This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2972543, IEEE Transactions on Knowledge and Data Engineering

CrowdChart: Crowdsourced Data Extraction from Visualization Charts

Chengliang Chai† Guoliang Li† Ju Fan‡ Yuyu Luo†
†Tsinghua University, China ‡Renmin University
{ chaicl15@mails., liguoliang@, luoyy18@mails.}tsinghua.edu.cn, fanj@ruc.edu.cn

Abstract—Visualization charts are widely utilized for presenting structured data. Under many circumstances, people want to digitalize the data in the charts collected from various sources (e.g., papers and websites), in oder to further analyze the data or create new charts. However, existing automatic and semi-automatic approaches are not always effective due to the variety of charts. In this paper, we introduce a crowdsourcing approach that leverages human ability to extract data from visualization charts. There are several challenges. The first is how to avoid tedious human interaction with charts and design effective crowdsourcing tasks. Second, it is challenging to evaluate worker's quality for truth inference, because workers may not only provide inaccurate values but also misalign values to wrong data series. Third, to guarantee quality, one may assign a task to many workers, leading to a high crowdsourcing cost. To address these challenges, we design an effective crowdsourcing task scheme that splits a chart into simple micro-tasks. We introduce a novel worker quality model by considering worker's accuracy and task difficulty. We also devise effective task assignment and early-termination mechanisms to save the cost. We evaluate our approach on real-world datasets on real crowdsourced platforms, and the results demonstrate the effectiveness of our method.

Index Terms—Data Visualization, Crowdsourcing, Truth Inference, Task Assignment.

1 INTRODUCTION

Visualization charts are indispensable to visualize structured data due to their perceptual advantages [25], because charts not only help people understand many aspects of data, such as distribution and variation trend [26], [27], but also provide intuitive comparisons for data from different sources [32]. For example, Figure 1 shows a line chart, which visualizes the numbers of crowdsourcing papers at three leading database conferences from 2015 to 2018.

In many cases people want to extract the underlying data from charts in order to further analyze the data, update the charts, or create new charts by integrating data from multiple sources [15], [18]. For example, considering Figure 1, if an analyst wants to do a survey from 2015 to 2019. Suppose that she only has the chart (visualizing the data from 2015 to 2018) and the raw data in 2019. She has to first extract the data from the chart, which is shown in a relational table in the figure. Then, she simply adds a new column containing the information of 2019 to the table and redraws a new chart. Taking another example, a business analyst may be interested in the financial report of Fortune 500 companies, which can be obtained in the form of charts. However, if she wants to manipulate the data in the charts or redesign the charts, it is important to firstly digitalize data from the charts.

- *Chengliang Chai is with the Department of Computer Science, Tsinghua University, Beijing, China. Email: chaicl15@mails.tsinghua.edu.cn.*
- *Guoliang Li is with the Department of Computer Science, Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing, China. Email: liguoliang@tsinghua.edu.cn. Corresponding author.*
- *Ju Fan is with the School of Information, Renmin University, Beijing, China. Email: fanj@ruc.edu.cn.*
- *Yuyu Luo is with the Department of Computer Science, Tsinghua University, Beijing, China. Email: luoyy18@mails.tsinghua.edu.cn.*

Indeed, data extraction from charts has attracted much interest from academia in recent years. Some automatic or semi-automatic chart data extraction tools have been developed [35], [18]. Automatic tools like [35] apply computer vision and machine learning models to first recognize the text in a chart and then infer the underlying data points. However, the performance of such methods is far from satisfactory: accuracy of both the text recognition and data point extraction is around 60% - 70% [18]. Some semi-automatic approaches [35], [18] are also proposed in the HCI community. They first leverage the users to specify some core parts in charts, like drawing the boundary of a chart, identifying the values of the x-axis and y-axis. They then utilize image processing tools to extract data. However, these methods also have several limitations. First, they mostly rely on machine-based algorithms to extract values in charts, which is also not accurate enough, especially for charts with complicated patterns(e.g. many legends or intersections). Second, they are not general and usually have restrictions on the charts to achieve good performance. For instance, [35] and [18] cannot handle line and stacked bar charts.

Fortunately, crowdsourcing can be used to leverage hundreds of thousands of crowd workers to solve large-scale machine-hard tasks. We propose a crowdsourcing chart data extraction framework CROWDCHART that harnesses crowd workers on crowdsourcing platforms like Amazon Mechanical Turk (AMT) [1] to extract data from charts at relatively low cost. We study the following research challenges that naturally arise in the framework.

The first challenge is quality control for crowdsourced chart data extraction. Due to the openness of crowdsourcing, workers often yield relatively low-quality results, or even noise when extracting data from charts. Consider the specific scenario of chart extraction: workers may be careless when reading the numbers in a chart, and their quality may also be affected by visual features of the chart, such as chart type, log-scaled y-axis, etc. For

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2972543, IEEE Transactions on Knowledge and Data Engineering

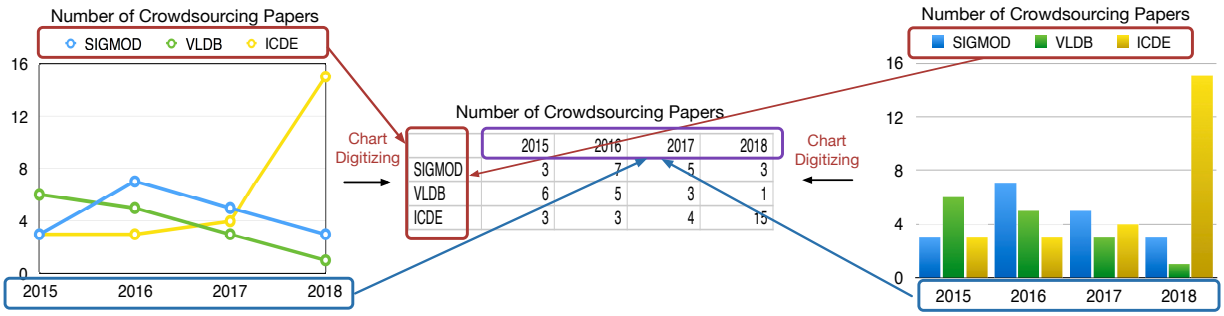

Fig. 1: Example for Chart Extraction

example, stacked bar charts are harder to be recognized than pie charts. Moreover, even for careful workers, their quality may be significantly influenced by a kind of common errors, data series misalignment. For example, in Figure 1, a worker extracts the three data points $[5, 3, 4]$ in 2017 correctly, but she may align 4 to VLDB and 3 to ICDE, leading to data series misalignment. Although there exist some works [36], [42] on crowdsourcing quality control for numerical data, they cannot effectively address the above difficulties. We apply a redundancy-based strategy that assigns a task to multiple workers and aggregates their answers to infer the truth. We introduce a novel truth inference model that incorporates worker accuracy, chart characteristics and the effect of misalignment together into a Gaussian model to measure the worker quality. Based on this, we develop effective techniques for accurate worker quality estimation and truth inference.

The second challenge is how to reduce the crowdsourcing cost. As we may have many charts to extract, a straightforward method in our redundancy-based strategy may generate a large number of tasks, which incur significant monetary cost. To address this, we continuously evaluate the quality of tasks based on current answers using a confidence-based model, and introduce an early-termination strategy that terminates the tasks with high-quality inferred results. Moreover, we devise a dynamic task assignment method that assigns a task to the workers who can mostly improve the task quality. We estimate the updated truth distributions based on existing answers and the upcoming workers' quality before they answer the task, and then select the best worker to assign a task. Our method can not only improve quality, but also early terminates many tasks as early as possible.

Third, it is challenging to design effective crowdsourcing tasks for chart extraction. A straightforward method is to crowdsource an entire chart and ask the worker to submit a relational table. Obviously, such task is overwhelming to workers who are usually good at "micro" tasks (see survey [21]). To address the problem, we design an effective crowdsourcing task scheme that splits a chart into a batch of micro-tasks, each of which extracts a specific part of the chart. Then, we can digitalize the relational table by aggregating crowd answers of the tasks.

To summarize, we make the following contributions.

(1) We propose a crowdsourced chart data extraction framework. To the best of our knowledge, it is the first systematic work that utilizes the crowd to extract data from charts.

(2) We design a truth inference model to derive accurate answers and workers' quality simultaneously.

(3) We develop effective task assignment and early-stopping techniques to largely reduce the crowdsourcing cost.

(4) We evaluate our approach on real datasets on AMT. The results demonstrate its superiority over existing methods.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 formalize the problem and introduce the framework. We propose truth inference and task assignment techniques in Sections 4 and 5 respectively. Section 6 presents experiments and we conclude in section 7.

2 Related Work

This section reviews the related works which fall into two categories: 1) algorithms and tools with respect to data extraction from charts; 2) crowdsourcing techniques including cost control, quality control and crowdsourcing system.

Note that this paper extends our conference version [9], where the main extension is summarized as follows. First, while the conference version only focuses on truth inference, this paper introduces a new task assignment approach, which assigns each incoming worker with a task that achieves the most improvement of the overall quality. We present the proposed approach and evaluate its performance respectively in Sections 5 and 6.4. Second, compared with the short version, we provide more technical details of truth inference, such as the inference algorithm as well as its complexity analysis. Third, for more comprehensive empirical evaluation, we add more baselines and compare our proposed approaches with them in Section 6.

2.1 Data Extraction from Charts

Approaches with respect to data extraction from charts can be categorized into two groups: automatic [35], [44], [17] and semi-automatic [18], [30], [39] tools. Some automatic frameworks [16], [17] only focus on extracting legend keys from charts. [44]. They generate edge maps, vectorize them and utilize rule-based methods to extract keys from the line, bar or pie charts. However, these works can not obtain the numerical data in charts. Moreover, accurate edge maps are hard to retrieve from real-world charts because of large variance of charts quality. AI-Zaidy et al. [6] proposed a method that can only handle bar charts. Savva et al. [35] is a system that automatically extracts data from bar chart and pie chart. However, it only achieves 71% and 64% accuracy on bar chart and pie chart respectively and has many restrictions on styles of charts. For semi-automatic approaches, WebPlotDigitizer [33] is a digitizing tool that extracts data from bar, line, pie charts, which has both automatic and manual mode. Since the automatic mode has a low accuracy because of a simple color detection algorithm, people always tend to use the manual mode to detect values. In this mode, the user has to specify much necessary information, which is tedious and time-consuming, especially for multi-series data. Similarly, [39], [30], [18] are semi-automatic tools that require many manual operations of a user to extract

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2972543, IEEE Transactions on Knowledge and Data Engineering

data from a chart, which has poor scalability. Moreover, these approaches always set many constrains like distinct colors for each series of data, which limits the generalization.

2.2 Crowdsourcing

2.2.1 Truth Inference

The most straightforward method to infer the truth from crowd workers is majority voting(MV). For numerical data in our problem, the MV takes the average as the truth. However, MV regards all workers as equal, which does not always hold in practice. Therefore, some existing state-of-the-art works[23], [24], [36] have been proposed to infer the truth of numerical data by considering workers' quality. They model each answer as a Gaussian distribution $\mathcal{N}(u, \sigma^2)$, where u is regarded as the truth and σ^2, the variance, incorporates the workers' quality. Shan et al. [36] focuses on tabular data, taking both categorical and numerical data into account. Li et al. [23] considers the source (e.g. web pages, wiki) reliability and confidence interval of the variance to obtain a precise estimation. Then the Expectation Maximization(EM) algorithm [13] is applied to derive both the workers' quality and truth. However, these approaches mentioned above do not consider the characteristics of the data extraction task in this paper. We take into many significant factors such as types of charts and types of Y-axis(log-scale or not) into consideration to model the difficulty of each question. Then the difficulty is incorporated into the computation of the Gaussian variance. Also, when inferring the truth, we consider the misalignment of data, which is a common phenomenon in data extraction task.

2.2.2 Task Assignment

Task assignment aims to dynamically select which tasks should be assigned to an incoming worker, which can improve the quality most. Most crowdsourcing systems [29], [14] leverage the crowd's ability to process machine-hard queries like collecting data from the open world, but they assign tasks randomly to workers. Li et al.[19], [20], [22] judiciously select tasks that bring the largest quality improvement. However, they only focus on tasks with categorical answers. In this paper, CrowdChart proposes a confidence-based task assignment model for data extraction tasks. Given an incoming worker, considering her quality, we will assign the task that has low confidence and can be improved much to her.

2.2.3 Other Crowdsourcing Techniques

Recently, crowdsourcing has attracted much attention in academia and industry. To encapsulate the complexity of interacting with the crowd, several crowd-powered database system like Deco [31], CrowdDB [14] and CDB [19], [20] were proposed. They implement and optimize crowdsourcing operators like crowdsourced selection [34], crowdsourced join [10], [11], crowdsourced sort [28], crowdsourced collection [8]. The optimization goal is to trade-off the quality, cost and latency.

3 Overview

In this section, we first formalize the problem of chart data extraction in Section 3.1, and then introduce a CrowdChart framework in Section 3.2.

3.1 Problem Definitions

This paper focuses on extracting *tabular data* from visualization charts using crowdsourcing. Formally, we define fundamental concepts used in our works in this section.

Chart model. Formally, let us consider a collection of charts $\mathcal{C} = \{C_1, C_2, \ldots, C_{|\mathcal{C}|}\}$. For ease of representation, C is interchangeably utilized to denote both a chart and the data appearing in the chart. A chart C consists of a sequence of *legend keys*, which is denoted by $C.K = [k_1, k_2, \ldots, k_m]$. A legend key is used to refer to a group of data visualized in the chart. For example, the chart in Figure 1 has three legend keys, namely `SIGMOD`, `VLDB` and `ICDE`, where each of these keys identifies the numbers of crowdsourcing papers over years of the corresponding conference. Based on this, the data model of a chart is defined as follows.

Definition 3.1 (Chart Data Model). Given a chart C, the data visualized in C consists of the following two elements: (1) A sequence of keys $K = [k_1, k_2, \ldots, k_m]$; (2) a set of tuples $T = \{t_1, t_2, \ldots, t_n\}$, where each tuple $t_i = [t_{i1}, t_{i2}, ..., t_{im}]$ represents the data points in the i-th labels of the horizontal axis, i.e., x-axis. Note that the order of data points in each tuple t_i must be the same with the order of keys in K.

Figure 1 shows an example of chart data with three keys $K = [\texttt{SIGMOD}, \texttt{VLDB}, \texttt{ICDE}]$ and four tuples t_1 to t_4. For example, tuple $t_1 = [3, 6, 3]$ contains the data points corresponding to `SIGMOD`, `VLDB` and `ICDE` in 2015 respectively. It is easy to see that data in a chart naturally corresponds to a relational table where the legend keys are row names, points in x-axis denote the column names and each tuple corresponds to a data column in the table.

The chart model is general for a variety of commonly used charts, including line chart, bar chart (stacked bar chart), and pie chart. Note that the pie chart is a special case with only one tuple containing the ratios or number of various keys.

Crowdsourcing task design. As analyzed previously, the automatic and semi-automatic approaches [35], [6], [12], [44] have limitations on achieving superior performance for chart data extraction. To address the problem, we harness the crowd intelligence to extract data from charts. A straightforward approach is to crowdsource each entire chart to crowd workers and ask them to submit a relational table as illustrated in Figure 1. Although simple, the approach is not effective due to the following reasons. First, it will incur high crowdsourcing latency. Since a chart usually contains many data points, it is time-consuming for a worker to extract them all. Second, as answers from the crowd may be noisy, a commonly used strategy is to first assign a task to multiple workers and aggregate their answers to infer results. However, it is not easy to aggregate entire tables.

To achieve better performance, we introduce a fine-grained approach that splits a chart into a batch of *micro*-tasks[1] to reduce latency and improve quality. Specifically, we design four types of crowdsourcing tasks, as illustrated in Figure 2.

Preprocessing tasks. As quality of chart data extraction may depend on visual features of the chart, we define the following three types of *preprocessing* tasks before extracting the data: 1) chart type classification that categories the chart type, 2) Y-axis classification that identifies whether the y-axis is log-scaled and 3) legend identification that collects a sequence of legend keys.

1. For simplicity, we use micro-task and task exchangeably if context is clear.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2972543, IEEE Transactions on Knowledge and Data Engineering

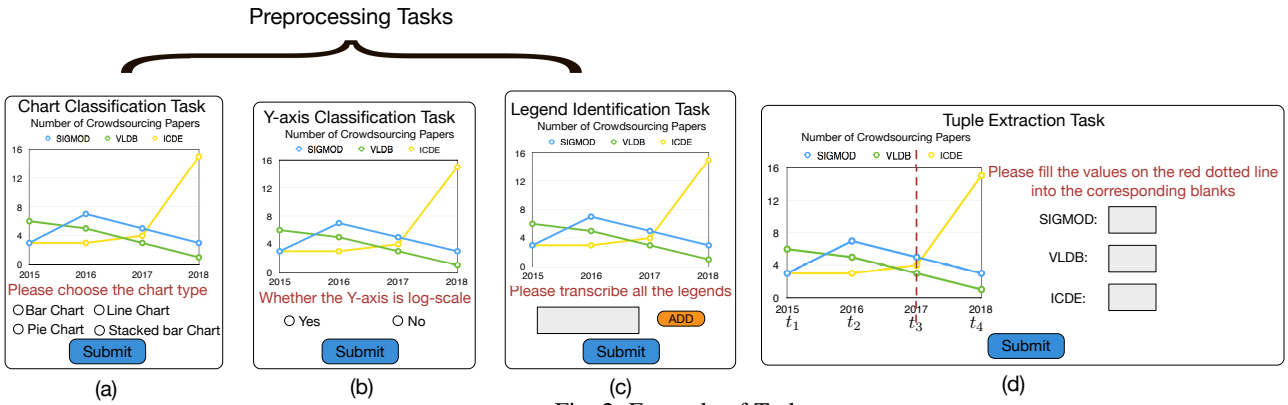

Fig. 2: Example of Tasks

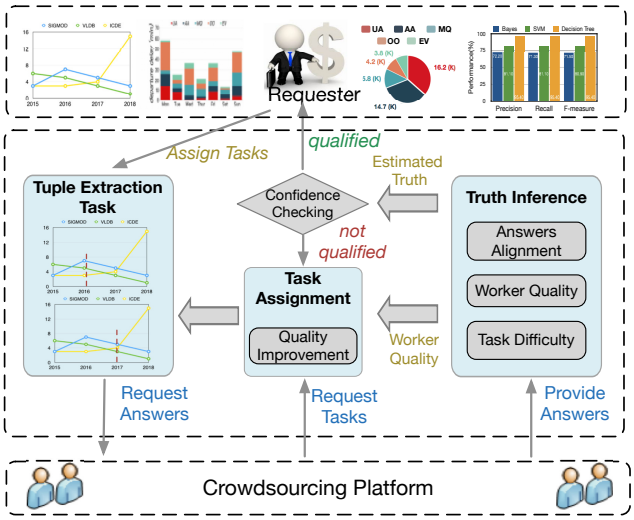

Fig. 3: Framework of CrowdChart

We utilize crowdsourcing to address preprocessing tasks, because these tasks are quite effortless for the crowd who can easily produce good results. Although some simple preprocessing tasks, e.g., chart classification, can be also solved by ML algorithms [40], [38], the accuracy of ML algorithms on more complicated preprocessing tasks is far from satisfactory. For example, as reported in previous studies, the accuracy of recognizing legend keys is normally less than 60% [16], [17].

1) *Chart classification task*. Intuitively, different types of charts have different difficulty levels for data extraction. For example, a pie chart is easier to extract because its visual structure is more compact and it only has one tuple (as discussed above). In contrast, line charts and stacked bar charts are more difficult, because they contain multiple data groups and the exact data numbers are harder to recognize. Thus, we first ask the crowd to classify the chart, which provides guidance for the other crowdsourcing tasks.The task is defined as below.

Definition 3.2 (Chart Classification Task). Given a chart C, a chart classification task is a multiple-choice question to the crowd. The current version of CrowdChart supports four choices, `bar chart`, `line chart`, `pie chart` and `stacked bar chart`, and asks the crowd to choose the one that C belongs to.

An example chart classification task is shown in Fig. 2(a), where a crowd worker will select the choice `Line Chart`.

2) *Y-axis classification task.* Another factor affecting the difficulty is whether y-axis is *log-scale*. Naturally, it is not easy for human to recognize data points given log-scaled y-axis because the numbers are not uniformly distributed. Thus, we also leverage the crowd to identify this issue as one of the preprocessing steps.

Definition 3.3 (Y-axis Classification Task). Given a chart C, y-axis classification task is a binary-choice question to the crowd, where choice `Yes` means y-axis of C is log-scaled and `No` means it is not.

An example task is shown in Fig. 2(b) where a crowd worker will select `No` for the question.

3) *Legend identification task.* Here is the definition of the legend identification task.

Definition 3.4 (Legend Identification Task). Given a chart C, legend identification task is a fill-in-blanks question that asks the crowd to collect a sequence of legend keys, denoted by $K = [k_1, k_2, \ldots, k_m]$.

Fig. 2(c) illustrates an example of legend identification task with three keys `SIGMOD`, `VLDB` and `ICDE` to be collected. One may ask whether it is necessary to use crowdsourcing to identify labels of x-axis, e.g., years in Fig. 2(c). Based on our evaluation, they are easy to identify by automatic algorithms due to their fixed locations and neat arrangement.

To summarize, the above three types of tasks are used for preprocessing steps. First, the result of these tasks, such as legend keys, can be directly used for further data extraction. Second, some results, such as chart type and log-scaled y-axis, can be used for evaluating difficulties of chart data extraction. Moreover, based on our observations, these tasks are quite effortless for the crowd worker who can provide very accurate answers. Thus, we will not elaborate these tasks and focus on a more challenging tuple extraction task as below.

Tuple extraction task. The central task for chart data extraction is to identify the tuples. We design a tuple extraction task that crowdsources an entire tuple instead of putting values of a tuple in different tasks. The reason is two-fold. First, most crowdsourcing platforms charge a fixed amount of commission fee for each HIT. Thus, it is more economical to put a certain number of questions in a task. Second, after a crowd worker understands the chart, extracting a tuple will not introduce much more effort to her compared with an individual value.

Definition 3.5 (Tuple Extraction Task). Given a chart C, a sequence of legend keys $K = [k_1, k_2, \ldots, k_m]$ and a label i in horizontal axis, tuple extraction task is a fill-in-blanks question that collects the i-th tuple $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]$.

For example, Fig. 2(d) shows a tuple extraction task, which aims to collect values corresponding to SIGMOD,VLDB and ICDE respectively. Then the chart in Fig. 2(d) can be divided into $N = 4$ tuple extraction tasks. Note that the order of the sequence in collected tuples is consistent with that of pre-collected legend keys. In addition, it can also be called data extraction task.

3.2 The CROWDCHART Framework

Figure 3 shows the overall framework of CrowdChart. After completing the preprocessing tasks, the requester publishes some tuple extraction tasks. Then given workers who request these tasks, our task assignment module assigns the most appropriate tasks to workers. Next, when a crowd worker submits the answer of a task, we first align the answer based on the worker's quality and answers that have been submitted by other workers corresponding to this task. Then we infer the truth considering the workers' quality and task difficulty using the Expectation-Maximization (EM) algorithm [13]. The inferred truth and workers' quality can also be utilized to guide task assignment in the next iteration. The ultimate goal of CrowdChart is to leverage the truth inference and task assignment to estimate the value in T, so as to obtain a satisfactory result compared with the truth based on a confidence-based model. For values that already have high confidence, an early-stopping method is applied to save the crowdsourcing costs.

The tuple extraction task in this framework is quite challenging because the workers are more error-prone to provide noisy answers. First, the workers need to recognize numeric values from the chart, which is much more difficult than the existing crowdsourcing on categorical values (e.g., positive or negative in sentiment analysis) [43]. For example, one worker may recognize the number of SIGMOD crowdsourcing paper in 2016 as 6 or 7 although the ground truth is 7. There are few existing works [36], [42] that study crowdsourcing numerical values. However, they cannot easily solve the problem as they do not consider difficulty levels such as the chart type, log-scaled y-axis, etc. Second, misalignment is a kind of worker errors that can significantly influence the quality. For example, when extracting data, answers may be misaligned with their legend keys. For t_4 in Fig. 2(d), we can see that the ground truth of the year 2018 is that [3 (SIGMOD), 1 (VLDB), 15 (ICDE)]. For example, if a worker w_1 answers $[3, 2, 15]$, we can take it as an accurate and aligned answer. However, if a worker w_2 gives an answer $[15, 1, 3]$, she is likely to misalign legends "SIGMOD" and "ICDE" carelessly, but she answers the values with perfect quality. If we do not consider such kind of errors, it will cause a high bias on both truth and workers' quality estimation. The above obstacles motivate us to study quality control problems for crowdsourced chart data extraction.

We first define the worker model as follows. We use $\mathcal{W}$ to denote a pool of workers, and $a_i^w = [a_{i1}^w, a_{i2}^w, ..., a_{im}^w]$ to denote a sequence of answers for data points in task t_i by worker $w \in \mathcal{W}$. O_i denotes the set of workers that provide answers for t_i and A_{ij} denotes the set of answers of t_{ij} provided by multiple workers. The overall obtained answers is denoted by A, where $a_{ij}^w \in A$ and $a_i^w, A_{ij} \subset A$. For example, suppose workers w_1, w_2 answer task t_3 with $a_3^{w_1} = [5, 3, 4]$ and $a_3^{w_2} = [6, 3, 4]$. Then $O_3 = \{w_1, w_2\}$ and $A_{31} = \{5, 6\}$. Then we define the truth inference problem as below.

Algorithm 1: Framework of CrowdChart

Input: A collection of charts $\mathcal{C}$.
Output: Estimated truth of each data point $\hat{t_{ij}}$.

```
1 Publish preprocessing tasks for each chart;
2 Sample some charts to train an initial model;
3 while not all data points have been resolved do
4     Select the best task(tuple) t_i for the incoming worker w
      using the task assignment module;
5     Obtain the answers a_i^w from the crowd;
6     Estimate the truth t̂_ij, j ∈ [1, m] using the truth
      inference module;
7     if all points in t_i have high confidence (> α) then
8         Label t_i as resolved ;
9 return Estimated truth of each t̂_ij;
```

Definition 3.6 (Truth Inference). For each point t_{ij}, given workers' answers set A_{ij}, the truth inference problem is to compute a well-estimated value $\hat{t_{ij}}$ for true value t_{ij}^*.

Then, the output of the truth inference model is an estimated truth distribution, from which we can compute the confidence of the estimated truth using the Confidence Checking module. If it already has a high confidence (qualified), we do not need to assign more tasks to save the cost and return the final inferred answer to the requester. For those tasks that have not achieved the required confidence (not qualified), we consider assigning tasks based on the task assignment module defined as below.

Definition 3.7 (Confidence-Aware Task Assignment). Given an incoming worker w, the task assignment module aims to select a task that has not achieved the required confidence and assign it to her, so that the quality of the task can be improved the most.

When a worker w comes, currently, each data point in t_i has an estimated truth with a certain confidence. Given the quality of w and already obtained answers of each task t_i, we can compute an expected quality improvement for each task and assign the one with the highest improvement to w.

Algorithm 1 shows the pseudo-code of CROWDCHART. It first publishes preprocesing tasks for each chart (line 1). Then a small part of charts are sampled to be answered by the crowd workers, so as to derive initial parameters like worker qualities (line 2). Secondly, we use the task assignment module to select the most appropriate task (tuple) whose quality can be improved the most for a crowd worker (line 4). After the crowd answers the task, we infer the truth of each data point in the tuple (line 6). If all points in t_i have high confidence, we label t_i as resolved and no longer ask t_i. Otherwise, we repeat these steps until all points are resolved.

4 TRUTH INFERENCE

In this section, we will introduce how to infer the truth of each data point given multiple workers' answers, considering characteristics of tuple extraction tasks. We first discuss how to model workers

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2972543, IEEE Transactions on Knowledge and Data Engineering

quality and their answers in section 4.1. To further improve the model, difficulties of data points are incorporated into our framework (Section 4.2). In section 4.3, we introduce answers alignment, which is a characteristic quality control methodology in charts extraction task. Finally, the overall inference algorithm is illustrated(Section 4.4).

4.1 Modeling Workers' Answers and Quality

Workers' answers are essential to infer the workers' quality and ground truth, and thus a suitable answers model of the extraction task is necessary. Different from multi-choice tasks, the answers of data extraction tasks are numerical values. For a numerical task, its quality depends on how close it is to the ground truth. For example, suppose a data point has a value 998.5. If a worker's answer is 1000, we take it as a good answer because they are close even if it is not equal to the ground truth. Therefore, we propose that the workers' quality depends on the ratio between their answer and the ground truth rather instead of the difference between them because data have different scales. For example, suppose two data points have ground truth t^*_{ij} = 1000 and $t^*_{i'j'}$ = 100 respectively. If workers w_1 and w_2 provide answers $a^{w_1}_{ij}$=990 and $a^{w_2}_{i'j'}$=90 for two points, we can obviously deduce that w_1 has a higher quality than w_2, though their differences between the ground truths are the same. If they answer $a^{w_1}_{ij}$=980 and $a^{w_2}_{i'j'}$=98 respectively, even if $|a^{w_1}_{ij} - t^*_{ij}| > |a^{w_2}_{i'j'} - t^*_{i'j'}|$, we can estimate that they have nearly the same quality. Therefore, we use the Gaussian distribution to model each answer given by worker w. The distribution takes the ground truth t^*_{ij} as its mean and uses variance to model worker quality, i.e.,

$$\begin{aligned} a^w_{ij} &\sim \mathcal{N}(t^*_{ij}, \phi^w_{ij}) \\ &\sim \frac{1}{\sqrt{2\pi\phi^w_{ij}}} \exp(-\frac{(a^w_{ij} - t^*_{ij})^2}{2\phi^w_{ij}}), \phi^w_{ij} = (\sigma^w_{ij})^2 \end{aligned} \quad (1)$$

where ϕ^w_{ij} is the variance and σ^w_{ij} is the standard deviation. Generally speaking, if w has a good quality, then variance ϕ^w_{ij} will be small because the answer is likely to be close to the ground truth t^*_{ij}. Motivated by this, we use q_w to denote the quality of w and $-\ln q_w \in [0, +\infty]$ to denote the ratio,

$$\sigma^w_{ij} = -\ln q_w \times t^*_{ij}, q_w \in [0, 1]. \quad (2)$$

Since $-\ln q_w$ is a monotonous function, when q_w is close to 1, which indicates a high quality worker, the standard deviation σ^w_{ij} is small because $-\ln q_w$ is close to 0. If $q_w = 0$, which means that the worker w has an extremely low quality, the deviation between the answer and truth approaches to infinity. Therefore, $-\ln q_w$ depicts how far the answer given by w from the truth t^*_{ij}. Since different charts have different scale of values on Y-axis, we utilize the multiplication of $-\ln q_w$ and the truth t^*_{ij} to represent the deviation. The higher the q_w is, the higher probability that the answer given by w is closed to the truth. For example, when $t^*_{i'j'}$ = 100, $q_w = 0.9$ ($\sigma^w_{ij} \approx 0.1 \times 100 = 10$), we can infer that $p(80 < a^w_{ij} < 120) = 0.95$.

4.2 Difficulty of Data Points

The quality of workers' answers does not solely depend on their expertise. Different difficulty levels of the tasks should also be considered, where several factors are taken into consideration for difficulty estimation, including types of charts, the scale of Y-axis and number of legends. Not surprisingly, as discussed in section 3, some complicated charts like line charts and stacked bar charts are challenging even for a human to extract. Also, values along the log-scale Y-axis are always hard for some workers to recognize. Besides these, we also take the number of legends into consideration for difficulty estimation. Intuitively, the larger the number of legends, the more values a crowd worker needs to extract in each task, which leads to more workload and difficulty.

Next, we compute the difficulty of task t_i of a chart C, considering features $\mathbf{x}^1_i$, $\mathbf{x}^2_i$ and $\mathbf{x}^3_i$, which denote the chart classification, scale of Y-axis and legends number respectively. $\mathbf{x}^1_i$ is a one-hot vector with length 4, where we consider `bar chart`, `line chart`, `pie chart` and `stacked bar chart`. For example, $\mathbf{x}^1_i = [0, 1, 0, 0]^{\mathrm{T}}$ indicates that it is a line chart. Concretely, $\mathbf{x}^2_i$ is either 1 or 0, which indicates whether the Y-axis is log-scale or not and $\mathbf{x}^3_i = m$. Then, we use $d_i = \frac{1}{1+e^{-\sum^3_{k=1} \gamma_k \times \mathbf{x}^k_i}}$ to compute the difficulty of task t_i, where γ denotes the weights of different features (γ_1 is a vector with length 4).

Next, we will incorporate the difficulty d_i into the answers formulation in section 4.1. Obviously, the more difficult the task is, the larger the difference between ground truth and workers' answer will be. Therefore, we can rewrite equation 2 as $\sigma^w_{ij} = -d_i t^*_i \ln q_w$. However, the limitation of this formulation is that it assumes that workers' quality is independent with the task difficulty. The assumption may not hold in practice because for some high quality workers, the task difficulty cannot influence their quality, but some malicious workers will provide incorrect answers even for easy tasks. Considering this, we propose equation,

$$\sigma^w_{ij} = -d_i^{\tau_w} t^*_{ij} \ln q_w, \tau_w \in [0, 1]. \quad (3)$$

where the parameter τ_w aims to model how much the task difficulty can impact the worker w's answer. For example, if τ_w is close to 0, d_i will have little influence on the variance since $d_i^{\tau_w}$ is close to 1, which means that the difficulty has no influence on the variance. On the contrary, when τ_w is close to 1, $d_i^{\tau_w}$ is close to d_i, indicating that the difficulty does have impact on the variance.

4.3 Answers Alignment

As illustrated in section 3, misalignment will inevitably happen when extracting data from a large number of charts because in many cases, the visual sequence of data points in the chart cannot match the sequence of these legends in the text region. Especially for line charts, the visual sequence of data points always varies along with the X-axis due to the fluctuation of lines. Thus workers may not be careful enough to capture the variation. Note that this phenomenon cannot be neglected because it will influence both the workers quality and inferred truth. For example, if the misaligned answers are directly used to compute the ground truth, we will derive a truth with high bias, which results in that the worker who answered that task is estimated as a low quality worker. In order to solve the problem, we propose a probability-based solution to align the answers.

Recap that each task has a fixed sequence of legends extracted from the preprocessing task, For example, for all tasks generated from in Fig. 2(d), the sequence $K = [\texttt{SIGMOD}, \texttt{VLDB}, \texttt{ICDE}]$. According to definition 3.6, we aim to infer the truth of data points in the task, i.e., $t^*_i = [t^*_{i1}, t^*_{i2}, ..., t^*_{im}]$ based on the obtained answers. Given answers $a^w_i = [a^w_{i1}, a^w_{i2}, ..., a^w_{im}]$ for task

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2972543, IEEE Transactions on Knowledge and Data Engineering

t_i provided by w, we can generate a set of $m!$ possible sequences S. Each sequence $s_i \in S$ and s_{ij} denotes the j-th answer in sequence s_i.

The alignment problem is to find the sequence that is the most likely to match t_i^*. In other words, given the truth t_i^* and the worker's variance σ_w, we want to compute the probability of each possible sequence. However, since we do not know the ground truth, we use current estimated truth $\hat{t_i} = [\hat{t_{i1}}, \hat{t_{i2}}, ..., \hat{t_{im}}]$ to compute the probability, $p(s_i, \hat{t_i}) = \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi\phi_{ij}^w}} \exp(-\frac{(s_{ij}-\hat{t_{ij}})^2}{2\phi_{ij}^w})$.

Since the number of legends in a chart is small (less than 5 in most time) in practice, it is not expensive to enumerate $m!$ sequences and select the one with the largest probability. Therefore, we select the sequence s^* with the largest probability $s^* = \arg\max_{s_i \in S} p(s_i, \hat{t_i})$.

For example, given a task $\hat{t_i} = [100, 80, 50]$, a worker $w(\sigma_{ij}^w = 5\% t_{ij}^*)$ provides answer $a_i^w = [55, 75, 90]$ and we want to align a_i^w with $\hat{t_i}$. There are 6 possible sequences with respect to $a_i^w, (s_1 = [55, 75, 90], s_2 = [55, 90, 75]), s_3 = [75, 55, 90], s_4 = [90, 55, 75], s_5 = [75, 90, 55], s_6 = [90, 75, 55])$. Take s_6 as an example, $p(s_6, \hat{t_i}) = 0.01 \times 0.04 \times 0.02$, which is the largest one among $s_1 - s_6$. So we align a_i^w as [90,75,55].

Here we consider the cold start problem. When we start the crowdsourcing task and obtain very few answers, the confidence of the estimated truth is low. So it is inaccurate to align based on them. Considering an extreme case, suppose the first answer for task t_i is misaligned. If we align the following answers based on it, all of them will be misaligned. Therefore, for the first two answers, we will not start the alignment operation. After that, we will align the answers using the above probabilistic model.

4.4 Inference Algorithm

In this section, we will infer the truth and workers' quality based on current obtained answers using the maximum likelihood estimation. Note that when we collect an answer, we first align the answer and then do the inference, and thus we do not need to consider the misalignment problem at the inference step. Given parameters $\theta = \{\theta_w\}, \theta_w = \{\gamma, q_w, \tau_w\}$, which will be estimated in the M-step, the objective function is to maximize the likelihood of workers' answers,

$$\arg\max_{\theta} P(A|\theta) = \arg\max_{\theta} \sum_{\mathcal{T}^*} P(A, \mathcal{T}^*|\theta), \tag{4}$$

where $\mathcal{T}^* = \{t^*\}$ is the truth of all the data points, which is taken as the hidden variable. To solve this optimization problem, we use the Expectation Maximization (EM) algorithm [13], which iteratively computes the truth distribution $\mathcal{T}^*$ and parameters θ. Next, we provide the details of the E-step and M-step.

Expectation Step. In the E-step, given the values of θ and the observation A_{ij}, we compute the posterior probability of the hidden variable $\mathcal{T}^*$ as following,

$$\begin{aligned} &P(t_{ij}^* = z|A_{ij}, \theta) \propto \\ &\prod_{w \in O_{ij}} P(a_{ij}^w|t_{ij}^*, \theta_w) \times P(t_{ij}^* = z) \\ &= \prod_{w \in O_{ij}} \frac{1}{\sqrt{2\pi\phi_{ij}^w}} \exp(-\frac{(a_{ij}^w - t_{ij}^*)^2}{2\phi_{ij}^w}) \times P(t_{ij}^* = z), \end{aligned} \tag{5}$$

where $P(t_{ij}^* = z) \sim \mathcal{N}(t_{ij}^0, \phi_{ij}^0)$ is the priori distribution of the truth t_{ij}^*. We use average(variance) of answers in A_{ij} as the mean(variance) of the priori distribution, i.e., $t_{ij}^0 = \frac{\sum_{w \in O_{ij}} a_{ij}^w}{|A_{ij}|}$, $\phi_{ij}^0 = \frac{\sum_{w \in O_{ij}} (t_{ij}^0 - a_{ij}^w)^2}{|A_{ij}|}$. As equation 5 shown, $P(t_{ij}^* = z|A_{ij}, \theta)$ is represented as the products of some Gaussian distributions, so it also follows Gaussian distribution, denoted as $t_{ij}^* \sim \mathcal{N}(\mu_{ij}, \phi_{ij})$. We use $f(t_{ij}^*)$ to denote the probability density function of t_{ij}^*. Since for Gaussian distribution, $f'(\mu_{ij}) = 0$ and $f''(\mu_{ij}) \propto \phi_{ij}^{-2}$, which can be utilized to compute the mean and variance of the distribution of t_{ij}^*.

$$\begin{aligned} \mu_{ij} &= \frac{t_{ij}^0}{\phi_{ij}^0} * \phi_{ij} + \sum_{w \in O_{ij}} \frac{a_{ij}^w}{\phi_{ij}^w} * \sigma_{ij} \\ \phi_{ij} &= \frac{1}{\frac{1}{\phi_{ij}^0} + \sum_{w \in O_{ij}} \frac{1}{\phi_{ij}^w}} \end{aligned} \tag{6}$$

Therefore, given the parameter θ, we can compute ϕ_{ij}^w using equation 3. Since we do not know the ground truth t_{ij}^*, we utilize the estimated mean μ_{ij} in the last iteration as the truth to derive the value of ϕ_{ij}^w. Thus we obtain the distribution of the truth t_{ij}^*. We can see from equation 6 that the mean μ_{ij} is computed by the weighted average of workers' answers. Intuitively, the answer of a higher quality worker (with a small ϕ_{ij}^w) will be assigned to a higher weight and trusted more. ϕ_{ij} is a normalized term. Next, we will illustrate how to utilize the truth distribution to estimate these parameters.

Maximization Step. In the M-step, given the estimated distribution of the truth t_{ij}^*, we compute optimal values for parameters θ so that the expectation of the joint likelihood of the observation (equation 7) is maximized,

$$\begin{aligned} Q(\theta) &= E_{\mathcal{T}^*}[\log P(A, \mathcal{T}^*|\theta)] \\ &= \sum_i \sum_j E_{t_{ij}^*}[\log P(t_{ij}^*) + \sum_{w \in O_{ij}} \log P(a_{ij}^w|t_{ij}^*, \theta_w)] \end{aligned} \tag{7}$$

In equation 7, the term $E_{t_{ij}^*}[\sum_{w \in O_{ij}} \log P(a_{ij}^w|t_{ij}^*, \theta_w)] = \sum_{w \in O_{ij}} -(\frac{1}{2}\log 2\pi\sigma_{ij}^w + \frac{(a_{ij}^w - u_{ij})^2 + \sigma_{ij}}{2\phi_{ij}^w}$, Similarly, the term $E_{t_{ij}^*}[\log P(t_{ij}^*)] = -(\frac{1}{2}\log 2\pi\sigma_{ij}^0 + \frac{(t_{ij}^0 - u_{ij})^2 + \sigma_{ij}}{2\phi_{ij}^0}$. And thus,

$$\begin{aligned} Q(\theta) &= -(\frac{1}{2}\log 2\pi\sigma_{ij}^0 + \frac{(t_{ij}^0 - \mu_{ij})^2 + \sigma_{ij}}{2\phi_{ij}^0}) \\ &+ \sum_{w \in O_{ij}} -(\frac{1}{2}\log 2\pi\sigma_{ij}^w + \frac{(a_{ij}^w - \mu_{ij})^2 + \sigma_{ij}}{2\phi_{ij}^w}) \end{aligned} \tag{8}$$

Then gradient descent is utilized to find the values of θ that lead to a locally optimal solution for $Q(\theta)$.

We summarize the process of truth inference in Algorithm 2. Suppose a worker w provides answers a_i^w for task t_i. Then we align those answers (line 1) and then add each answer into A_{ij} (line 3). Next we initialize a truth based on current answers and previous parameters based on equation 6 (line 4). Then the EM algorithm is applied to compute the parameters (line 6) and truth (line 7) until converge. At last, the inferred truth $\hat{t_{ij}}$ is returned, where $\hat{t_{ij}} = \mu_{ij}$.

Time Complexity Analysis. For each data point t_{ij} in a task, we have to iterate workers in O_{ij} to complete the E-step. Since each task contains m points, the complexity is $O(m|O_{ij}|)$. In the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2972543, IEEE Transactions on Knowledge and Data Engineering

Algorithm 2: Inference Algorithm

Input: Current answers set $\{A_{i1}, A_{i2}, ..., A_{im}\}$, new answers a_i^w for task t_i and current parameters θ
Output: Estimated truth $\hat{t_{ij}}$

```
Align answers in a_i^w;
for j from 1 to m do
    A_ij ← A_ij ⋃ {a_ij^w};
    Initialize t̂_ij using θ by equation 6 ;
    while not converged do
        Update parameters θ to maximize the equation 7 ;
        Compute μ_ij and φ_ij of t_ij in task t_i with updated
        parameters using equation 6;
return t̂_ij;
```

M-step, we also need to loop for each data point and workers who answer the task. Moreover, suppose the gradient descent of each parameter takes g steps to converge and the EM algorithm takes e steps to converge. The total complexity is $O(egm|O_{ij}|)$. Since e and g are const in practice, less than 50, the time complexity is linear to the number of answers concerning the task.

5 Task Assignment

In this section, we study how to select a task for an incoming worker, where we have two problems. One is whether every task needs to be asked. For some tasks, since they have been answered by enough number of workers or a few high quality workers and thus derive high confidence, we do not need to ask more. For other tasks with low confidence, we should assign them to incoming workers to obtain more confident results (See section 5.1). Second, given an incoming worker w, we need to assign her a task that has not achieved a high confidence. We aim to select the task whose quality can be improved the most. To this end, for each task, we estimate the expected distribution of the truth if the task is answered by w. Based on the current estimated distribution, we can compute a quality improvement for each task. Then we select the best one to assign to w (See section 5.2).

5.1 Confidence-based Model for Early Stopping

Given the truth distribution of a data point $t_{ij}^* \sim \mathcal{N}(\mu_{ij}, \sigma_{ij})$ obtained through the truth inference algorithm, we can compute the confidence if we regard μ_{ij} as the answer. We adopt the $(1-\alpha)$ confidence interval for the estimated truth, where $1-\alpha$, also known as the confidence level, is usually near to 1 such as 90%, 95%. We will trust the answer and stop to assign questions with respect to the task if it satisfies,

$$P((1-b)\mu_{ij} < t_{ij}^* < (1+b)\mu_{ij}) > 1-\alpha \tag{9}$$

which gives the $(1-\alpha)$ confidence interval of t_{ij}^* as $r = [(1-b)\mu_{ij}, (1+b)\mu_{ij}]$, where b controls the width of the interval and is always small, like $b = 0.1$. For example, as Fig. 4(a) shows, the likelihood that the truth lies in a small range(r) is low, so we have to ask more to satisfy the above confidence requirement. However, in Fig. 4(b), the likelihood is much higher because of the small variance of the estimated distribution, which satisfies the confidence requirement in equation 9. Therefore, for each data point, if the distribution of the estimated truth satisfies equation 9, we will not ask more to save the cost because the estimated value has already had a high confidence. Otherwise, we will assign a task containing this data point to an appropriate worker in order to satisfy the confidence requirement as soon as possible. The reason why we use a small proportion of μ_{ij} ($2b\mu_{ij}$) as the confidence interval rather than a range with fixed length is that data points have different scales. For example, given two points with $\mu = 10$ and $\mu' = 1000$ respectively and a range with fixed length 10, obviously the likelihood $P(5 < t < 15)$ for μ is much higher than that of $P(995 < t < 1005)$ for μ'. Therefore, we cannot use a range with fixed length to measure the confidence of data with different scales.

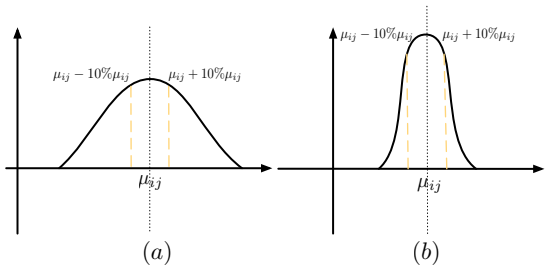

Fig. 4: Examples of Confidence Interval

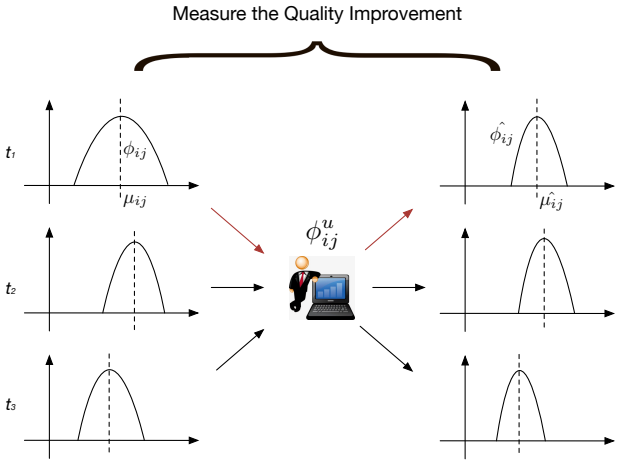

Fig. 5: Example of the Task Assignment Algorithm.

5.2 Task Assignment Algorithm

This section presents our approach of assigning tasks to each incoming crowdsourcing worker. The basic idea is illustrated in Fig. 5. Let us consider three tasks t_1, t_2 and t_3, and the estimated distribution of each task is shown in the left part of the Fig. 5 [2]. Note that the estimated distribution is obtained by the truth inference technique presented in the previous section, i.e., $t_{ij}^* \sim \mathcal{N}(\mu_{ij}, \phi_{ij})$. Now suppose that a worker u requests for a new task. We examine each task and compute the updated estimated distribution (in the right part of the Fig. 5) if the worker's answer a_{ij}^u is included into the task. Then, we compute the quality improvement based on the distributions before and after including the answer. We select the task with the most improvement on answer quality, e.g., the first task shown in Fig. 5. To fulfill the above process, we devise methods for distribution estimation update and quality improvement computation, which are described as follows.

Updated Distribution Estimation. Since a_{ij}^u is not known in advance, we have to generate all possible answers according to the current distribution, and then compute the expected distribution of the truth, denoted by $t_{ij}^* \sim \hat{\mathcal{N}}(\hat{\mu_{ij}}, \hat{\phi_{ij}})$ as the

2. For simplicity, we consider each task only has one data point.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2972543, IEEE Transactions on Knowledge and Data Engineering

Algorithm 3: Task Assignment Algorithm

Input: Incoming worker w, current distribution $\mathcal{N}(\mu_{ij}, \phi_{ij})$
Output: Task t assigned to worker w.

```
1 for t_i in T do
2     for j from 1 to m do
3         if I_f(t*_ij) = 0 then
4             Compute the expected distribution of t̂*_ij given
              the incoming worker w.
5     Compute the quality improvement I(t_i).
6 Compute the t_i with the highest quality improvement.
7 return t_i;
```

updated distribution, where $\hat{\mu_{ij}}$ and $\hat{\phi_{ij}}$ denote the updated mean and variance respectively. The distribution is computed by $E_{a^u_{ij}}[P(t^*_{ij}|A_{ij} \bigcup \{a^u_{ij}\}, \theta)]$.

Theorem 5.1. $\hat{\mu_{ij}} = (\frac{t'_{ij}}{2\phi^u_{ij}} + \frac{t^0_{ij}}{\phi^0_{ij}} + \sum_{w \in O_{ij}} \frac{a^w_{ij}}{\phi^w_{ij}}) * \hat{\sigma_{ij}}$ and $\hat{\phi_{ij}} = (\frac{1}{2\phi^u_{ij}} + \frac{1}{\phi^0_{ij}} + \sum_{w \in O_{ij}} \frac{1}{\phi^w_{ij}})^{-1}$

Proof 5.1. We prove the Theorem 5.1 as following,

$$E_{a^u_{ij}}[P(t^*_{ij}|A_{ij} \bigcup \{a^u_{ij}\}, \theta)]$$
$$= \int_{-\infty}^{+\infty} P(t^*_{ij}|A_{ij} \bigcup \{z\}, \theta) P(a^u_{ij} = z) dz$$

Based on equation 1, we know that $a^u_{ij} \sim \mathcal{N}(t^*_{ij}, \phi^u_{ij})$. Since we do not know the truth, we use $\hat{t_{ij}}$ and ϕ^u_{ij} to generate the answer of the incoming worker u. Thus, the above equation equals to (c denotes a const in the equation),

$$\frac{1}{\sqrt{2\pi\phi^u_{ij}}} \exp(-\frac{(z - t^*_{ij})^2}{2\phi^u_{ij}}) \frac{1}{\sqrt{2\pi\phi^u_{ij}}} \exp(-\frac{(z - \hat{t_{ij}})^2}{2\phi^u_{ij}}) dz$$
$$= -c \times exp(-\sum_{w \in O_{ij}} \frac{(t^*_{ij} - a^w_{ij})^2}{2\phi^w_{ij}}) \times$$
$$\int_{-\infty}^{+\infty} exp(-\frac{(z - \frac{t^*_{ij} + \hat{t_{ij}}}{2})^2 + \frac{(t^*_{ij} - \hat{t_{ij}})^2}{4}}{\phi^u_{ij}}) dz$$

Since $\int_{-\infty}^{+\infty} exp(-\frac{(z - \frac{t^*_{ij} + \hat{t_{ij}}}{2})^2}{\phi^u_{ij}}) dz$ is a const, we have,

$$-c \times exp(-\sum_{w \in O_{ij}} \frac{(t^*_{ij} - a^w_{ij})^2}{2\phi^w_{ij}} - \frac{(t^*_{ij} - \hat{t_{ij}})^2}{4\phi^u_{ij}})$$

In order to obtain the mean and variance, we use the similar method as equation 6 to compute the derivative. Then we prove theorem 5.1.

Quality Improvement Computation. In this step, we select the most appropriate task for the incoming worker u considering the quality improvement. Since each task contains m data points, we need to take the quality improvement of all of them into consideration. For each data point, we use the entropy function [37] to measure the uncertainty of the estimated truth, i.e., $H(t^*_{ij}) = \frac{1}{2}\ln(2\pi e \phi_{ij})$. Considering all data points in task t_i, the entropy is computed as $H(t^*_i) = \sum_{j=1}^{m} H(t^*_{ij})(1 - \mathbb{I}_f(t^*_{ij}))$, where $\mathbb{I}_f(t^*_{ij}) = 1$ if t^*_{ij} satisfies equation 9 and 0 otherwise. Similarly, given an incoming worker u, the updated entropy of truths in task t^*_i is denoted as $H(\hat{t^*_i})$. $H(\hat{t^*_i}) = \sum_{j=1}^{m} H(\hat{t^*_{ij}})(1 - \mathbb{I}_f(t^*_{ij}))$ and $H(\hat{t^*_{ij}}) = \frac{1}{2}\ln(2\pi e \hat{\phi_{ij}})$. The entropy captures the amount of inconsistency, i.e., the lower H is, the more consistent the answers are, and the higher quality will be achieved.

Overall, for task assignment, we can leverage the coming worker w's quality and the task t_i's current distribution $\mathcal{N}$ to estimate the expected distribution $\hat{\mathcal{N}}$ using theorem 5.1. We use $\mathcal{I}(t_i) = H(t^*_i) - H(\hat{t^*_i})$ to denote the expected quality of improvement if worker u answers the task t_i. Thus the task with the highest improvement in quality is selected, i.e., $\arg\max_{t_i \in \mathcal{T}} I(t_i)$.

Algorithm 3 shows the algorithm of task assignment. Given an incoming worker, we need to select the task that can bring the most quality improvement and assign to her (line 6). To this end, we enumerate every task and compute the improvement (line 5). When computing each task, we do not need to consider the data point(s) that have satisfied equation 9 in it(line 4). Apparently, for the task that all data points in it have satisfied equation 9, we do not assign it anymore.

Time Complexity Analysis. In the assignment algorithm, we have to iterate all unresolved tasks and select the best one, so the complexity is $O(m|\mathcal{T}|)$. For each task, given the incoming worker w, we will compute the expected distribution using theorem 5.1, so as to derive the quality improvement. Therefore the overall complexity is $O(vm|\mathcal{T}|)$, where v is the average number of answers for each task. Since v and m are usually small in practice, less than 10, the complexity is linear to the number of tasks.

6 Evaluation

We have implemented CrowdChart using Python 3.6 on a Ubuntu server Intel 2.4GHz Processor and 32GB memory on top of CrowdOTA [41], which is an online task assignment framework built on AMT. This section evaluates the performance of CrowdChart. In Section 6.1, we introduce our two real datasets to be evaluated. Then some basic experimental settings are discussed in Section 6.2. Next, we discuss the experiment results of our truth inference and task assignment module respectively in Sections 6.3 and 6.4. Then, we evaluate effect of some parameters in Section 6.5 and investigate efficiency in Section 6.6.

TABLE 1: Datasets.

	$\mathcal{C}$	#Data points	#Line Chart	#Bar Chart	#Pie Chart
Paper	75	890	40	35	0
Web	180	2550	110	50	20

6.1 Datasets

We use two real datasets to evaluate our approach, the details of which are summarized in Table 1. (1) `Paper`: We extract 75 charts (including 890 data points in total) from several research papers, which consist of 40 line charts and 35 bar charts (including 5 stacked bar charts). The ground truth is the data used to draw those charts. (2) `Web`: We crawl 180 charts from the web (including 2550 data points in total), which include 110 line charts, 50 bar charts (including 8 stacked bar charts) and 20 pie charts. Specifically, for ease of collecting ground-truth, the charts are crawled from the websites with meta-data [2], [3], [4], [5].

6.2 Experimental Settings

Crowdsourcing settings. We conduct experiments on the popular crowdsourcing platform, Amazon Mechanical Turk (AMT). As

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2972543, IEEE Transactions on Knowledge and Data Engineering

task assignment is not natively supported by AMT, we leverage the framework of CrowdOTA [41], which utilizes the "External Questions" function of AMT, builds a web server and interacts with AMT using the APIs for assigning tasks. For preprocessing tasks, we include the three kinds of task in a single human intelligence task(HIT) and pay $\$0.1$ for the HIT. For tuple extraction tasks, an HIT is used to extract one tuple t_i like Fig. 2(d), which costs $\$0.05m$, where m is the number of values in t_i. For example, we set the price of task shown in Fig. 2(b) as $\$0.15$.

Preprocessing tasks. Before extracting tuples, we first publish all preprocessing tasks to the crowd. We report the result quality of these tasks here. On both datasets, the result quality of chart classification task and y-axis classification task achieves an accuracy of 99%. This validates our claim that such tasks are very easy for workers. For legend identification task, the accuracy on both datasets is 95%. The main errors are that workers miss one or two characters for some legend keys when transcribing the text, which will not affect the following tuple extraction much.

Warm-up step. To cope with the cold start problem, we first pick a proportion(20%) of tasks and assign each of them to 5 workers. After that, we use the truth inference approach in section 4 to infer the truth of those questions as well as the parameters set θ. If some of tasks have satisfied the confidence requirement, we return it to the requester. For remaining tasks, we combine them together with other 80% tasks to the next online truth inference and task assignment step. In this step, we initialize the parameters using θ obtained in the warm-up step and update them with online process going. Note that although we have the warm-up step, we can not avoid the cold start problem completely. This because the tasks may be requested by new workers who are not involved in the warm-up step. We assign the average of workers' quality (q_w, τ_w) obtained in the training step as the quality of new workers, which can be updated during the inference process.

Evaluation metrics. In the evaluation, we mainly compare the cost and quality of CrowdChart with other baselines. (1) Cost. We utilize the monetary cost to evaluate the cost of different approaches. Note that, for different methods, the cost used for preprocessing tasks is the same, and thus we do not report this part. (2) Quality. For quality, we use the metric Mean Normalized Absolute Distance `MNAD` [24] to measure the overall absolute distance from each approachs results to the ground truths, which indicates how close the results are to the ground truths. As different data values may have different scales, we normalize the distance based on the method proposed in [24].

6.3 Evaluation on Truth Inference

This section evaluates the truth inference module in CrowdChart, compared with the following state-of-the-art approaches with the focus on inferring the truth of numeric data.

(1) Average (`AV`): Average is a simple and intuitive method to tackle continuous answers. Given several answers of a data point by multiple workers, it computes the average as the truth.

(2) `GTM` [42]: `GTM` is a truth discovery framework for numeric data, which considers the source reliability (workers' quality) and utilizes the EM algorithm to infer the truth.

(3) `T-Crowd` [36]: `T-Crowd` is a crowdsourcing framework for tabular data, including both categorical and numeric data. In our scenario, we do not have categorical data, so we only compare with its technique designed for continuous data.

We compare CrowdChart with `AV`, `GTM` and `T-Crowd` respectively. For a fair comparison, we utilize the same assignment module to assign tasks for all methods and leverage their different truth inference approaches to infer the truth. Specifically, as `GTM` and `T-Crowd` also use Gaussian distribution to model the truth, we can also compute the confidence using equation 9. For `AV`, we use the mean and variance of the answers of a task to compute a Gaussian distribution. Then, we apply the same task assignment method on them, including the early-stopping strategy. We set $\beta = 0.1$ and vary the confidence level from 0.85 to 0.95 to test the performance.

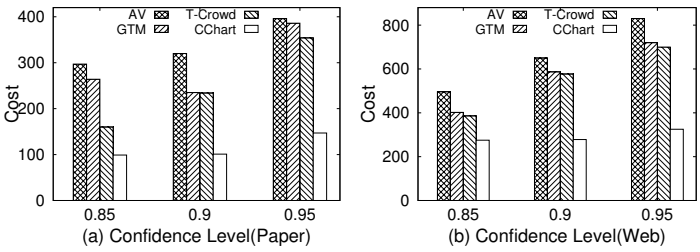

Fig. 6: Evaluation on Truth Inference: Cost

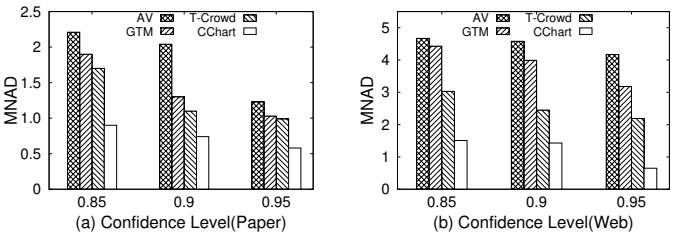

Fig. 7: Evaluation on Truth Inference: Quality

Figures 6 show the evaluation on crowdsourcing cost, which is the monetary cost defined in the evaluation metric above. We can see from Fig 6(a) that CrowdChart saves more than two times of cost compared with other state-of-the-art works when achieving the same confidence level on the `Paper` dataset. For example, when the confidence level is 0.9, CrowdChart incurs a cost of \$101 while `AV`, `GTM` and `T-Crowd` use \$320, \$235 and \$234 respectively. This because CrowdChart will align the answers, which narrows down the variance of inferred answers and improve the workers' quality estimation. Thus CrowdChart can achieve the confidence requirement with much less number of tasks. Moreover, we can see that with increase of the confidence level, the cost grows up. This is reasonable because we should ask more to keep higher confidence. Similar observation can also be found on the `Web` dataset (Fig. 6(b)).

Fig. 7 shows the result on quality. When confidence level is 0.9, we can see from Fig. 7(a) that on dataset `Paper`, CrowdChart achieves the best quality, with the MNAD of 0.74, which improves 30% compared with `T-Crowd` with the second smallest MNAD (1.1). CrowdChart also outperforms `GTM` a lot because CrowdChart considers the answers alignment and task difficulty. For instance, when the confidence level is 0.95, CrowdChart has an MNAD of 0.58 while `AV` and `GTM` are 1.23 and 1.03 respectively. `AV` has the worst quality because it does not consider the workers' quality and task's difficulty. `GTM` performs better than `AV` because it considers the task's difficulty. The significant improvement of CrowdChart is attributed to the truth inference techniques, such as answer alignment and worker model.

6.4 Evaluation Task Assignment

In this section, we evaluate the task assignment module in CrowdChart comparing with several baselines.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2972543, IEEE Transactions on Knowledge and Data Engineering

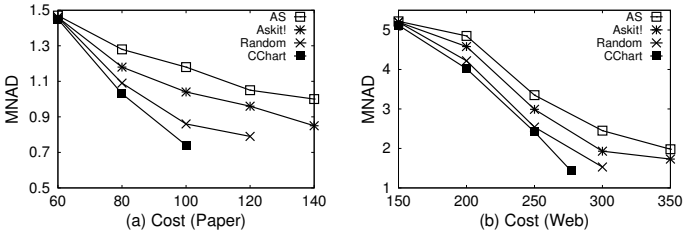

Fig. 8: Evaluation on Task Assignment

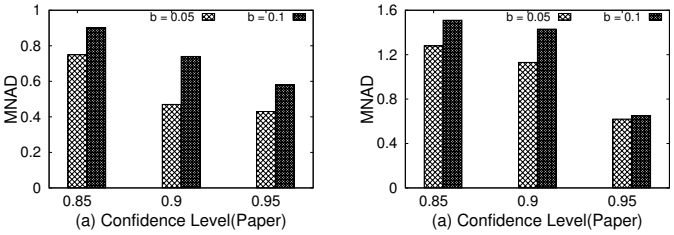

Fig. 10: Evaluating b: Quality

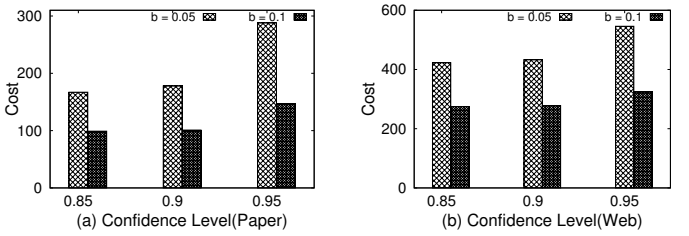

Fig. 9: Evaluating b: Cost

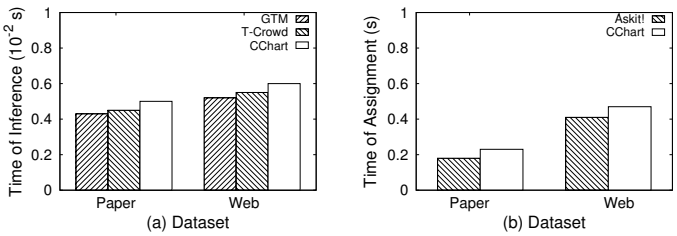

Fig. 11: Evaluation on Efficiency

(1) Assign three questions per task(AS): AS is an algorithm that assigns each task to three different workers without considering the confidence.

(2) Randomly assignment (Random): Random computes the confidence of the estimated truth of each task and it adopts an early-stopping strategy to stop to assign tasks to high confidence task. For those tasks are not terminated, Random assigns tasks randomly.

(3) Askit! [7]: Askit! computes the entropy of each task to measure the uncertainty of it, and then assign the task with the highest uncertainty to the incoming worker.

For fair comparison, we use the same truth inference algorithm in CrowdChart to infer the truth and test different task assignment strategies. We set $b = 0.1$, confidence level as 0.9 and vary the cost (number of asked questions) for evaluation.

Fig. 8 shows the performance on quality and cost in task assignment. We can see from Fig. 8(a) that on the Paper dataset, when the cost is low, the methods have a similar MNAD. For example, when the cost is \$60, they have MNAD around 1.47. This because we have not assigned many tasks and the advantages of our algorithm have not been revealed. However, with the number of obtained answers accumulating, e.g., when the cost is \$101, CrowdChart early stops because it has achieved the confidence requirement. At that time, CrowdChart achieves an MNAD of 0.74, while Random, Askit! and AS have an MNAD of 0.86, 1.04 and 1.18 respectively. Askit! and Random perform better than AS because they consider the uncertainty and the confidence respectively. CrowdChart further outperforms Askit! and Random because it assigns the tasks that can improve the quality most to the workers. Random achieves the quality requirement when the cost is \$101 but has a higher MNAD than CrowdChart when it stops with the same confidence level (0.9) because it does not consider any task assignment strategy about whom to ask. Askit! and AS achieve further higher MNAD with even more costs. On Web dataset, CrowdChart still outperforms others. For example, when the cost is \$278, CrowdChart achieves the quality requirement and has an MNAD of 1.43 while Random, AS and Askit! are 2.18, 2.48 and 2.96 respectively.

6.5 Effect of width of confidence interval (b)

In this part, we evaluate the influence of parameter b on CrowdChart in Fig. 9 and 10. We can see that in Fig. 9(a), on Paper dataset (the confidence level is 0.9), CrowdChart costs more when $b = 0.05$ (\$167) compared with $b = 0.1$ (\$99), because we need to ask more in order to achieve a higher confidence requirement. Moreover, on Web dataset, we can see that when the confidence level is 0.95, CrowdChart costs \$325 if $b = 0.1$ while \$546 if $b = 0.05$. When it comes to the quality, we can see from Fig. 10(a) that CrowdChart achieves a higher quality when $b = 0.05$ because it requires the estimated truth more closer to the ground truth. For example, when the confidence level is 0.95, on Paper dataset, CrowdChart has MNAD of 0.43($b = 0.05$) compare with 0.58($b = 0.1$). Similarly, on Web dataset, when the confidence level is 0.95, we can see that CrowdChart costs about two times more when $b = 0.05$ than $b = 0.1$ and the quality is improved from 0.65 to 0.62. Based on the observations, we choose $b = 0.1$ as its default value, because it can save much cost while merely damaging the quality.

6.6 Evaluation on Efficiency

We evaluate the efficiency of our truth inference module in Fig. 11(a), compare with GTM and T-Crowd. We set $b = 0.1$ and the confidence level as 0.9 to evaluate the efficiency. We record the time of inferring the truth of each data point, compute the average and report the results. The reason why we do not compare with AV is that it just computes the average, which is a const time complexity. We can see from the figure that our algorithm can infer the truth of each data point within 10ms, which has a similar efficiency with GTM and T-Crowd. For task assignment, we evaluate the efficiency of assigning each online task. AS and Random assign tasks randomly so we do not compare with them. We can see from Fig. 11(b) that CrowdChart can assign a task within 0.5 second, which is similar to that of Askit!. Therefore, our algorithm has a high efficiency and can be used in practice scenarios.

7 Conclusion

In this paper, we propose a crowdsourced chart data extraction framework CrowdChart, which aims to extract the underlying data from the charts into a relational table, including the schema of rows and columns . We use well-designed tasks to interact with the crowd workers. To improve the quality, we design a truth inference model to derive accurate answers and workers' quality simultaneously. Moreover, we develop effective task assignment and early-stopping techniques to reduce the monetary cost. Finally,

we evaluate our approach on real datasets on AMT and the results demonstrate its superiority over existing methods.

Acknowledgement. This work was supported by the 973 Program of China (2015CB358700), NSF of China (61632016, 61521002, 61661166012, 61602488), Huawei, TAL education. Ju Fan was supported by the NSF of China (U1711261), and the Research Funds of Renmin University of China (18XNLG18).

REFERENCES

[1] https://www.mturk.com/.
[2] https://echarts.baidu.com/.
[3] https://www.amcharts.com/.
[4] https://www.highcharts.com/.
[5] http://deepeye.tech/.
[6] R. A. Al-Zaidy and C. L. Giles. Automatic extraction of data from bar charts. In *K-CAP 2015*, pages 30:1–30:4, 2015.
[7] 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), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, pages 1261–1264, 2012.
[8] C. Chai, J. Fan, and G. Li. Incentive-based entity collection using crowdsourcing. In *ICDE 2018*, pages 341–352, 2018.
[9] C. Chai, G. Li, J. Fan, and Y. Luo. Crowdsourcing-based data extraction-from visualization charts. *ICDE*, 2020.
[10] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. Cost-effective crowdsourced entity resolution: A partial-order approach. In *SIGMOD 2016*, pages 969–984, 2016.
[11] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. A partial-order-based framework for cost-effective crowdsourced entity resolution. *VLDB J.*, 27(6):745–770, 2018.
[12] M. Cliche, D. S. Rosenberg, D. Madeka, and C. Yee. Scatteract: Automated extraction of data from scatter plots. In *ECML PKDD 2017*, pages 135–150, 2017.
[13] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.
[14] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD 2011*, pages 61–72, 2011.
[15] A. Gross, S. Schirm, and M. Scholz. Ycasd - a tool for capturing and scaling data from graphical representations. *BMC Bioinformatics*, 15:219, 2014.
[16] W. Huang, R. Liu, and C. L. Tan. Extraction of vectorized graphical information from scientific chart images. In *(ICDAR 2007*, pages 521–525, 2007.
[17] W. Huang, C. L. Tan, and W. K. Leow. Model-based chart image recognition. In *GREC 2003*, pages 87–99, 2003.
[18] D. Jung, W. Kim, H. Song, J. Hwang, B. Lee, B. H. Kim, and J. Seo. Chartsense: Interactive data extraction from chart images. In *CHI 2017.*, pages 6706–6717, 2017.
[19] G. Li, C. Chai, J. Fan, X. Weng, J. Li, Y. Zheng, Y. Li, X. Yu, X. Zhang, and H. Yuan. CDB: optimizing queries with crowd-based selections and joins. In *SIGMOD 2017*, pages 1463–1478, 2017.
[20] G. Li, C. Chai, J. Fan, X. Weng, J. Li, Y. Zheng, Y. Li, X. Yu, X. Zhang, and H. Yuan. CDB: A crowd-powered database system. *PVLDB*, 11(12):1926–1929, 2018.
[21] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. Crowdsourced data management: A survey. *IEEE Trans. Knowl. Data Eng.*, 28(9):2296–2319, 2016.
[22] K. Li and G. Li. Approximate query processing: What is new and where to go? *Data Science and Engineering*, 3(4):379–397, 2018.
[23] 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.
[24] 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 2014*, pages 1187–1198, 2014.
[25] Y. Liu, X. Lu, Y. Qin, Z. Tang, and J. Xu. Review of chart recognition in document images. In *Visualization and Data Analysis 2013, Burlingame, CA, USA, February 4-6, 2013*, page 865410, 2013.
[26] Y. Luo, X. Qin, N. Tang, and G. Li. Deepeye: Towards automatic data visualization. In *ICDE 2018*, pages 101–112, 2018.
[27] Y. Luo, X. Qin, N. Tang, G. Li, and X. Wang. Deepeye: Creating good data visualizations by keyword search. In *SIGMOD 2018*, pages 1733–1736, 2018.
[28] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.
[29] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR 2011*, pages 211–214, 2011.
[30] G. G. Méndez, M. A. Nacenta, and S. Vandenheste. ivolver: Interactive visual language for visualization extraction and reconstruction. In *CHI 2016*, pages 4073–4085, 2016.
[31] H. Park, R. Pang, A. G. Parameswaran, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: A system for declarative crowdsourcing. *PVLDB*, 5(12):1990–1993, 2012.
[32] X. Qin, Y. Luo, N. Tang, and G. Li. Deepeye: An automatic big data visualization framework. *Big data mining and analytics*, 1(1):75–82, 2018.
[33] A. Rohatgi. Webplotdigitizer, version 3.8. 2015.
[34] A. D. Sarma, A. G. Parameswaran, H. Garcia-Molina, and A. Y. Halevy. Crowd-powered find algorithms. In *ICDE*, pages 964–975, 2014.
[35] M. Savva, N. Kong, A. Chhajta, F. Li, M. Agrawala, and J. Heer. Revision: automated classification, analysis and redesign of chart images. In *UIST 2011*, pages 393–402, 2011.
[36] C. Shan, N. Mamoulis, G. Li, R. Cheng, Z. Huang, and Y. Zheng. T-crowd: Effective crowdsourcing for tabular data. In *ICDE 2018*, pages 1316–1319, 2018.
[37] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.
[38] M. Shao and R. P. Futrelle. Recognition and classification of figures in PDF documents. In *GREC 2005*, pages 231–242, 2005.
[39] B. Tummers. Datatheif iii. 2015.
[40] S. N. P. Vitaladevuni, B. Siddiquie, J. Golbeck, and L. S. Davis. Classifying computer generated charts. In *CBMI 2007*, pages 85–92, 2007.
[41] X. Yu, G. Li, Y. Zheng, Y. Huang, S. Zhang, and F. Chen. Crowdota: An online task assignment system in crowdsourcing. In *ICDE 2018*, pages 1629–1632, 2018.
[42] B. Zhao and J. Han. A probabilistic model for estimating real-valued truth from conflicting sources. *Proc. of QDB*, 2012.
[43] 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.
[44] Y. P. Zhou and C. L. Tan. Hough technique for bar charts detection and recognition in document images. In *ICIP 2000*, pages 605–608, 2000.

Chengliang Chai received his bachelor's degree in Computer Science and Technology from the Harbin Institute of Technology in 2015. He is currently a PhD student in the Department of Computer Science, Tsinghua University, Beijing, China. His research interests lie in crowdsourcing data management and data mining.

Guoliang Li is currently working as a professor in the Department of Computer Science, Tsinghua University, Beijing, China. He received his PhD degree in Computer Science from Tsinghua University, Beijing, China in 2009. His research interests mainly include data cleaning and integration, spatial databases and crowdsourcing.

Ju Fan is currently working as an associate professor in the School of Information, Renmin University, Beijing, China. He received his PhD degree in Computer Science from Tsinghua University, Beijing, China in 2009. His research interests mainly include data cleaning and integration, social network and crowdsourcing.

Yuyu Luo received his bachelor's degree in Software Engineering from the University of Electronic Science and Technology of China in 2018. He is currently a Master student in the Department of Computer Science, Tsinghua University, Beijing, China. His research interests include data cleaning and data visualization.