2011 IEEE International Conferences on Internet of Things, and Cyber, Physical and Social Computing

Task Matching in Crowdsourcing

Man-Ching Yuen[1], Irwin King[1,2], and Kwong-Sak Leung[1]
[1]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong
[2]AT&T Labs Research, San Francisco, USA
{mcyuen, king, ksleung}@cse.cuhk.edu.hk; irwin@research.att.com

***Abstract*—Crowdsourcing is evolving as a distributed problem-solving and business production model in recent years. In crowdsourcing paradigm, tasks are distributed to networked people to complete such that a company's production cost can be greatly reduced. A crowdsourcing process involves operations of both requesters and workers. A requester submits a task request; a worker selects and completes a task; and the requester only pays the worker for the successful completion of the task. Obviously, it is not efficient that the amount of time spent on selecting a task is comparable with that spent on working on a task, but the monetary reward of a task is just a small amount. Literature mainly focused on exploring what type of tasks can be deployed to the crowd and analyzing the performance of crowdsourcing platforms. However, no existing work investigates on how to support workers to select tasks on crowdsourcing platforms easily and effectively. In this paper, we propose a novel idea on task matching in crowdsourcing to motivate workers to keep on working on crowdsourcing platforms in long run. The idea utilizes the past task preference and performance of a worker to produce a list of available tasks in the order of best matching with the worker during his task selection stage. It aims to increase the efficiency of task completion. We present some preliminary experimental results in case studies. Finally, we address the possible challenges and discuss the future directions.**

***Index Terms*—crowdsourcing; task model; task matching algorithm**

I. Introduction

Nowadays, many tasks that are trivial for humans continue to challenge even the most sophisticated computer programs, such as image annotation. Therefore, these tasks cannot be computerized and they are traditionally performed by an employee in a company. Crowdsourcing is a distributed problem-solving and business production model. In an article for Wired magazine in 2006, Jeff Howe defined "crowdsourcing" as "an idea of outsourcing a task that is traditionally performed by an employee to a large group of people in the form of an open call" [2]. The objective of crowdsourcing is to reduce a company's production costs and make more efficient use of labor and resources [3]. An example of crowdsourcing tasks is the creative drawings, such as the Sheep Market [4]. The Sheep Market is a web-based artwork to implicate thousands of workers in the creation of a massive database of drawings. Workers create their version of "a sheep facing to the left" using simple drawing tools. Each worker is responsible for a drawing receives a payment of two cents for his labor.

Because of the popularity of Web 2.0 technology, crowdsourcing websites attract much attentions at present [10], [9]. Some popular examples of crowdsourcing websites are Amazon Mechanical Turk (or MTurk)[1], CrowdFlower[2], Taskcn[3] and TopCoder[4]. A crowdsourcing site has two groups of users: requesters and workers. The crowdsourcing site exhibits a list of available tasks, associating with reward and time period, that are presented by requesters; and during the period, workers compete to provide the best submission. Meanwhile, a worker selects a task from the task list and completes the task because the worker wants to earn the associated reward. At the end of the period, a subset of submissions are selected, and the corresponding workers are granted the reward by the requesters. In addition to monetary reward, a worker gains credibility when his task is accepted by the requester.

Nowadays, recommendation systems [11], [5], [8] are used to suggest relevant items (news, books, movies, etc.) attracting particular users on the Web. In these systems, assigning each user-item pair a score indicating the user's rating on the item, based on which a ranking list of items is generated to the user as suggestions. In this paper, we address the issue of recommendation on crowdsourcing platforms. Different from traditional recommendation systems, multiple information and various relational dependencies in crowdsourcing systems should be utilized to improve recommendation results.

In this paper, we propose a novel task matching idea on crowdsourcing platforms. It can elicit the preference of task workers and collect their performance histories. In this way, a list of tasks sorted by the order of best matching can be provided to task workers when selecting tasks. It aims to increase the efficiency of task completion. Task matching in crowdsourcing is important because of the following reasons:

- **Motivate workers of diverse background to work on crowdsourcing tasks in long run.** Currently, on crowdsourcing sites, most workers only provide moderate contributions [7] and there is a significant population of young and well-educated Indian workers [6]. It can attract more workers to contribute their efforts in long run if a worker find a suitable task on a crowdsourcing site easily.
- **Improve the quality of work.** Workers perform better if they are familiar with the tasks. Chilton et al. showed that task workers only browsed the first few pages on crowdsourcing sites when searching for tasks [1]. The task list for a worker of Amazon Mechanical Turk site is

[1]Amazon Mechanical Turk website: https://www.mturk.com/
[2]CrowdFlower website: http://crowdflower.com
[3]Taskcn website: http://www.taskcn.com/
[4]TopCoder website: http://www.topcoder.com/

usually displayed on hundreds of pages. A worker selects a task from the list of available tasks sorted by a specified feature of tasks such as task creation date and reward amount. When the tasks posted on the first few pages are not suitable for a worker, the worker might choose a task that he does not familiar with and try to complete it to earn the rewards; otherwise, the worker does not select any task. Working with a unfamiliar task might decrease the quality of work.

The rest of this paper is organized as follows. Section II introduces the background and related works on task matching in crowdsourcing. Section III describes our algorithms of task matching in crowdsourcing. Section IV presents our preliminary experimental results based on case studies. Section V addresses possible challenges and concludes with future directions.

II. Background and Related Work

Crowdsourcing is an idea of outsourcing a task to a large group of networked people in the form of an open call to reduce the production cost. A worker has to select a task from more than ten thousands of tasks to work on in order to earn the tiny associated reward of such a few cents. Obviously, it is not efficient that the amount of time spent on selecting a task is comparable with that spent on working on a task. However, no existing work investigates on how to support task workers to select tasks on crowdsourcing platforms easily and effectively.

Task matching on crowdsourcing platforms are necessary. Amazon Mechanical Turk is used as an example. In February 2011, the number of available HITs (Human Intelligence Tasks) for qualified task workers and unqualified task workers on Amazon Mechanical Turk were about 80,000 and 2,000 on average per day. Qualified task workers passed qualification tests on Amazon Mechanical Turk, while unqualified task workers did not. It is extremely time-consuming for a task worker to select a right task even the list of available tasks can be sorted by a specified feature of HITs such as HIT creation date, HITs available, reward amount, expiration date, title or time allotted. Chilton et al. [1] showed that task workers look mostly at the first page of the most recently posted tasks and the first two pages of the tasks with the most available instances but in both categories the position on the result page is unimportant to workers. Some workers searching by almost all the possible categories and looking more than 10 pages deep. In 2011, Rachael King, a writer for Bloomberg Businessweek[5] in San Francisco, claimed that he earned a measly USD$ 4.38 for spending eight hours in a day crowdsourcing for Amazon Mechanical Turk. In his experience, he spent most of time to search for a right task which the wage was low.

Task matching can help workers to find their preferred tasks easier and faster, and it can encourage more workers to contribute and thus increase the population of workers. The task searching result is highly related to the rate of task completion and the quality of resulted work. In 2010, Chilton et al. [1] showed that a task with favorable positioning in the search results was completed 30 times faster and for less money than when its position was unfavorable. Task matching helps requesters to collect completed results of tasks in a shorter period of time without manipulating the position of their tasks, and the tasks can be presented to the workers who preferred to work on.

In this paper, our task matching algorithm can help a worker to get a list of tasks sorted by which categories of tasks are most preferred by the worker and which categories of tasks are most accepted for the worker based on the workers' task selection history and performance history.

III. Our Proposed Task Matching Algorithm

In this section, we first formulate our task matching algorithm in crowdsourcing and then describe our algorithm.

A. Definitions

Before proceeding further, we start with the definition of crowdsourcing task domain which includes the relationships among requesters, workers and tasks on a crowdsourcing platform. Next, we define the worker performance record of a worker to elict the worker's interest and performance. Then, we give a mathematical description of TaskRank which is used to sort the available tasks in the order of best matching with a worker.

Definition 1: A **crowdsourcing task domain** $\mathcal{CTD}$ of a crowdsourcing platform is a 4-tuple $(\mathcal{U}, \mathcal{V}, \mathcal{C}, \mathcal{T})$.

1) $\mathcal{U} = \{u_x | x = 1, ..., U_N\}$ is a set of requesters who distribute tasks on a crowdsourcing platform and U_N is the total number of requesters.
2) $\mathcal{V} = \{v_y | y = 1, ..., V_N\}$ is a set of workers who work on tasks on a crowdsourcing platform and V_N is the total number of workers.
3) $\mathcal{C} = \{c_i | i = 1, ..., C_N\}$ is a set of categories of tasks and C_N is the total number of categories of tasks.
4) $\mathcal{T} = \{T_i | i = 1, ..., C_N\}$ is a set of tasks in all categories, where T_i is a set of tasks in category c_i and is defined as $T_i = \{t_{i,j} | j = 1, ..., T_N\}$ such that T_N is the total number of tasks in category c_i.
   A task in category c_i, $t_{i,j}$, has the following attributes:
   a) $r_{i,j}$ is the requester of task $t_{i,j}$ where $r_{i,j} = \exists u_x \in \mathcal{U}$.
   b) $W_{i,j} = \{w_{i,j,k} | k = 1, ..., W_N\}$ is a set of workers who work on task $t_{i,j}$ and W_N is the number of workers who work on task $t_{i,j}$, where $w_{i,j,k} = \exists v_y \in \mathcal{V}$.
   c) $e_{i,j}$ is the time allotted by requester $r_{i,j}$ for worker $w_{i,j,k}$ to complete task $t_{i,j}$.
   d) $A(r_{i,j})$ is a set of the workdone of task $t_{i,j}$ accepted by requester $r_{i,j}$.
   e) $m_{i,j}$ is the monetary reward offered by requester $r_{i,j}$ for worker $w_{i,j,k}$ to complete task $t_{i,j}$.
   f) $d(w_{i,j,k})$ is the workdone of task $t_{i,j}$ completed by worker $w_{i,j,k}$.

[5]Bloomberg Businessweek website: http://www.businessweek.com/technology/

Worker $w_{i,j,k}$ earns $m_{i,j}$ if $d(w_{i,j,k}) \in A(r_{i,j})$; otherwise, worker $w_{i,j,k}$ earns 0 if $d(w_{i,j,k}) \notin A(r_{i,j})$.

Definition 2: A **worker performance record** $\mathcal{WPR}$ of a worker v_y on a crowdsourcing platform is a 4-tuple $(\mathcal{AR}, \mathcal{CPS}, \mathcal{RPS}, \mathcal{TPS})$ which is defined as:

1) $T_i(v_y) = \{t_{i,j} | 1 \leqslant j \leqslant T_N; v_y = \exists w_{i,j,k} \in W_{i,j}\}$ is a set of tasks in category c_i for worker v_y, such that $T_i(v_y) \subset T_i \subset T$.
2) $T'_i(v_y) = \{t_{i,j} | 1 \leqslant j \leqslant T_N; v_y = \exists w_{i,j,k} \in W_{i,j}$ and $d(w_{i,j,k}) \in A(r_{i,j})\}$ is a set of tasks in category c_i for worker v_y and the workdone of task $t_{i,j}$ by worker v_y is accepted by requester $r_{i,j}$, such that $T'_i(v_y) \subset T_i(v_y)$. $T_i(v_y) - T'_i(v_y)$ is a set of tasks in category c_i completed by worker v_y but not accepted by requester $r_{i,j}$.
3) **Acceptance rate** $\mathcal{AR}_i(v_y)$ of tasks in category c_i for worker v_y is defined as:
$$\mathcal{AR}_i(v_y) = \frac{|T'_i(v_y)|}{|T_i(v_y)|} \tag{1}$$
4) **Task category preference score** $\mathcal{CPS}_i(v_y)$ on category c_i of worker v_y is defined as:
$$\mathcal{CPS}_i(v_y) = \frac{|T_i(v_y)|}{\sum_{p=1}^{C_N} |T_p(v_y)|} \tag{2}$$
5) **Reward preference score** $\mathcal{RPS}_i(v_y)$ on category c_i for worker v_y is defined as:
$$\mathcal{RPS}_i(v_y) = \frac{\sum_{j=1}^{T_N} m_{i,j}}{|T_i(v_y)|} \tag{3}$$
6) **Time alloted preference score** $\mathcal{TPS}_i(v_y)$ on category c_i for worker v_y is defined as:
$$\mathcal{TPS}_i(v_y) = \frac{\sum_{j=1}^{T_N} e_{i,j}}{|T_i(v_y)|} \tag{4}$$

Definition 3: **TaskRank** $\mathcal{TR}$ of a task $t_{i,j}$ in category c_i for a worker v_y is given as follows:

$$\begin{aligned}\mathcal{TR}_{i,j}(v_y) =& \mathcal{AR}_i(v_y) * \mathcal{CPS}_i(v_y) * \\ & \left(1 - \left|\frac{m_{i,j} - \mathcal{RPS}_i(v_y)}{\mathcal{RPS}_i(v_y)}\right|\right) * \\ & \left(1 - \left|\frac{e_{i,j} - \mathcal{TPS}_i(v_y)}{\mathcal{TPS}_i(v_y)}\right|\right)\end{aligned} \tag{5}$$

B. Overview

To increase the efficiency of task completion, the matching algorithm utilizes the past task preference and performance of a worker to produce a list of available tasks in the order of best matching with the worker during his task selection stage. Table I shows our proposed task matching algorithm.

TABLE I
THE TASK MATCHING ALGORITHM IN CROWDSOURCING

```
**Procedure** ComputeTaskRank(worker $v_y$, performance record $\mathcal{WPR}(v_y)$)
{Execute when worker $v_y$ log into the crowdsourcing system}
**Input**:
$T$, the set of available tasks in all categories
$C_N$, the number of task categories
$T_N$, the number of available tasks in category $c_i$
**Algorithm**:
**for** $i = 1 \rightarrow C_N$ **do**
    **for** $j = 1 \rightarrow T_N$ **do**
        Based on $T$ and $\mathcal{WPR}(v_y)$
        Compute **TaskRank** $\mathcal{TR}$ of each available task for $v_y$
        $j \leftarrow j + 1$
    **end for**
    $i \leftarrow i + 1$
**end for**
**Output**:
The list of available tasks of all categories sorted by **TaskRank** in descending order for $v_y$

**Procedure** UpdateRecord(worker $v_y$, completed task $t_{i,j}(v_y)$)
{Execute when worker $v_y$ complete a task $t_{i,j}(v_y)$}
**Input**:
$T'$, the set of expired tasks in all categories
$C_N$, the number of task categories
$T_N$, the number of expired tasks in category $c_i$
**Algorithm**:
**for** $i = 1 \rightarrow C_N$ **do**
    Based on $T'$ and $t_{i,j}(v_y)$
    Update **acceptance rate** $\mathcal{AR}_i(v_y)$ for $v_y$
    Update **task category preference score** $\mathcal{CPS}_i(v_y)$ for $v_y$
    Update **reward preference score** $\mathcal{RPS}_i(v_y)$ for $v_y$
    Update **time alloted preference score** $\mathcal{TPS}_i(v_y)$ for $v_y$
    Update **performance record** $\mathcal{WPR}(v_y)$ of $v_y$ for each category
    $i \leftarrow i + 1$
**end for**
**Output**:
Updated **performance record** $\mathcal{WPR}(v_y)$ of $v_y$ for all categories
```

The process of generating the task list of best matching proceeds in the following steps:

1) At the beginning, a set of task categories has to be predefined on a crowdsourcing platform, so that any task offered by a requester can be categorized into one of the task categories. Examples of task categories are image annotation and translation. When a requester offers a task on a crowdsourcing platform, the requester has to identify which task category that the task belongs to.
2) The crowdsourcing platform maintains a worker performance record for each worker. The worker performance record of a worker stores his task selection preference and his task acceptance rate. The task selection preference of a worker is computed based on the information of the tasks selected by the worker in the past. The information includes task category, reward and time alloted, and they are the major task selection criteria for a worker. The task acceptance rate of a worker reflects the performance of the worker on each task category.
3) When a worker log into the crowdsourcing platform, the platform lists the available tasks in the order of best matching with the worker based on his TaskRank of tasks. TaskRank provides a method to rate the available

tasks for workers to select. A higher TaskRank value means a higher likelihood that a worker will choose to work on. For a worker, the TaskRank of a task measures the worker's interest on the task and his ability to complete the task successfully. In other words, different workers might have different TaskRank on the same task. As mentioned before, most workers only browsed the first few pages when searching tasks; while some requesters try to manipulate the position of their tasks in the search results to workers [1]. The available tasks list in the order of best matching with a worker can help the worker to pay attention on the most suitable and interested tasks easily and quickly.

IV. Preliminary Results in Case Studies

In August 2011, we ran a case study in Hong Kong to examine the related effectiveness of our proposed task matching algorithm. The experiment involves 12 participants of different backgrounds. 7 of the participants are males and 5 of the participants are females. The age range of the participants is from 18 to 50. Some participants are students, while the others have various occupations. There are 4 cateorgies of tasks, and they are grammar correction, vocabulary usage, calculation and knowledge acquisition in science. Each cateorgy has 10 tasks. The time alloted for tasks varies from 10 to 30 minutes. The monetary reward for each task is the same. Each participant is required to complete 10 out of 40 tasks and rate 4 additional tasks. We use the Mean Absolute Error (MAE) metrics to measure the prediction quality of our proposed approach with the random approach. MAE is defined as

$$MAE = \frac{\sum |r_{w,t} - \hat{r}_{w,t}|}{N} \qquad (6)$$

where $r_{w,t}$ denotes the rating that worker w gave to task t, and $\hat{r}_{w,t}$ denotes the rating that worker w gave to task t which is predicted by our approach, and N denotes the number of tested ratings. In MAE metrics, a lower value indicates higher accuracy.

Table II shows the MAE comparison between our proposed algorithm and random approach. We observed that our proposed algorithm has higher accuracy than random approach in predicting users' preferences. Moreover, feedback from workers show that they prefer to work on tasks which they performed before in long run.

TABLE II
MAE comparison with random approach

	MAE
TaskRank	0.50
Random	1.08

V. Conclusion and Future Work

In this paper, we have proposed a novel idea on task matching in crowdsourcing. It utilizes the past task preference and performance of a worker to produce the available task list in the order of best matching with the worker during task selection. It can motivate workers to contribute their efforts on crowdsourcing tasks in long run, improve the quality of work and increase the efficiency of task completion. Besides, we have described and formulated our algorithm in detail. We have presented some preliminary experimental results in case studies.

In the future, we plan to conduct some experiments on Amazon Mechanical Turk for performance evaluation. Some potential future extensions include evaluating computational complexity of our algorithm in real time and examining how to categorize tasks on a crowdsourcing platform. Too many task categories increases the computational complexity; while too few task categories decreases the accuracy of our algorithm.

Acknowledgment

This work was partially supported by a grant from Microsoft (Project No. CUHK 6902498) and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 413210). It is affiliated with the Microsoft-CUHK Joint Laboratory for Human-centric Computing and Interface Technologies.

References

[1] L. B. Chilton, J. J. Horton, R. C. Miller, and S. Azenkot. Task search in a human computation market. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 1–9, New York, NY, USA, 2010. ACM.

[2] J. Howe. The rise of crowdsourcing. *Wired*, 14(6), June 2006.

[3] J. Howe. *Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business*. Crown Business, 2008.

[4] A. M. Koblin. The sheep market. In *Proceeding of the seventh ACM conference on Creativity and cognition*, C&C '09, pages 451–452, New York, NY, USA, 2009. ACM.

[5] H. Ma, I. King, and M. R. Lyu. Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 39–46, New York, NY, USA, 2007. ACM.

[6] J. Ross, L. Irani, M. S. Silberman, A. Zaldivar, and B. Tomlinson. Who are the crowdworkers?: shifting demographics in mechanical turk. In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, CHI EA '10, pages 2863–2872, New York, NY, USA, 2010. ACM.

[7] O. Stewart, D. Lubensky, and J. M. Huerta. Crowdsourcing participation inequality: a scout model for the enterprise domain. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 30–33, New York, NY, USA, 2010. ACM.

[8] X. Xin, I. King, H. Deng, and M. R. Lyu. A social recommendation framework based on multi-scale continuous conditional random fields. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1247–1256, New York, NY, USA, 2009. ACM.

[9] M.-C. Yuen, L.-J. Chen, and I. King. A survey of human computation systems. In *CSE '09: Proceedings of IEEE International Conference on Computational Science and Engineering*, pages 723–728. IEEE Computer Society, 2009.

[10] M.-C. Yuen, I. King, and K.-S. Leung. A survey of crowdsourcing systems. In *SocialCom '11: Proceedings of The Third IEEE International Conference on Social Computing*. IEEE Computer Society, 2011. To be appeared.

[11] T. C. Zhou, H. Ma, I. King, and M. R. Lyu. Tagrec: Leveraging tagging wisdom for recommendation. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 04*, pages 194–199, Washington, DC, USA, 2009. IEEE Computer Society.