-k and group-by queries," in *Proc. 16th Int. Conf. Database Theory*, 2013, pp. 225–236.
[21] A. D. Sarma, A. Parameswaran, H. Garcia-Molina, and A. Halevy, "Finding with the crowd," Stanford University. [Online]. Available: http://ilpubs.stanford.edu:8090/1049/
[22] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom, "CrowdScreen: Algorithms for filtering data with humans," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2012, pp. 361–372.
[23] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang, "CDAS: A crowdsourcing data analytics system," *Proc. VLDB Endowment*, vol. 5, no. 10, pp. 1040–1051, 2012.
[24] A. Krause and C. Guestrin, "A note on the budgeted maximization of submodular functions," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CALD-05–103, Mar. 2005.
[25] S. Khuller, A. Moss, and J. Naor, "The budgeted maximum coverage problem," *Inf. Process. Lett.*, vol. 70, no. 1, pp. 39–45, 1999.
[26] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "Repairs with probabilistic inference," *Proc. VLDB Endowment*, vol. 10, no. 11, pp. 1190–1201, 2017.
[27] V. Ganti, *Data Cleaning*. San Rafael, CA, USA: Morgan & Claypool, 2011. [Online]. Available: http://books.google.com.hk/books?id=V12tXwAACAAJ

[28] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches," *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 3–13, 2000.

[29] B. Kanagal, J. Li, and A. Deshpande, "Sensitivity analysis and explanations for robust query evaluation in probabilistic databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2011, pp. 841–852.

[30] M. van Keulen and A. de Keijzer, "Qualitative effects of knowledge rules and user feedback in probabilistic data integration," *VLDB J.*, vol. 18, no. 5, pp. 1191–1217, 2009.

[31] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the world-wide web," *Commun. ACM*, vol. 54, no. 4, pp. 86–96, 2011.

[32] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 9, pp. 2296–2319, Sep. 2016.

[33] A. I. Chittilappilly, L. Chen, and S. Amer-Yahia, "A survey of general-purpose crowdsourcing techniques," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 9, pp. 2246–2266, Sep. 2016.

[34] D. C. Brabham, "Crowdsourcing as a model for problem solving: An introduction and cases," *Convergence*, vol. 14, no. 1, pp. 75–90, 2008.

[35] T. Malone, R. Laubacher, and C. Dellarocas, "Harnessing crowds: Mapping the Genome of collective intelligence," MIT, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, USA, Research Paper No. 4732–09, February 2009, sloan Research Paper No. 4732–09.

[36] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng, "Truth inference in crowdsourcing: Is the problem solved?" *Proc. VLDB Endowment*, vol. 10, no. 5, pp. 541–552, 2017.

[37] A. Feng *et al.*, "CrowdDB: Query processing with the VLDB crowd," *Proc. VLDB Endowment*, vol. 4, no. 12, pp. 1387–1390, 2011.

[38] E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi, "Crowdsourcing for top-k query processing over uncertain data," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 41–53, Jan. 2016.

[39] R. Meng, L. Chen, Y. Tong, and C. Zhang, "Knowledge base semantic integration using crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1087–1100, May 2017.

[40] A. G. Parameswaran and N. Polyzotis, "Answering queries using humans, algorithms and databases," in *Proc. 5th Biennial Conf. Innovative Data Syst. Res.*, 2011, pp. 160–166.

[41] R. Gomes, P. Welinder, A. Krause, and P. Perona, "Crowdclustering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2011, pp. 558–566.

[42] M. Das, S. Thirumuruganathan, S. Amer-Yahia, G. Das, and C. Yu, "Who tags what? An analysis framework," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1567–1578, 2012.

[43] H. Kaplan, I. Lotosh, T. Milo, and S. Novgorodov, "Answering planning queries with the crowd," *Proc. VLDB Endowment*, vol. 6, no. 9, pp. 697–708, 2013.

[44] Y. Tong, C. C. Cao, and L. Chen, "TCS: Efficient topic discovery over crowd-oriented service data," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 861–870.

[45] S. Goldberg, D. Z. Wang, and T. Kraska, "CASTLE: Crowd-assisted system for textual labeling & extraction," in *Proc. 1st AAAI Conf. Hum. Comput. Crowdsourcing*, 2013.

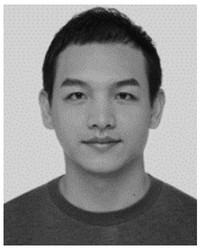
Chen Zhang (Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, in 2015. He is currently a postdoctoral research fellow with the Hong Kong University of Science and Technology as well as an associate professor with the Shandong University of Finance and Economics. His research interests include crowdsourcing and data integration.

Haodi Zhang received the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, in 2016. He is currently an assistant professor with the College of Computer Science and Software Engineering, Shenzhen University and the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University, China.

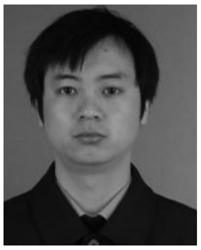
Weiteng Xie received the bachelor's degree from the College of Physics and Electronic Science, Hubei Normal University, Huangshi, China, in 2017. He is currently working toward the master's degree in the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China and the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University, Shenzhen, China.

Nan Liu is currently working toward the undergraduate degree in the College of Engineering Computer Science Engineering Division, University of Michigan, Ann Arbor, Michigan, and is currently visiting Shenzhen University. His research interests include crowdsourcing, crowd-aided uncertainty reduction, path selection, etc.

Qifan Li received the bachelor's degree from the College of Mechanical Engineering, Dongguan University of Technology, Dongguan, China, in 2018. He is currently working toward the master's degree in the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China and the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University, Shenzhen, China.

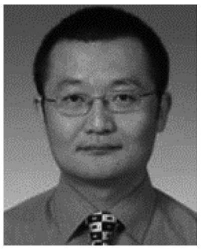


Di Jiang received the PhD degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 2014. He is currently a senior scientist with WeBank AI. His research interests include information retrieval, natural language processing, and massive data management.



Peiguang Lin received the PhD degree from the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, in 2006. He is an associate professor with the School of Computer Science and Technology, Shandong University of Finance and Economics, Shandong, China. His current research interests include massive data processing and integrated, and network information processing.

Kaishun Wu (Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, in 2011. He is currently distinguished professor with the College of Computer Science and Software Engineering, Shenzhen University and the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University, China.



Lei Chen (Fellow, IEEE) received the PhD degree in computer science from the University of Waterloo, Waterloo, Canada, in 2005. He is currently a professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include crowdsourcing over social media, social media analysis, probabilistic and uncertain databases, and privacy-preserved data publishing.