2018 IEEE 34th International Conference on Data Engineering

CrowdOTA: An Online Task Assignment System in Crowdsourcing

Xiang Yu [#], Guoliang Li [#], Yudian Zheng [*], Yan Huang [†], Songfan Zhang [†,‡], Fei Chen [†]

[#]*Department of Computer Science, National Laboratory for Information Science&Technology, Tsinghua University, China*
[*] *Twitter Inc, USA* [†]*TAL Education Group, China* [‡]*Sichuan University, China*
{x-yu17@mails,liguoliang}.tsinghua.edu.cn, yudianz@twitter.com, {galehuang,yangsongfan,chenfei_ai}@100tal.com

***Abstract*—Crowdsourcing is widely accepted as a means for resolving tasks that are hard for computers, e.g., entity resolution. Unfortunately, Crowdsourcing may yield relatively low-quality results if there is no proper quality control. Although previous studies attempt to eliminate workers by estimating workers' qualities via qualification tests or hidden tests, the qualities estimated may not be accurate, because workers may have diverse qualities across tasks. Thus, the quality of the results could be further improved by wisely assigning tasks to the workers who are specialized in the tasks and online task assignment is an effective way to achieve this goal. However, existing crowdsourcing platforms either do not support online task assignment (e.g., CrowdFlower) or are not user-friendly because they require to write complicated codes (e.g., Amazon MTurk). To address these limitations, we develop an online task assignment system, which can on-the-fly assign workers with appropriate tasks. We have deployed our system on top of MTurk. We demonstrate the following scenarios using our system. Firstly, requesters can easily utilize our system to enable online task assignment in order to improve answer quality. Moreover, requesters do not need to write any code. Secondly, our system can infer the quality of workers, and requesters can design and test their own task assignment algorithms using our proposed information. Thirdly, our system can monitor tasks and workers in real time, and the requesters can eliminate bad workers or terminate the crowdsourcing process to reduce the unnecessary cost.**

I. Introduction

Crowdsourcing provides an effective way to resolve computer-hard tasks, e.g., entity resolution and sentiment analysis (see a survey [11] and a book [13]). Due to its openness, crowdsourcing may yield relatively low-quality results, or even noise. Thus we need to design effective quality-control techniques to improve the quality. A widely-adopted strategy is to assign each task to multiple workers (called task assignment) and derive the result by aggregating worker answers (called truth inference). Task assignment aims to assign tasks to appropriate workers and truth inference attempts to infer the quality of workers and the truth of tasks. However, workers may have diverse qualities across tasks. Firstly, good workers usually give high-quality results while bad workers give low-quality results by randomly selecting answers [18], [2]. Secondly, many tasks are domain specific, and workers are usually good at tasks in domains they are familiar with but provide low-quality answers in unfamiliar domains [5], [16]. Although qualification tests (workers are required to perform a qualification test when they first come to answer real tasks, and the test contains a set of tasks with known ground truth) and hidden tests (a set of tasks with known ground truth is mixed into the real tasks and workers cannot distinguish between them), they have some limitations. Qualification tests cannot detect the workers who carefully answer qualification tasks but randomly answer refal tasks. Hidden tests involve additional cost and mistakenly eliminate high-quality workers.

Online task assignment can address these limitations by on-the-fly assigning tasks to appropriate workers [14], [6], [15]. However, existing crowdsourcing platforms either do not support online task assignment, e.g., CrowdFlower, or require users to write complex codes, e.g., Amazon MTurk. It is hard for requesters to use online task assignment. To address these limitations, we develop an online task assignment system, CrowdOTA. When a worker requests tasks, CrowdOTA on-the-fly selects k tasks to the worker. CrowdOTA implements multiple online task assignment algorithms and requesters can select any algorithm to assign their tasks. When a worker completes k tasks and submits their answers to CrowdOTA, our system computes the worker quality and infers the truth of each task. We have deployed our system on top of Amazon MTurk. Next we illustrate the key characteristics of our system as follows.

- **Improving Result Quality.** Compared with traditional offline task assignment that generates the tasks offline and randomly assigns tasks to workers, our system can improve the result quality by on-the-fly generating tasks and assigning them to appropriate workers [5], [18], [9].
- **Real-Time Monitoring.** Our system can monitor tasks and workers in real time. If a worker has bad quality, our system can on-the-fly block her from answering more tasks. If the quality of tasks are not acceptable, the requester can terminate and redesign their tasks.
- **Enabling Online Task Assignment.** Requesters do not need to write any code, and they only need to set the parameters, such as the price of each task, the number of assignments for each task in a configuration file. They can

[1]Guoliang Li is the corresponding author.

2375-026X/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDE.2018.00193

utilize our system to directly enable online task assignment. Our system provides multiple task assignment algorithms. Requesters can use our default task assignment algorithm, or select their preferred task assignment algorithms.

• **Testing New Task Assignment Algorithm.** Requesters can design and test their own assignment algorithms. Our system provides a database that keeps the required information in designing new assignment algorithms, including the inferred workers' quality, the status of current assignments, and the answers of the current assignments. Our source code has been open-sourced via the following link: https://github.com/TsinghuaDatabaseGroup/crowdota.

II. System Overview

A. System Architecture

Figure 1 shows the architecture of our system. Requesters can submit their tasks and deploy their applications by APPManager. APPManger interacts with the underlying crowdsourcing platform to publish tasks. When a worker requests tasks, HITCreator calls TaskAssignment to get k tasks, packs them into a HIT (Human Intelligent Task), uses a selected HTML template to render the HIT, and sends it to the crowdsourcing platform. TaskAssignment includes multiple task assignment algorithms which are used to select k tasks. When a worker completes k tasks, HITReceiver accepts the results and calls ResultInference to infer the worker quality and the truth of tasks. ResultInference includes multiple truth inference algorithms. All the tasks, workers, answers are stored in Database. Requesters can access Database to get the detailed information to design their own task assignment algorithms. Next, we introduce the details of each component.

• APPManager: Suppose a requester wants to deploy a crowdsourcing application with n tasks in CrowdOTA. The requester first needs to provide a folder which contains three files. (1) **Task File** is a csv file which stores the set of n tasks; (2) **UI Template File** is used to render k tasks as a user understandable HIT in HTML templates (the requester can also use our provided templates); (3) **Configuration File** contains all required information about the application, including the type of tasks (e.g., single-choice task), the url of the server to manage the HITs, the number of tasks in each HIT (k), the amount of money paid for each HIT (b), the total number of assignments of all HITs (m), the type of assignment and inference algorithms. We also provide some guidelines for the requester about how to set the required information about the application. For example, the number of tasks in each HIT (k) is related to the amount of money paid for each HIT (b). The total number of assignments of all HITs (m) is decided by the total budgets (denoted as B) provided by the requesters and $m = B/b$. We will give a detailed example to show how an application can be deployed and how the related parameters can be set in Section II-B. There are still works that study how to wisely set the budget [7].

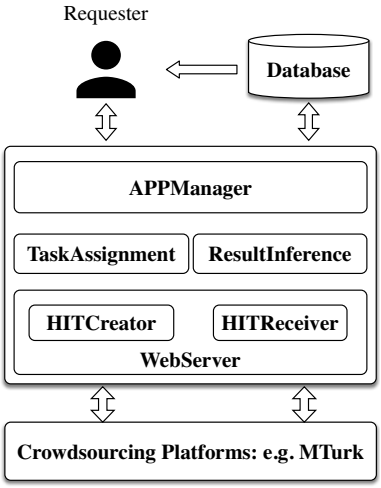

Figure 1. CrowdOTA Architecture.

CrowdOTA contains four types of built-in task types with corresponding UI templates, assignment and inference algorithms. Requesters only need to provide the tasks and configuration files to deploy their applications. Next, we briefly discuss the four types of tasks. (1) **Single-Choice Task**: workers are asked to choose one answer from multiple choices, e.g., select 'YES' or 'NO' in entity resolution. (2) **Multiple-Choice Task**: workers are asked to select one or more answers from multiple choices, e.g., select the objects from an image. (3) **Fill-in Task**: workers are asked to fill-in blank for given tasks, e.g., fill in the university of a professor. (4) **Collection Task**: workers are asked to answer open tasks, e.g., collect top-100 universities in US.

• Web Server: Web Server mainly processes the requests from the workers and consists of two parts: HITCreator and HITReceiver. (1) HITCreator: if a worker requests a HIT, HITCreator will call the TaskAssignment to on-the-fly generate k tasks. Then it returns an HTML page with these k tasks to the worker. (2) HITReceiver: if a worker submits a HIT, HITReceiver stores the HIT's information in the Database. HITReceiver calls ResultInference to update the worker quality and truth of the task in the Database. Once all HITs of the application are finished, the results of tasks will be stored in the Database.

• TaskAssignment: TaskAssignment is one of the two core components in CrowdOTA, which on-the-fly selects k tasks for a worker (see Section III-A for more details).

• ResultInference: Result Inference is another core component in CrowdOTA, which infers workers' quality and the truth of tasks. We will discuss the details in Section III-B.

• Database: Database is the component that stores tables including worker model, task model and HIT model. It contains the following tables.

(1) Table HITGroupInfo corresponds to HIT Group in

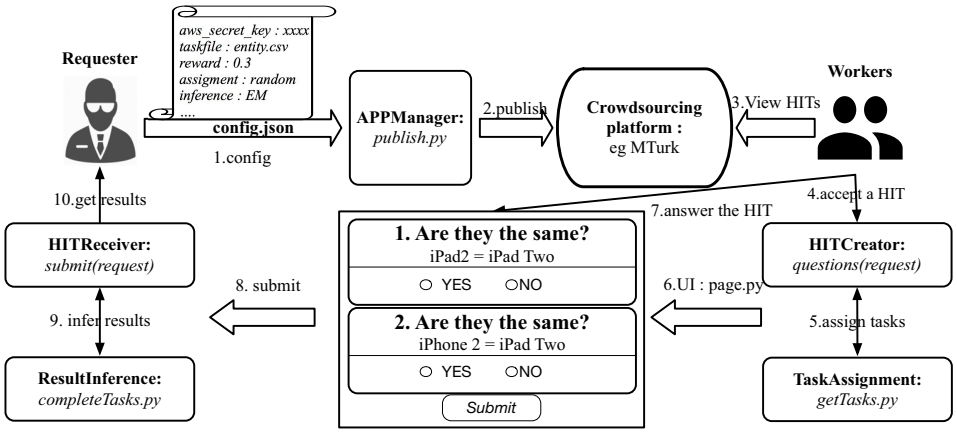

Figure 2. The Workflow of CrowdOTA.

MTurk where information of each HIT Group will be recorded. All HITs in one HIT Group share the same setting. The columns include HITGroupId (a unique Id), HITtitle (title of the HIT shown in MTurk), HITdescription (description of HIT shown in MTurk, explaining what the HIT is), questionsPerHIT (number of tasks in one HIT), taskType (type of tasks, e.g., "SC" short for Single-Choice), taskFile (the file that tasks can be loaded from), HITRemains (number of remaining HITs in the HITGroup).

(2) Table HITInfo contains two columns, HITId and HITGroupId. Based on HITId, one can get the corresponding HITGroupId ,then get the setting of the HIT.

(3) Table TaskInfo records each task's information. The columns include taskId, taskDescription (description of each task shown to workers in a HIT), taskLabels (labels for workers to choose in choice-like questions), HITGroupID (HIT Group the task belongs to), answered (the number that the task have been answered), result (result according to the current answers) and distribution (a json-format string records the probability distribution for each label).

(4) Table Assignment corresponds to assignment in MTurk. When a worker accepts a HIT, an assignment is generated. The columns include assignmentId (gotten from MTurk and return to MTurk if one accepts the worker's answer to the HIT), workerId (the worker who accepts the HIT), HITId (HIT which is accepted), arriveTime (time when the worker accepts the HIT), finishTime (time when the worker submits the answer of HIT) and accept (allow the worker to answer the HIT or return a blank page to reject the worker).

(5) Table Submission records all information about submissions. When a worker submits a HIT, the answers to the tasks in the HIT will be recorded. The columns include HITGroupId (HIT group the HIT belongs to), workerId (worker who answers the HIT), taskId (id of the task that the worker answered), HITId (id of the HIT) and result (result that the worker answers to the task).

(6) Table WorkerInfo records the information of worker, workerId and quality. The quality is a real number which can be gotten by truth inference algorithms.

B. System Workflow

CrowdOTA is a python project. There are some required softwares/programming tools to install, including Python, Django, Boto Library, Mysql and Stunnel. Django is used to construct a web server, which is used to interact with workers. When a worker requests a HIT on MTurk, a request will be sent to the HITCreator of our system. Thus we need to construct a web server that listens to the request. Since MTurk requires to use https server, we deploy Stunnel which can turn any insecure TCP port into a secure encrypted port using OpenSSL package for cryptography to support https. We use Boto Library to interact with MTurk, like publishing tasks or confirming workers' answers.

Figure 2 shows the workflow of our system. To use our system, requesters need to upload a csv file containing all the tasks and the configuration file *config.json*. It contains the settings such as the type of the application, the task file, the information of HITs and the specified Amazon MTurk key. Then APPManager will publish HITs to crowdsourcing platforms according to the *config.json*. When a worker requests a task, HITCreator will be asked to select k tasks. When a worker completes a HIT, HITReceiver will receive the answers and infer the worker quality and tasks' truth.

We provide two interfaces, called `getTasks(·)` and `completeTasks(·)`, which allow requesters to design their own task assignment and truth inference algorithms.

The interface `getTasks(workerId,hitId,taskNum)` is a function in "*getTasks.py*", which assigns the worker (with workerId) with a HIT (with hitId and the number of tasks, taskNum). If requesters want to test their own algorithms, they can rewrite the function to select tasks.

The interface `completeTasks(workerId,hitId)` is the function in "*completeTasks.py*", which denotes that worker (workerId) completes a HIT (hitId). The answers are recorded, and inference algorithm should be given and called to update the worker quality and truth for each task. Requesters can rewrite the function to test their own inference algorithms.

Example 1: We show how an application can be deployed. Suppose a requester wants to deploy an entity resolution application which can be treated as "Single-Choice

Task" in CrowdOTA and the requester generates $n = 1000$ tasks where each task has the choices "equal" and "non-equal". The requester first creates an application folder in the APPManager component. In the folder, the requester provides tasks file, configuration file with all settings. You can set the number of tasks as $k = 10$, money paid for a HIT as $b = \$0.2$, the total number of assignments as $m = 350$, the assignment algorithm as "random" and the inference algorithm as "EM". After creating the folder, requester uses APPManager to publish the tasks.

When a worker requests a HIT on MTurk, WebServer acquires the worker's id from MTurk and passes it to TaskAssignment, which selects $k = 10$ tasks, and returns a HIT consisting of the selected 10 tasks to the worker. When a worker completes a HIT, WebServer calls the ResultInference, which infers and updates the truth for each task and worker quality based on the task model and worker model. After obtaining the answers of all $m = 350$ HITs, CrowdOTA gets the final result for each task from Database. Note that if the requester wants to test their own assignment algorithm, it is only required to rewrite the `getTasks(·)` function in the file *getTasks.py*.

III. Algorithms

A. Task Assignment Algorithms

Task assignment algorithms are called when a worker requests a HIT. So the input is the worker (with workerId), the worker quality, and the current assignment status. The output is k selected tasks. CrowdOTA has implemented the following task assignment algorithms.

(1) Random. It randomly selects k tasks.

(2) CDAS [12]. It uses the quality-sensitive answering model to measure the confidence of tasks' current results. It then selects k tasks with the lowest confidences.

(3) AskIt! [1]. It uses the entropy to define the uncertainty of each task, and assigns k most uncertain tasks.

(4) QASCA [18]. When a worker comes, it estimates the improvement of quality if assigning each combination of k tasks to the worker, and select the optimal k tasks which result in the highest expected quality improvement.

(5) iCrowd [5]. It uses the graph-based estimation model to estimate the quality from observed answers, then assigns each task to the worker who has highest estimated quality of answering the task.

(6) CrowdDQS [8]. It uses marginal likelihood estimation to iteratively compute the expected value of a worker's accuracy. When a worker comes, it selects the tasks with maximum potential gain in the quality if the worker gives their current estimated true answers.

(7) CDB [10]. It estimates the quality improvement of each question and selects questions with large improvement.

B. Truth Inference Algorithms

Truth inference algorithms are called when a worker completes a HIT. The input is the current answers and the output is the worker quality and the inferred results. CrowdOTA has implemented the following truth-inference algorithms. (1) Majority Voting[17], [11]; (2) EM Algorithm [3], [4].

Acknowledgment

This work was supported by the 973 Program of China (2015CB358700), NSF of China (61632016, 61472198, 61521002, 61661166012), and TAL education.

References

[1] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W.-C. Tan. Asking the right questions in crowd data sourcing. In *ICDE*, pages 1261–1264, 2012.

[2] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. Cost-effective crowdsourced entity resolution: A partial-order approach. In *SIGMOD*, pages 969–984, 2016.

[3] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[4] 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)*, pages 1–38, 1977.

[5] J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng. icrowd: An adaptive crowdsourcing framework. In *SIGMOD*, pages 1015–1030, 2015.

[6] J. Feng, G. Li, H. Wang, and J. Feng. Incremental quality inference in crowdsourcing. In *DASFAA*, pages 453–467, 2014.

[7] Y. Gao and A. Parameswaran. Finish them!: Pricing algorithms for human computation. *PVLDB*, 7(14):1965–1976, 2014.

[8] A. R. Khan and H. Garcia-Molina. Crowddqs: Dynamic question selection in crowdsourcing systems. In *SIGMOD*, pages 1447–1462. ACM, 2017.

[9] G. Li. Human-in-the-loop data integration. In *VLDB*, pages 1015–1026, 2017.

[10] 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*, pages 1463–1478. ACM, 2017.

[11] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. Crowdsourced data management: A survey. *TKDE*, 28(9):2296–2319, 2016.

[12] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. Cdas: a crowdsourcing data analytics system. *PVLDB*, 5(10):1040–1051, 2012.

[13] A. Marcus and A. Parameswaran. Crowdsourced data management industry and academic perspectives. *Foundations and Trends in Databases*, 6(1-2):1–161, 2015.

[14] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD*, pages 229–240, 2013.

[15] X. Zhang, G. Li, and J. Feng. Crowdsourced top-k algorithms: An experimental evaluation. *PVLDB*, 9(8):612–623, 2016.

[16] Y. Zheng, G. Li, and R. Cheng. Docs: a domain-aware crowdsourcing system using knowledge bases. *PVLDB*, 10(4):361–372, 2016.

[17] 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.

[18] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng. Qasca: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*, pages 1031–1046, 2015.