Worker Recommendation for Crowdsourced Q&A Services: A Triple-Factor Aware Approach

Zheng Liu
Department of Computer Science and Engineering
HKUST
zliual@cse.ust.hk

Lei Chen
Department of Computer Science and Engineering
HKUST
leichen@cse.ust.hk

ABSTRACT

Worker Recommendation (WR) is one of the most important functions for crowdsourced Q&A services. Specifically, given a set of tasks to be solved, WR recommends each task with a certain group of workers, whom are expected to give timely answers with high qualities. To address the WR problem, recent studies have introduced a number of recommendation approaches, which take advantage of workers' expertises or preferences towards different types of tasks. However, without a thorough consideration of workers' characters, such approaches will lead to either inadequate task fulfillment or inferior answer quality.

In this work, we propose the Triple-factor Aware Worker Recommendation framework, which collectively considers workers' expertises, preferences and activenesses to maximize the overall production of high quality answers. We construct the Latent Hierarchical Factorization Model, which is able to infer the tasks' underlying categories and workers' latent characters from the historical data; and we propose a novel parameter inference method, which only requires the processing of positive instances, giving rise to significantly higher time efficiency and better inference quality. What's more, the sampling-based recommendation algorithm is developed, such that the near optimal worker recommendation can be generated for a presented batch of tasks with considerably reduced time consumption. Comprehensive experiments have been carried out using both real and synthetic datasets, whose results verify the effectiveness and efficiency of our proposed methods.

PVLDB Reference Format:
Zheng Liu, Lei Chen. Worker Recommendation for Crowdsourced Q&A Services: A Triple-Factor Aware Approach. *PVLDB*, 11 (3): 380 - 392, 2017.
DOI: 10.14778/3157794.3157805

1. INTRODUCTION

Recently a large number of crowdsourced Q&A applications have come into being, such as Stack Exchange, Yahoo Answer, and Quora. As a result, it becomes increasingly convenient to make use of the crowd's intelligence. With the quick growth of crowd's participation, one of the central issues in managing crowdsourced Q&A platforms is worker recommendation, which is to find appropriate workers for the published tasks and actively ask for their solutions. Taking Quora as an example, when people are going to publish their tasks, the platform provides some candidate workers who are expected to give satisfactory feedbacks, and allows people to request answers from these workers.

The worker recommendation problem has drawn wide attention and several primary works have been proposed, such as [27, 33, 16, 35, 29]. Fundamentally, most of the existing methods treat the worker recommendation as the "expert finding" problem, where two basic operations are involved. Firstly, tasks are categorized according to some specific criteria. Secondly, workers are ranked with their expertise (or authority) to each category of tasks based on their historical answers. In this way, given a new task, its corresponding category can be recognized and workers with the top expertise on this category will be recommended for it.

In spite of the existing progress, the worker recommendation problem cannot be effectively addressed only with workers' expertise. To facilitate timely acquisition of high quality answers, the worker recommendation should jointly consider the following three factors: **expertise**, **preference** and **activeness**. The detailed reasons to incorporate these three factors are given as follows. First of all, the crowds have diverse skills in different domains. To make sure the acquired answers are reliable, the recommended workers must be equipped with enough expertise on the related areas of the given tasks. Secondly, the crowd's preferences towards different types of tasks are discrepant. For example, some people may prefer mathematical problems to those on classical literature; if tasks on both categories are available, they will probably choose the mathematical ones with higher priority. To guarantee workers' acceptance, the recommendation should consider the ones who have satisfactory preferences towards the type of presented tasks. Lastly, the crowd's activenesses are significantly different from each other: the enthusiastic workers would probably solve a large number of tasks within a short period of time, while the passive ones might only make very little production. For the sake of timely feedbacks, the recommended workers are desirable to be active in generating new answers. (For better demonstration of the above three factors, real world data exploration will be demonstrated in Section 3.)

Proceedings of the VLDB Endowment, Vol. 11, No. 3
Copyright 2017 VLDB Endowment 2150-8097/17/11... $ 10.00.
DOI: 10.14778/3157794.3157805

In this work, we come up with the **Tri**ple-factor Aware Worker **Rec**ommendation framework (referred as TriRec), where all the aforementioned factors are jointly considered for the worker recommendation. In TriRec, we construct the **L**atent **H**ierarchical **F**actorization **M**odel (referred as LHFM), which categorizes the tasks based on their contents and characterizes the workers with their expertises, preferences and activenesses. With LHFM, we can directly identify the category of a newly presented task, and figure out the workers who will not only complete the given task with large probabilities, but also be capable of making high quality answers. In addition to LHFM, we develop the optimized worker recommendation strategy for the scenarios where new tasks are processed in batches. With such a strategy, the overall recommendation effect can be maximized for a whole batch of presented tasks.

However, the realization of TriRec is non-trivial, where the following technical challenges have to be conquered. **(1)** The answering activities (i.e., which tasks have been answered by each of the workers) are jointly affected by the workers' preferences and activenesses. If a task is not answered by a worker, it could be resulted from either the worker's non-interest to the given task, or the limitation of her activeness. As such, both factors, preference and activeness, are not directly reflected by the observation data. Meanwhile, the majority of workers have only answered a small fraction of tasks in history (e.g., a common worker in Stack Overflow may produce around a few hundreds answers; while, there are hundreds of millions tasks ever issued). Therefore, explicit reasoning of the non-answered tasks will be not only possibly erroneous, but also severely time consuming. **(2)** The optimal worker recommendation, formed as a weighted bipartite assignment problem, cannot be solved by the conventional approaches in a temporally scalable manner. (e.g., Hungarian algorithms or Successive Shortest Paths, whose time cost could be as expensive as $O(|W|^2|T|^2)$). Whereas, in real applications, large-scale cases might be frequently confronted. (For example, there are over 6 million registered workers in Stack Overflow, which will make it almost impractical to optimally process merely a batch of few hundreds tasks.)

To address the above challenges, we come up with the following technical mechanisms. **(1)** LHFM probabilistically models the answering activities as unified distributions of workers' preferences and activenesses. With the maximization of corresponding posteriori, workers' preferences and activenesses can be judiciously learned from the historical observation. Meanwhile, LHFM's probabilistic formulations avoid explicit reasoning of the negative instances, and the parameter inference can be conducted with purely positive instances. (An instance associates with a pair of task and worker, which is positive if the task is answered by the worker; otherwise, it is negative.) In this way, potential errors can be avoided and the inference process will be greatly accelerated. **(2)** The optimal worker recommendation is produced with our sampling-based algorithm. In our proposed approach, the workers' suitableness towards each type of tasks are indexed offline; based on which worker recommendation can be efficiently produced for the newly presented tasks through sampling the pre-constructed table. In particular, near optimal result (whose expectation is above 1/4 of the optimal solution) can be obtained with O(|T|) time complexity.

The major contributions of this work are summarized as the following points.

• We propose TriRec to maximize the production of high quality answers in crowdsourced Q&A services, where workers' expertises, preferences, and activenesses are collectively considered. To the best of our knowledge, this is the first work incorporating all these factors.

• We come up with the Latent Hierarchical Factorization Model (LHFM), which encodes tasks with their semantic signatures, and characterizes workers with their expertises, preferences and activenesses. With LHFM, workers' suitableness towards a given task can be directly acquired.

• Our proposed model avoids explicit reasoning of the negative instances, which eliminates potential errors and enables the parameters to be inferred with high efficiency.

• The optimal worker recommendation is produced with the sampling-based algorithm, which not only considerably reduces the time consumption, but also generates the approximation result whose expectation is lower-bounded by 1/4 of the optimal solution.

• We conduct extensive experiments with both real and synthetic datasets, whose results verify our improvement over the state-of-the-art approaches.

The rest of our paper is organized as follows. We review the related work in Section 2. In Section 3, we explore the data from crowdsourced Q&A services and demonstrate the workers' characters of expertises, preferences and activenesses. In Section 4, we introduce the key definitions of this work. The LHFM and Optimal Worker Recommendation are discussed in Section 5 and 6, respectively; and the experimental study is reported in Section 7. Finally, we draw the conclusion and discuss the future work in Section 8.

2. RELATED WORK

The related works are reviewed from the following two aspects: Worker Recommendation in Crowdsourced Q&A services and Crowdsourced Task Allocation.

Worker Recommendation. Worker recommendation is an important function for Q&A services. A large proportion of the existing works fall into the domain of Expert Finding (EF for short). Given people's historical answering records, EF estimates the workers underlying capabilities of solving each type of tasks correctly. Representative works include (but do not limited to) [27, 35, 1, 29]. In particular, [27] proposes a model to determine the tasks' latent topics and to estimate the workers' expertises over different topics through link analysis; [35] takes the correlation among tasks' categories into consideration, and workers' expertises are collaboratively estimated with answer records in all the relevant categories; [1] proposes a semi-supervised coupled mutual reinforcement framework for simultaneous calculation of workers' expertise and answers' quality; and [29] formulates the expert finding problem as a graph-regularized matrix factorization process, with workers' social relationship taken into account. Some of these techniques are also employed in our estimation of worker expertise, such as latent representations of workers and tasks, and factorization formulation. What's more, the techniques for exploration of auxiliary data (e.g., social relationship) can also be incorporated into our framework to further improve the inference quality. Despite some works on workers' preferences estimation, e.g., [26, 15, 30], which employ incremental collaborative filtering, latent semantic indexing and content-based matrix factorization as the inference strategies, they are technically similar to the existing studies on expert finding. Because of leaving out the activeness factor, they are unable to infer the workers' preferences with high qualities, which has been theoretically analyzed and experimentally tested in the previous sections. Most of the existing works simply focus on one aspect of workers' characters, rather than comprehensively considering their expertises, preferences and activenesses, they are inferior to maximize the overall production of high quality answers.

Crowdsourced Task Allocation. Task allocation is an intensively studied issue in crowdsourcing research. For example, [10, 4] proposes the offline task assignment strategy in the spatial scenario; [5] goes beyond the basic settings (with only spatial and temporal constraints) by considering the situation where workers are equipped with diversified skills; besides, [21] develop optimal online strategies to deal with the cases where workers and tasks are presented in a streaming fashion. In spite of the fundamentally different application scenarios, these works also involve the optimal solution for weighted bipartite assignment problems. However, all the proposed methods in these works incur running time of $O(|W|^{\epsilon_1}|T|^{\epsilon_2})$ ($\epsilon_1, \epsilon_2 \geq 1$), which is significantly above the $O(|T|)$ time consumption in our work, and not scalable enough to handle the real crowdsourced Q&A services where a huge number of workers are presented.

In addition, a number of task allocation works have been proposed for matching tasks to workers with "right skills", such as [32, 31, 11]. Particularly, [32] estimates the workers reliabilities on every specific task, and selects the top-k tasks for each sequentially arrived worker to achieve the largest answer quality improvement; [31] utilizes the taxonomy of skills to match the tasks with workers of necessary skills; instead of using a predefined taxonomy, [11] leverages the knowledge base to analyze tasks skill domains, based on which tasks are assigned to workers with appropriate skills so as to gain the maximum ambiguity reduction. Our work is substantially different from the above ones from the following aspects. Firstly, because of the application to the crowdsourced Q&A scenario, our optimization goal is the overall acquisition of timely and qualified answers (measured in terms of EFP), rather than simply the answers quality. Secondly, workers expertise has to be captured in a totally different way because of the application scenario's difference. Thirdly, the recommendation for crowdsourced Q&A services requires effective estimation and utilization of workers' activeness and preferences; however, neither factors are crucial for task assignment scenarios (e.g., task allocation in AMT), thus neglected by the aforementioned methods.

3. DATA EXPLORATION

In this section, we explore the data collected from Stack Overflow[1] to analyze three types of workers' characters: preferences, expertises and activenesses. The explored dataset is composed of historical tasks and the scores of workers' answers towards them. The content of each task is represented by a bag of keywords (e.g., {*c++*, *polymorphism*, *virtual_function*, *dynamic_linkage*}), and each worker's score on a specific task is normalized by the corresponding task's maximum score (so that it is confined within [0, 1]).

[1]https://stackoverflow.com

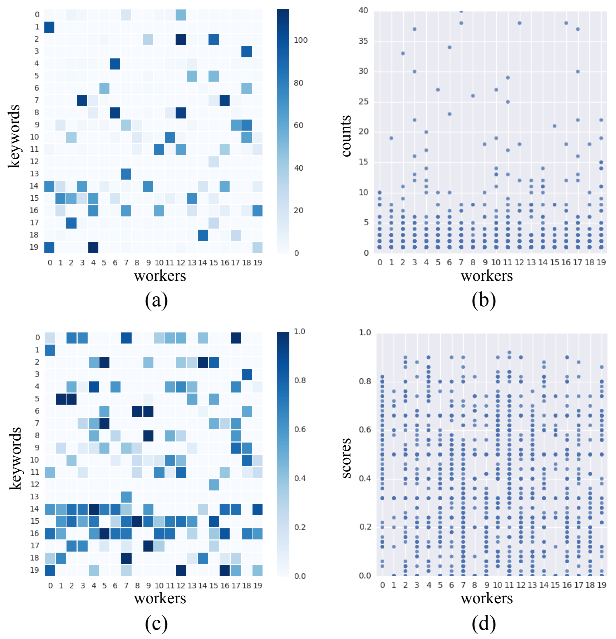


Figure 1: (a), (b): Heat Map and Dots of count@keyword; (c), (d): Heat Map and Dots of score@keyword.

We aggregate each worker's (e.g., worker w) associated records into two values:

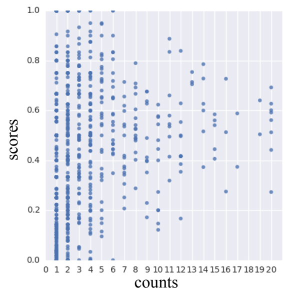
- *count@keyword*, which equals to the count of w's answered tasks containing a certain keyword; and
- *score@keyword*, which equals to the average score of w's answered tasks with the corresponding keyword.

We demonstrate the aggregated results from four aspects: 1) workers' counts on different keywords (reflection of preference), 2) workers' average scores on different keywords (reflection of expertise), 3) the relationship between workers' average scores and counts on the common keywords (correlation of preference and expertise), and 4) the distribution of workers' annual counts of answers (reflection of activeness).

Counts on keywords. We randomly select 20 workers who produced over 100 answers within the year of 2016 and demonstrate their preferences in terms of *count@keyword* in Figure 1 (a) and (b). (20 popular keywords are randomly chosen for the demo of heat map.) In the demonstration results, only a few blocks in the heat map are deeply colored, and most of the points' counts in the dots map are below 10, which reflects that each worker would only make frequent answers for tasks associated with a small set of keywords. Besides, the workers' differences in the heat map indicate their greatly diversified preferences over different keywords.

Average scores on keywords. We further demonstrate the workers' expertises with score@keyword in Figure 1 (c) and (d). Similar to the observation in (a) and (b), each worker is only good at tasks associated with a small set of keywords, whose answers towards such tasks are rated with comparatively high scores; in addition, workers' expertises are significantly diversified across the keywords.

Counts and average scores on common keywords. In Figure 2 (a), each dot represents the tuple of a worker's (count@keyword, score@keyword) to a common keyword. The averages and standard variances of score@keyword for different count@keyword are aggregated in Figure 2 (b). According to the presented results, the variation of answers' counts does not clearly affect the value distributions on


(a)

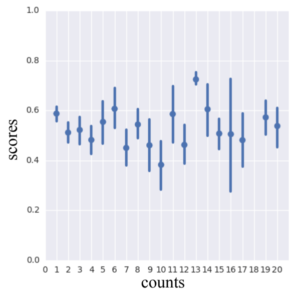

(b)

Figure 2: Average score and counts on common Keywords: (a) Dots of (count@keyword, score@keyword) on common keywords; (b) average scores and standard variances of different counts.

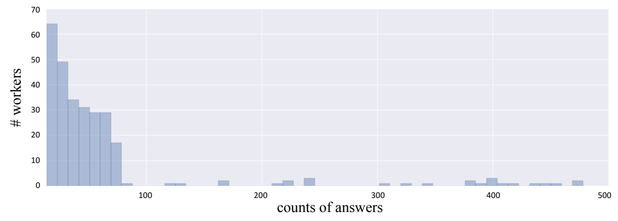


Figure 3: Distribution of annual counts of answers.

score@keyword, which indicates workers' expertises and preferences are not necessarily correlated. As a result, both factors need to be taken into account so as to find the workers who will not only probably accept a given task, but also be capable of making qualified answers.

Annual Counts of Answers. We randomly sample 300 workers (from the Top 60,000 ones with the greatest numbers of answers), whose annual counts of answers in 2016 are aggregated in Figure 3. It is observed that workers' activenesses are greatly diversified: most workers made less than 100 answers, while some others may even produce over 400 answers.

Summary. The exploration results reveal that workers have greatly diversified preferences, expertises and activenesses, which significantly affect their choices of tasks, quality of answers, and frequencies of making new answers. To guarantee the effectiveness of recommendation, a worker is desirable to be equipped with satisfactory preference and expertise for the given tasks, and active in generating new answers. Thus, all these factors need to be precisely captured and jointly considered within the recommendation strategy.

4. DEFINITION AND INFRASTRUCTURE

Task. Each task is represented by its content, in the form of Bag-of-Words. Tasks' contents reflect their underlying categories; e.g., a task with words {polymorphism, virtual function, dynamic linkage} indicates its affiliation with objective oriented programming. Each task will be recommended with c_t different workers, whom will be invited to give their answers. (c_t is referred as the task's capacity.)

Worker. Each worker w is associated with the historical data of her answered tasks T_w and scores of her answers $\{s_t^w\}_{t \in T_w}$. For the unification of scale, each score s_t^w is normalized by the maximum score $\hat{s}_t$ of the corresponding task (i.e., $s_t^w/\hat{s}_t$), so that it can be confined within the range of $[0, 1]$. From the historical data, the workers' expertises, preferences and activenesses are inferred, which will provide guidance for the future worker recommendation. Notice that we only take the workers who have some answering

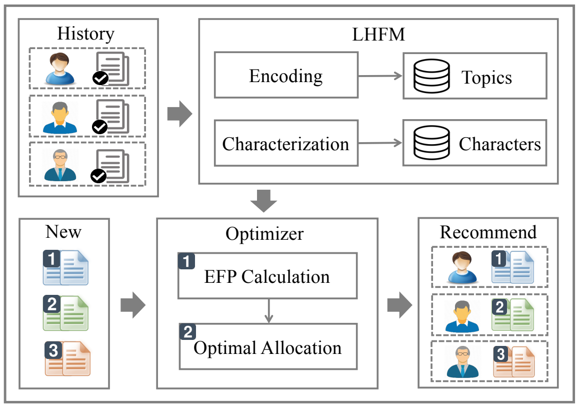


Figure 4: System Infrastructure.

records in history into consideration, i.e., everyone has answered some tasks in the history. To deal with the newly registered workers, we may resort to the active exploration methods (e.g., [9]) to initiatively acquire users' data; such an operation can be seamlessly joined with our framework. In this paper, each worker can be only recommended to a finite number of tasks, and the maximum is referred as the worker's capacity (denoted as c_w).

Latent Hierarchical Factorization Model. The Latent Hierarchical Factorization Model (LHFM) represents tasks in the semantic space and characterizes workers with their latent factors. In particular, each task is mapped into the latent semantic space based on its content, which provides a natural indicator of its underlying category; and each worker is characterized by two latent vectors, representing her expertise and preference to each type of task, and one latent scalar, reflecting her activeness level. The tasks' and workers' latent features are tightly-coupled in LHFM, whose collaborative explanation towards the observation data (i.e., answers' appearances and scores) facilitates more accurate probabilistic reasoning (as discussed in [24, 23]). What's more, the explicit incorporation of workers' activenesses and preferences not only provides each individual answers appearance likelihood, but also judiciously explains a workers count of answers within a certain period.

Worker Recommendation. As discussed in the Introduction, for a newly presented task, the $\underline{\text{active}}$ workers with high $\underline{\text{expertises}}$ and $\underline{\text{preferences}}$ are desirable for recommendation. To quantitatively measure a worker's suitableness in terms of all these aspects, we come up with **Expected Final Payoff** (EFP), whose definition is stated as follows.

Definition1. *(EFP) Suppose task t is recommended with worker w, who will* 1) *accept and answer task t in a valid time period with probability p_t^w; and* 2) *the produced answer will be of quality s_t^w (i.e., normalized score). The expected final payoff (denoted as* $\text{EFP}(t, w)$*) of making such a recommendation is $p_t^w * s_t^w$.*

Apparently, timely answer can be acquired with a large probability iff. the worker is active and has a strong preference towards the presented task; and the answer's quality will be good iff. the worker is equipped with high expertise. As such, a large value of expected final payoff indicates the worker's satisfaction of all the desired properties, whereby reflecting her suitableness of being recommended.

When tasks are processed in stream and the FCFS principle (first-come, first served) is adopted, the naive top-K strategy can be directly applied for worker recommendation,

Table 1: General Notations

Notation	Description
T	the whole set of tasks for current stage
W	the whole set of workers for the current stage.
W_t	workers in W who have answered task t.
T_w	tasks in T which have been answered by worker w.
c_t	tasks' capacity.
c_w	worker's capacity.
β_i	the i-th topic.
θ_t	task t's latent semantic vector.
z_t	task t's underlying topic
e_w	worker w's latent expertise vector.
ρ_w	worker w's latent preference vector.
a_w	worker w's activeness scaler.
Ψ_w	worker w's count of answers.
I_t^w	worker w's acceptance of task t.
s_t^w	worker w's score to task t.
$\mathrm{EFP}(t, w)$	the EFP of recommending task t with worker w.

where the most appropriate workers will be recommended to the currently submitted task. While, we still have to find out the worker recommendation strategy when tasks are presented and processed in a batch manner. In particular, it is necessary to figure out how to maximize the overall EFP for a whole batch of presented tasks. Such a problem is referred as Optimal Worker Recommendation (OWR). Given tasks' and workers' capacity constraints, together with EFP between each pair of task and worker, OWR is formed as the following maximum weighted bipartite assignment problem.

Definition2. *(OWR) Given the presented batch of tasks, the available workers, the capacity constraints and EFP of each task-worker pair, the optimal worker recommendation assigns each task with a set of workers, which produces the maximum total EFP and preserves the capacity feasibility.*

System Infrastructure. We demonstrate the system infrastructure in Figure 4. In particular, the workers' historical answering records are fully utilized by LHFM, where latent topics are generated to encode the tasks' semantics, and latent characters are estimated to represent the workers in terms of expertise, preference and activeness. Given a set of new tasks, the system calculates workers' suitableness (i.e., EFP) towards each presented task; by solving the consequent OWR problem, each task gets its recommended set of workers, which leads to the maximum overall EFP.

We summarize the frequently used notations in Table 1 to facilitate comprehension.

5. LATENT HIERARCHICAL FACTORIZATION MODEL

LHFM consists of two functional components: 1) the Semantic Encoding module, which encodes a task to its latent semantic vector based on its content; and 2) the Worker Characterization module, which characterizes a worker with her latent vectors of expertise and preference, and the latent scalar of her activeness. With the model's parameters inferred from historical data, the expected final payoff can be calculated for future worker recommendation.

5.1 Probabilistic Formulation

Semantic Encoding. Following the conventional topic modeling approaches, we adopt K topics (denoted as $\{\beta_i\}_1^K$) to represent the underlying categories behind the tasks' contents[2]. Each of the topics is a specific distribution over the vocabulary, indicating every word's appearance probability.

The content of each task is composed of a small set of duplicable words. Because of such a "short-text" nature, we assume that the words of a common task are generated from one identical topic (e.g., task t is associated with topic β_{z_t}), which is a common practice to deal with short texts [28]. The index of each task's associated topic (i.e., z_t) is a hidden variable, whose distribution is specified by its latent semantic vector θ_t (i.e., the ith component $\theta_t[i]$ stands for the probability of $z_t = i$).

Given the topics $\{\beta_i\}_1^K$ and latent semantic vector θ_t, the generation probability for task t's content is presented as:

$$P(\{o\}_t | \{\beta_{z_t}\}_1^K, \theta_t) = \prod_{\{o\}_t} \sum_{z_t=1}^{K} P(o|\beta_{z_t}) P(\beta_{z_t}|\theta_t), \quad (1)$$

where o is a word in task t's content. It's obvious that θ_t indicates task t's underlying categorical information; therefore, it can be regarded as t's latent semantic signature.

Worker Characterization. Each worker w is characterized with her latent factors of activeness a_w, preference ρ_w and expertise e_w; these factors, together with the corresponding tasks' latent semantics, jointly explain the answers' appearances and scores with the following probabilistic formulations.

• Answers' Appearances. The workers' answering records are partitioned into different time slots (e.g., every month) and sequentially processed. In each time slot, two pieces of information are available: 1) the number of answers produced by each worker (count of answers); 2) the types of tasks answered by each worker (task selection). Both information are probabilistically reasoned as follows.

We model worker w's answer appearances with two sequential steps: Browsing and Selection. Firstly, the worker browses a total of b_w tasks subject to her activeness. Following the conventional modeling of counting data [3], we adopt the Poisson process to capture b_w's value, parameterized of w's activeness a_w, i.e. Poisson(a_w):

$$P(b_w|a_w) = (a_w)^{b_w} e^{-a_w} / b_w! \, . \quad (2)$$

With such a formulation, a_w can be interpreted as w's expected browsing frequency (e.g., the monthly number of w's browsed tasks on average). Secondly, from the browsed tasks, w makes selection according to her preference (represented by ρ_w) towards the type of a presented task (indicated by θ_t). In particular, ρ_w is a K-dimensional weight vector, reflecting w's odds of choosing each type of tasks; therefore, t's probability of being selected is captured by the Bernoulli distribution Ber($\rho_w \theta_t$), i.e.,

$$P(\mathrm{I}_t^w | \rho_w, \theta_t) = \begin{cases} \rho_w \theta_t, & \mathrm{I}_t^w = 1, \\ 1 - \rho_w \theta_t, & \mathrm{I}_t^w = 0, \end{cases} \quad (3)$$

where the binary variable I_t^w equals to 1 if the browsed task t is selected by w; or 0, otherwise. For the above equation, a larger product of $\rho_w \theta_t$ indicates that w will select t with a higher probability. Finally, being the joint result of Browsing and Selection, the count of answers follows the factorized Poisson distribution presented in Theorem 1.

[2]Selecting the optimal topic number is an open issue, despite HDP [20] or Perplexity [2]. However, for our problem where parameter inference is conducted in a supervised way, the optimal K can be selected as the minimum value leading to the best prediction performance [12, 19].

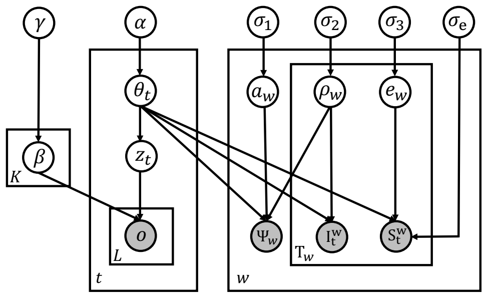


Figure 5: Graphical Model of LHFM.

Theorem1. *Given the worker's latent activeness a_w, latent preference ρ_w, and all the tasks' latent semantics $\{\theta_t\}_T$, the count of answers Ψ_w (i.e., $\Psi_w = |T_w|$) follows the Poisson distribution* Poisson($\sum_T a_w\rho_w\theta_t/|T|$)*:*

$$P(\Psi_w|\mathbf{\Theta}) = \frac{1}{\Psi_w!}(\frac{\sum_T a_w\rho_w\theta_t}{|T|})^{\Psi_w} e^{-\sum_T a_w\rho_w\theta_t/|T|}, \quad (4)$$

where $\mathbf{\Theta}$ is the whole set of parameters a_w, ρ_w, and $\{\theta_t\}_T$.

Proof. Presented in Section 9. □

• Answers' Scores. The answers' scores are jointly resulted from the workers' expertises and the tasks' categories. Specifically, worker w's expertise e_w is a K-dimensional vector, which indicates the comparative quality of w's answer towards each type of task. As a result, w's answer quality towards a presented task t is reflected by the product of e_w and θ_t, i.e., $e_w\theta_t$. Being an indicator of the answer's quality, the generation of score s_t^w is modeled with the factorized Gaussian distribution N($e_w\theta_t$, σ_e^{-1}), i.e.,

$$P(s_t^w|e_w,\theta_t) = \frac{1}{\sqrt{2\pi\sigma_e^{-1}}} e^{\frac{-\sigma_e(s_t^w - e_w\theta_t)^2}{2}}, \quad (5)$$

where σ_e serves as the regularization parameter for the fitting loss. Apparently, a large product of $e_w\theta_t$ will probably result in a answer's high score.

Generative Process. The graphical model of LHFM is illustrated in Figure 5, where the left part demonstrates the semantic encoding, and the right part shows the worker characterization. The hollow circles represent the hidden variables, and the solid ones stand for the observations. The parameters in the top row: γ, α, σ_1, σ_2, σ_3 and σ_e are the model's priors. Given our proposed probabilistic formulations, the generative process for observation data is presented as follows.

• Semantic Encoding.
  Draw each topic in $\{\beta_j\}_K$: $\beta_j \sim$ Dir(γ).
  Draw each task's latent semantic vector: $\theta_t \sim$ Dir(α).
  Draw each task's associated topic w.r.t. its latent semantic vector: $z_t \sim$ Mult(θ_t).
  Draw each word o in task t's content $\{o\}_t$ w.r.t. its associated topic: $o \sim$ Mult(β_{z_t}).
• Worker Characterization.
  For each worker w.
    Draw activeness scalar, $a_w \sim$ N(0, σ_1^{-1}).
    Draw preference vector, $\rho_w \sim$ N(0, $\sigma_2^{-1}I$).
    Draw expertise vector, $e_w \sim$ N(0, $\sigma_3^{-1}I$).
  For each time slot.
    Draw count of answers, $\Psi_w \sim$ Poisson($\frac{\sum_T a_w\rho_w\theta_t}{|T|}$).
    For each task t in T_w.
      Draw task selection, $I_t^w \sim$ Ber($\rho_w\theta_t$).
      Draw answer score, $s_t^w \sim$ N($e_w\theta_t$, σ_e^{-1}).

Remarks. For the above generative process, the worker characterization only involves the positive instances (tasks that have been answered) associated with each worker; while explicit reasoning for negative instances (tasks that have not been answered) is eliminated. Thus, unnecessary time consumption and possible errors caused by explaining the negative instances can be avoided in the parameter inference.

5.2 Parameter Inference

The log-likelihood of the generative process is derived as:

$$\begin{aligned}\mathcal{L} =& \log P(\mathbf{O},\mathbf{I},\mathbf{\Psi},\boldsymbol{S},\boldsymbol{Z},\boldsymbol{\theta},\boldsymbol{\beta},\boldsymbol{\rho},\boldsymbol{a},\boldsymbol{e}|\alpha,\gamma,\sigma_1,\sigma_2,\sigma_3,\sigma_e)\\ =& \log P(\mathbf{O}|\boldsymbol{Z},\boldsymbol{\beta})P(\boldsymbol{Z}|\boldsymbol{\theta})P(\boldsymbol{\theta}|\alpha)P(\boldsymbol{\beta}|\gamma) + \log P(\mathbf{\Psi}|\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{\rho}) +\\ & \log P(\mathbf{I}|\boldsymbol{\theta},\boldsymbol{\rho}) + \log P(\boldsymbol{S},|\boldsymbol{\theta},\boldsymbol{e},\sigma_e) +\\ & \log P(\boldsymbol{Z},\boldsymbol{\rho},\boldsymbol{a},\boldsymbol{e}|\sigma_1,\sigma_2,\sigma_3)\\ =& \mathcal{L}_{\mathrm{O}} + \mathcal{L}_{\Psi} + \mathcal{L}_I + \mathcal{L}_S + \mathcal{L}_\pi.\end{aligned} \quad (6)$$

The bold symbols represent the sets of homogeneous parameters, e.g., $\mathbf{O}$: $\{\{o\}_t\}_T$. $\mathcal{L}_{\mathrm{O}}$, $\mathcal{L}_\Psi$, $\mathcal{L}_I$, $\mathcal{L}_S$ stands for the latent factors' log-likelihoods w.r.t. 1) tasks' contents, 2) counts of answers, 3) tasks' selection and 4) answers' scores, respectively; and $\mathcal{L}_\pi$ is priors' log-likelihood for the workers' characterized factors.

Given the observations and model's priors, the values of latent parameters ($\boldsymbol{Z}$, $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, $\boldsymbol{a}$, $\boldsymbol{\rho}$, $\boldsymbol{e}$) are learned to maximize the likelihood function in Eq. 6. A Monte Carlo EM style algorithm [25] is developed for parameter inference. In particular, the semantic-related parameters (i.e., $\boldsymbol{Z}$, $\boldsymbol{\beta}$, $\boldsymbol{\theta}$) are inferred through Gibbs sampling (referred as Semantic Sampling); while, the worker characterization's factors (i.e., $\boldsymbol{\theta}$, $\boldsymbol{\rho}$, $\boldsymbol{a}$, $\boldsymbol{e}$) are estimated with gradient descent (named as positive-instance Gradient because of the involvement of purely positive instances). The semantic sampling and positive-instance gradient are iteratively carried out until the convergence of the likelihood function.

Semantic Sampling. In this step, we sample the associated topic of each task through Gibbs sampling. Given the topics and latent semantics, the full conditional probability of assigning task t with the k-th topic follows:

$$P(z_t = k|\theta_t,\boldsymbol{\beta}) = \frac{\prod_{o\in\{o\}_t}\theta_{t,k} * \beta_{k,o}}{\sum_{l=1}^{K}\prod_{o\in\{o\}_t}\theta_{t,l} * \beta_{l,o}}, \quad (7)$$

where $\theta_{t,k}$ and $\beta_{k,o}$ are the k-th and o-th components of the corresponding vectors. With the topic sampled for task t (i.e., z_t), the topics and latent semantic vector are updated according to the empirical distributions:

$$\begin{aligned}\beta_{j,o_i} &= \frac{\sum_T \mathrm{I}_{z_t=j} * |\{o|o=o_i\}_t| + \gamma}{\sum_T \mathrm{I}_{z_t=j} * |\{o\}_t| + \gamma|\mathbb{O}|},\\ \theta_{t,k} &= \frac{\sum_{\{o\}_t}\beta_{k,o} + \alpha}{\sum_{l=1}^{K}\sum_{\{o\}_t}\beta_{l,o} + \alpha K},\end{aligned} \quad (8)$$

where $\mathbb{O}$ denotes the vocabulary, and the binary indicator $\mathrm{I}_{z_t=j}$ equals to 1 if $z_t = j$; or 0, otherwise.

Following the conventional topic modeling processing, the above topic sampling and update operations will be repetitively carried out for a predefined number of iterations [6]. Then stabilized estimation for all the relevant variables ($\boldsymbol{Z}$, $\boldsymbol{\beta}$, $\boldsymbol{\theta}$) can be obtained.

Positive-instance Gradient. With the maximization of $\mathcal{L}_{\mathrm{O}}$ in semantic sampling, we iteratively adapt the model's parameters to maximize the summation of $\mathcal{L}_\Psi$, $\mathcal{L}_I$, $\mathcal{L}_S$ and $\mathcal{L}_\pi$ (denoted as $\mathbb{L}$). We use mini-batch gradient descent to

Algorithm 1: Parameter Inference

```
input  : $\mathbf{O}$, $\mathbf{\Psi}$, $\mathbf{I}$, $\boldsymbol{S}$, $K$, $\alpha$, $\gamma$, $\sigma_1$, $\sigma_2$, $\sigma_3$, $\sigma_e$.
output: $\boldsymbol{\beta}$, $\boldsymbol{Z}$, $\boldsymbol{\theta}$, $\boldsymbol{\rho}$, $\boldsymbol{a}$, $\boldsymbol{e}$.
1  begin
2      repeat
3          Semantic Sampling:
4          • Update $\boldsymbol{\beta}$, $\boldsymbol{Z}$, $\boldsymbol{\theta}$ acoording to Eq. 7, 8;
5          Positive-instance Gradient:
6          • for each worker $w$ do
7              for positive instances $T_w$ in each time slot do
8                  Update $a_w$, $\rho_w$, $e_w$ according to Eq. 11;
9          • for each task do
10             Update $\theta_t$ according to Eq. 12;
11     until a predefined number of iterations;
12 return $\boldsymbol{\beta}$, $\boldsymbol{Z}$, $\boldsymbol{\theta}$, $\boldsymbol{\rho}$, $\boldsymbol{a}$, $\boldsymbol{e}$;
```

update the parameters' values. In particular, $\mathbb{L}$'s partial gradients w.r.t. worker w's relevant factors are derived as:

$$\begin{aligned}\frac{\partial \mathbb{L}}{\partial a_w} &= \frac{\partial \mathcal{L}_\Psi}{\partial a_w} + \frac{\partial \mathcal{L}_\pi}{\partial a_w}\\ \frac{\partial \mathbb{L}}{\partial \rho_w} &= \frac{\partial \mathcal{L}_\Psi}{\partial \rho_w} + \frac{\partial \mathcal{L}_I}{\partial \rho_w} + \frac{\partial \mathcal{L}_\pi}{\partial \rho_w}\\ \frac{\partial \mathbb{L}}{\partial e_w} &= \frac{\partial \mathcal{L}_S}{\partial e_w} + \frac{\partial \mathcal{L}_\pi}{\partial e_w};\end{aligned} \tag{9}$$

while $\mathbb{L}$'s partial gradient w.r.t. task t's latent semantic vector is derived as follows:

$$\frac{\partial \mathbb{L}}{\partial \theta_t} = \frac{\partial \mathcal{L}_\Psi}{\partial \theta_t} + \frac{\partial \mathcal{L}_I}{\partial \theta_t} + \frac{\partial \mathcal{L}_S}{\partial \theta_t}, \tag{10}$$

where W_t represents the workers who have answered task t.

The above gradients' computation only relies on the information about positive instances, i.e., the number of answers Ψ_w, the answered tasks T_w, and the answers' scores $\{s_t^w\}_{T_w}$. As a result, it merely requires $\mathrm{O}(\sum_W |T_w|)$ (i.e., the scale of positive instances) times of derivate calculations for all the workers and tasks (as presented in Eq. 9, 10).

The parameters are updated according to the derived gradients. We adopt an iterative manner to coordinate the updating process, where $\boldsymbol{a}$, $\boldsymbol{\rho}$, and $\boldsymbol{e}$ are firstly adapted with fixed $\boldsymbol{\theta}$. For each of the workers, the following operations are carried out for her characterized factors based on Eq. 9:

$$\begin{aligned}a_w &\leftarrow a_w + \eta_1(\frac{\mathcal{L}_\Psi}{\partial a_w} + \frac{\mathcal{L}_\pi}{\partial a_w}),\\ \rho_w &\leftarrow \rho_w + \eta_1(\frac{\mathcal{L}_\Psi}{\partial \rho_w} + \frac{\mathcal{L}_I}{\partial \rho_w} + \frac{\mathcal{L}_\pi}{\partial \rho_w}),\\ e_w &\leftarrow e_w + \eta_1(\frac{\partial \mathcal{L}_S}{\partial e_w} + \frac{\partial \mathcal{L}_\pi}{\partial e_w}).\end{aligned} \tag{11}$$

The update for each worker is recursively carried out until the convergence to a local maxima. Based on the updated $\boldsymbol{a}$, $\boldsymbol{\rho}$ and $\boldsymbol{e}$, each task's latent semantic vector is further refined:

$$\theta_t \leftarrow \theta_t + \eta_2(\frac{\partial \mathcal{L}_\Psi}{\partial \theta_t} + \frac{\partial \mathcal{L}_I}{\partial \theta_t} + \frac{\partial \mathcal{L}_S}{\partial \theta_t}). \tag{12}$$

Similarly, the adaptation is repetitively performed until the convergence to another local maxima.

Monte Carlo EM. The parameter inference is summarized in Alg. 1. As discussed above, the algorithm is performed iteratively: the tasks' semantic vectors are inferred with Gibbs sampling (line 3-4), based on which the workers' characters are updated through gradient descent (line 6-8); the latent semantic vectors are further refined according to the updated workers' characters (line 9-10). While updating worker w's characters, the batch of her positive instances T_w is sequentially presented according to the time slots. The above operations will be repetitively conducted for a predefined number of iterations to generate the final estimation for all the model's parameters.

In Alg. 1, workers' preferences are judiciously adjusted together with their activenesses, without explicit reasoning of the negative instances. As a result, not only the time cost is greatly reduced, but more reliable inference can be produced as well. A toy example is presented as follows to briefly illustrate the mechanism.

Example1. *Given the batch of tasks within a specific time slot, e.g.,* $T = \{t_1, ..., t_{10}\}$*. Suppose only* t_1 *is answered by worker* w_1*; and according to* w_1*'s inferred activeness, she usually generates approximately one answer within each time slot. In this situation,* w_1*'s non-answering of* $\{t_2, ..., t_{10}\}$ *is probably resulted from the limitation of* w_1*'s activeness, rather than her preference. Therefore,* w_1*'s preference* ρ_{w_1} *will be increased for the relevant domains of* t_1*, without much decrement for the related domains of* $\{t_2, ..., t_{10}\}$*.*

Meanwhile, there is another worker w_2*, who also answers* t_1 *in the same time slot. However,* w_2*'s inferred preference over* $\{t_2, ..., t_{10}\}$ *is small, and* w_2 *would usually produce roughly three answers in each time slot. In this situation,* w_2*'s non-answering of* $\{t_2, ..., t_{10}\}$ *is likely to be caused by* w_2*'s limited preference. As a result,* w_2*'s preference* ρ_{w_2} *will be decreased for the relevant domains of* $\{t_2, ..., t_{10}\}$*, without much decrement for her activeness.*

The time cost of each iteration in Alg. 1 is divided into two parts. Firstly, the time complexity of semantic sampling is $\mathrm{O}(K\sum_T |\{o\}_t|)$, which is comparable to general topic analyzing approaches[3], such as [6]. Secondly, the positive-instance gradient only requires a time cost of $\mathrm{O}(K\sum_W |T_w|)$, thanks to the purely involvement of positive instances. The conventional approaches (e.g., [23, 13]) would process both positive and negative instances, which lead to the time consumption of $\mathrm{O}(K|W||T|)$. Given that $|T_w| \ll |T|$, the inference for workers' characters is significantly accelerated.

5.3 Expected Final Payoff Estimation

With LHFM's parameters inferred in Alg. 1, EFP can be calculated for a given pair of task and worker. First of all, as a direct deduction from Theorem 1, the probability of task t's being answered by worker w within an arbitrary time period is stated by the following theorem.

Theorem2. *For an arbitrary time period* τ*, task* t*'s being answered by worker* w *follows the Bernoulli distribution* $\mathrm{Ber}(1 - \exp(-a_w\rho_w\theta_t|\tau|/|T||\tau_0|))$*, i.e.,*

$$P(\mathrm{A}_t^w|a_w, \rho_w, \{\theta_t\}_T) = \begin{cases} 1 - e^{-\frac{a_w\rho_w\theta_t|\tau|}{|T||\tau_0|}}, & \mathrm{A}_t^w = 1;\\ e^{-\frac{a_w\rho_w\theta_t|\tau|}{|T||\tau_0|}}, & \mathrm{A}_t^w = 0;\end{cases} \tag{13}$$

where $|\tau_0|$ *is the length of one unit time slot;* A_t^w *equals to 1 if* w *answers* t*; or 0, otherwise.*

Proof. Presented in Section 9. □

[3]Semantic sampling could be safely omitted if there have been topics constructed from comprehensive corpus [23]

Given task t's content, the latent semantic vector θ_t can be firstly determined with Eq. 8. Suppose t is needed to be solved within time period τ, the probability of t's being answered by w (denoted as $P\{\mathrm{A}_t^w(\tau)=1\}$) is presented as:

$$P\{\mathrm{A}_t^w(\tau)=1\} = 1 - e^{-\frac{a_w \rho_w \theta_t |\tau|}{|T||\tau_0|}}, \tag{14}$$

according to the statement of Theorem 2. With the expected answer quality calculated with Eq. 5, i.e., $E\{s_t^w\} = e_w\theta_t$, we can derive the expected final payoff between t and w with the following equation:

$$\mathrm{EFP}(t,w) = P\{\mathrm{A}_t^w(\tau)=1\} * E\{s_t^w\}. \tag{15}$$

6. OPTIMAL WORKER RECOMMENDATION

Formed as a maximum weighted bipartite assignment problem, the optimal worker recommendation can be solved with traditional methods, such as Hungarian or Successive Shortest Path [14]. Whereas, the time complexity for these methods could be as high as $O(|T|^2|W|^2)$, which is temporally impractical for large scale problems. Even if we resort to the simple greedy algorithm for accelerated computation and put up with its approximation result, it will still require the running time of $O(|T|^2|W|)$.

In this paper, with workers indexed by their characterized factors, we design a sampling-based algorithm for the recommendation. Briefly, the workers are indexed by a $|K| \times |W|$ matrix, namely Utility Table, which records the workers' recommendation utilities toward each type of tasks. When a batch of tasks are presented, we will consecutively sample their underlying categories (based on their latent semantic vectors) and candidate workers (based on the Utility Table). From the sampled candidates, we recursively generate the recommendation result through sampling the candidates' utilities, until all the tasks are recommended with the required numbers of workers. In this way, we can acquire a solution with its approximation ratio greater than 1/4 using merely $O(|T|)$ running time.

6.1 Worker Indexing

We create the $|K| \times |W|$ utility table (denoted as $\mathbf{U}$) to index the workers with their recommendation utilities towards each type of tasks. In particular, we define the set of standard tasks $\{\bar{t}_1, ..., \bar{t}_K\}$, where $\bar{t}_k$'s semantic vector is one-hot at the k-th dimension (i.e., $\theta_{\bar{t}_k}[i] = 1$ iff. $i = k$; otherwise, $\theta_{\bar{t}_k}[i] = 0$). The ($k$, w)-th entry of the utility table records the EFP of task $\bar{t}_k$ and worker w, i.e.,

$$\mathbf{U}(k,w) = \mathrm{EFP}(\bar{t}_k, w). \tag{16}$$

6.2 Sampling-based Recommendation

Given a batch of tasks $\mathrm{T_{rec}}$, the worker recommendation is generated with the following steps.

Type Sampling. Given the content of each task $t \in \mathrm{T_{rec}}$, we extract its latent semantic vector θ_t based on LHFM's semantic encoding module (whose computation follows Eq. 8). With the extraction of latent semantic vectors, we determine each task's underlying type z_t through sampling the multinomial distribution specified by θ_t, i.e., $z_t \sim \mathrm{Mult}(\theta_t)$.

Candidates Generation. Based on the sampled types of all the tasks, $\{z_t\}_{\mathrm{T_{rec}}}$, we allocate each task with a candidate worker through sampling the utility table. Particularly,

Algorithm 2: Sampling-based Recommendation

```
   input : $\mathrm{T_{rec}}$, $\mathbf{U}$
   output: R
1  begin
2      initialize $\mathrm{R} \leftarrow \{\mathrm{R}_t \leftarrow \emptyset\}_{\mathrm{T_{rec}}}$;
3      for each task $t \in \mathrm{T_{rec}}$ do
4          extract latent semantic vector $\theta_t$ with Eq. 8;
5          sample type $z_t$ with $\mathrm{Mult}(\theta_t)$;
6          sample candidate $\kappa_t$ with $\mathrm{Mult}(\phi_{z_t})$;
7      $T'_{rec} \leftarrow \mathrm{T_{rec}}$;
8      repeat
9          sample task $t$ for recommendation with $\mathrm{Mult}(\varphi)$;
10         add $t$'s candidate: $\mathrm{R_t} \leftarrow \mathrm{R_t} + \kappa_t$;
11         remove $t$ from $T'_{rec}$ if count_$t = c_t$;
12         suspend $\kappa_t$ if count_$\kappa_t = c_{\kappa_t}$;
13         re-sample candidates and update $\boldsymbol{\mathcal{K}}$;
14     until $T'_{rec} = \emptyset$;
15 return R;
```

each worker will be chosen as task t's candidate, denoted as κ_t, with the multinomial distribution specified by vector ϕ_{z_t}:

$$\phi_{z_t,w} = \frac{\mathbf{U}(z_t,w)}{\sum_{w' \in W} \mathbf{U}(z_t,w')}. \tag{17}$$

Recommendation Sampling. The worker recommendation is produced by sequentially selecting the candidates with a looped sampling process. Specifically, given all candidates (denoted as $\boldsymbol{\mathcal{K}}$), task t will be sampled to acquire its candidate κ_t according to multinomial distribution φ, where

$$\varphi_t = \frac{\mathbf{U}(z_t,\kappa_t)}{\sum_{t' \in \mathrm{T_{rec}}} \mathbf{U}(z_{t'},\kappa_{t'})}. \tag{18}$$

Once task t is sampled, the candidate κ_t will be added to its recommendation list; and another worker (other than those who have been recommended to t) will be selected as the new candidate for task t.

Due to the capacity constraints, task t will be removed from $\mathrm{T_{rec}}$ once it is recommended with the required number (i.e., c_t) of workers; at the same time, worker w will be taken out from the current candidates and exempted from being a candidate any more, if she has been recommended to the maximum number (i.e., c_w) of tasks. Such a sampling process will be repetitively carried out, until all the tasks are recommended with the required number of workers.

The worker recommendation process is summarized in Alg. 2. The tasks' recommendation lists (R) are initialized to be empty sets (line 2); and in the first for-loop, the tasks' types and candidates are sequentially sampled (line 5, 6). In the second loop, the tasks are repetitively selected through sampling $\mathrm{Mult}(\varphi)$, with their candidates added to R (line 10). The sampled task t and its candidate worker κ_t will be removed from further recommendation (line 11, 12) if their capacity constraints are reached (i.e., count_$t = c_t$ or count_$\kappa_t = c_{\kappa_t}$). Besides, new candidates will be re-sampled for task t (if it is not removed) and the tasks associated with κ_t (if κ_t is suspended); and $\boldsymbol{\mathcal{K}}$ will be updated accordingly (line 13). The following toy example is presented to illustrate the recommendation process.

Example2. *Suppose there are two tasks for recommendation, t_1 and t_2, each of which requires one worker. With their extracted semantic vectors, θ_{t_1} and θ_{t_2}, their categories* *are sampled as z_{t_1} and z_{t_2}; then, the tasks' candidates are sampled according to the z_{t_1}-th and z_{t_2}-th rows of* $\mathbf{U}$. *Suppose w_1 (whose capacity is 1) is selected to be both κ_{t_1} and κ_{t_2}. Then, in the 1st round of recommendation sampling, t_1 is selected; thus, w_1 is recommended to t_1; besides, w_1 and t_1 are exempted from further sampling. Suppose w_2 is selected as t_2's new candidate; and in the 2nd round of recommendation sampling, since only one task is left, t_2 is recommended with w_2. Finally, the recommendation is made as: t_1 with w_1, and t_2 with w_2.*

Accelerated Sampling with Alias Method. To reduce the sampling cost, the Vose's Alias method [22] is adopted. In particular, given an arbitrary multinomial distribution to be sampled (e.g., Mult(φ)), two tables, Alias and Probabilities, are created as the auxiliary structures to enable the O(1) sampling. Given a m dimensional distribution, the structures' creation requires O(m) time and space costs. In our sampling-based algorithm, the Utility Table is static, therefore, its auxiliary structures are constructed offline, with O(|W|) time and space costs (given K is a constant value); while, the other auxiliary structures (for type and recommendation sampling) need to be specifically constructed for each batch of presented tasks, which will lead to O($|\mathrm{T_{rec}}|$) time and space costs at running time. Notice that the Alias and Probabilities tables of Mult(φ) might be adjusted after $\boldsymbol{\mathcal{K}}$'s being updated (line 13, Alg. 2), it will merely take O(1) operations since only a constant number of entries in φ need to be adapted.

Efficiency and Effectiveness. Alg. 2 consists of two looping components, where the first one takes $|\mathrm{T_{rec}}|$ repetitions, and the second one incurs $\sum_{\mathrm{T_{rec}}} c_t$. Given the O(1) sampling and updating cost within each repetition, the overall time complexity is O($|\mathrm{T_{rec}}|$) (provide that c_t is a constant). Together with the time cost for auxiliary structures' construction, the time cost will still be O($|\mathrm{T_{rec}}|$).

The major space cost is resulted from the storage of utility table, tasks' types and candidates, together their auxiliary structures for sampling, which equals to O($|W| + |\mathrm{T_{rec}}|$).

Alg. 2 is lower-bounded by 1/4 of the optimal solution, which is formally stated as Theorem 3.

Theorem3. *The expected total* EFP *of sampling-based recommendation is no less than* 1/4 *of the optimal result.*

Proof. Presented in Section 9. □

7. EXPERIMENTAL STUDY

The experimental study is conducted to evaluate the following aspects of our proposed methods: 1) the effect of worker recommendation with triple-factor awareness, 2) the effect of parameter inference with only positive instances (referred as Positive-only Inference), 3) the effect of worker recommendation using sampling-based algorithm.

7.1 Experiment Settings

Triple-factor Aware Worker Recommendation. The triple-factor aware worker recommendation is compared with those single-factor approaches, which are conventionally used by the existing works (e.g., [15, 17]). In particular, the baseline methods include **Preference-Purely**, **Expertise-Purely** and **Activeness-Purely** (Pre, Exp and Act for short), where workers' preferences, expertises and activenesses are the solely factors considered in recommendation (i.e., in Pre, workers with the highest preferences will be recommended to a presented task; while, Exp/Act will recommend workers with the highest expertises/activenesses).

Following the conventional offline evaluation of recommendation system[4] [8, 18], we adopt "**Accumulative Answer Score**" (denoted as *Acc*) to measure the effectiveness of triple-factor aware worker recommendation. Such a metric serves as a joint indicator of answer production and answer quality. In particular, given each task t's worker recommendation R_t, the accumulative answer score (for the whole set of tasks to be recommended T_{rec}) is calculated as the sum of answers' scores:

$$Acc(\{\mathrm{R}_t\}_{T_{rec}}) = \sum\nolimits_{t \in T_{rec}} \sum\nolimits_{w \in \mathrm{R}_t} s_t^w. \qquad (19)$$

Some workers may not return their answers; in that case, the corresponding scores will be set to zero (i.e., $s_t^w = 0$ if w's answer is not returned). The physical meaning of Acc is easy to comprehend: given a fixed recommendation size, the algorithm leading to higher Acc indicates better identification of the desired workers (those who are expected to make timely and quality answers), thus reflecting its higher effectiveness for worker recommendation.

In addition, we also use the total number of acquired answers (denoted as #Ans) and the average score of acquired answers (which equals to $\frac{Acc}{\#\mathrm{Ans}}$) to get a more comprehensive view of the recommendation result.

Positive-only Inference. The positive-only inference, which is adopted for worker characterization, is compared with the traditional binary-class collaborative filtering technique (denoted as Bcf) [23, 13]. Briefly, Bcf would assign unit weight (i.e., 1) to the positive instances, and presents the negative ones with weight δ ($0 < \delta < 1$), making the negative instances less credible. Besides, instead of processing all the negative instances, Bcf uniformly samples $\lceil 1/\delta \rceil$ negative instances after processing each positive instance.

To evaluate the positive-only inference's quality, we adopt **Recall@K**, which is a widely used metric for inference quality of recommender systems (e.g., [23]). In particular, given workers with the top-K preferences towards task t (R_t^K) and all the workers who have answered t (W_t), it is defined that:

$$\mathrm{Recall@K} = |R_t^K \cap W_t| / |W_t|. \qquad (20)$$

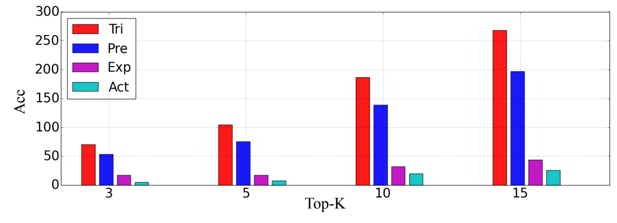
A high value of Recall@K means a large proportion of the recommended workers will give their answers to the presented task, thus reflecting better inference quality on workers' preferences and activenesses. Besides, the efficiency of positive-only inference is evaluated in terms of **time cost**. In particular, the overall running time of inferring the workers' characters is recorded for all the comparison methods.

Sampling-based Recommendation. As a solution for making the maximum weighted bipartite assignment, the sampling-based algorithm is compared with: 1) the Mixed Integer Programming (with Coin OR's solve[5]), which generates the optimal solution, and 2) the greedy method, which sequentially generates the approximation result with the descending order of the recommendation utilities (i.e., EFP).

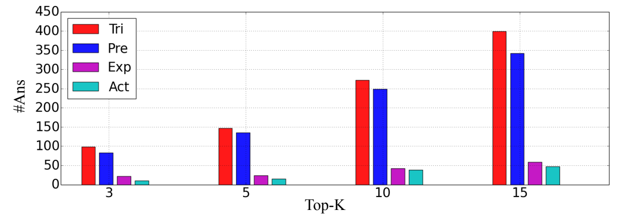
We adopt the optimal worker recommendation's objective, namely **Accumulative EFP** (AE for short), to evaluate

[4]A recommendation system is considered to be effective if it's able to accurately predict the users historical actions in the testing data, e.g., movies ratings for movie recommendation, or accumulated answer score in our work.

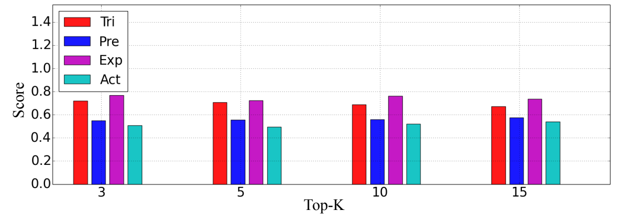
[5]https://projects.coin-or.org/Cbc



(a)



(b)



(c)

Figure 6: Experiment I: (a) result on Acc; (b) result on number of acquired answers; (c) result on average scores.

the effectiveness of sampling-based algorithm. In particular, given recommendation result $\{R_t\}_{T_{rec}}$, AE is computed as:

$$AE = \sum\nolimits_{t \in T_{rec}} \sum\nolimits_{w \in R_t} EFP(t, w). \tag{21}$$

In addition, we evaluate the efficiency of worker recommendation in terms of **time cost**, i.e., the time consumption of allocating all the tasks for the required number of workers.

We use real world data crawled from Stack Overflow for evaluation, where a total of 67080 tasks and 5547 workers (answers within 2016) are included. Besides, to further evaluate the performance of sampling-based recommendation, we create two synthetic datasets of tasks and workers, where the EFP is simulated with the Uniform distribution parameterized of 0 and 1: Unif$(0, 1)$, and the Bernoulli distribution parameterized of 0.05: Ber(0.05), respectively.

7.2 Triple-factor Recommendation Evaluation

We randomly choose 1000 tasks for the evaluation, where workers with the top-K utilities (i.e., the top-K EFP, preferences, expertises, activenesses for Tri, Pre, Exp and Act, respectively) are recommended for each task. The experimental results on accumulative answer scores (Acc), total number of acquired answers (#Ans) and average scores of acquired answers (Score) are presented in Figure 6.

The superiority of triple-factor aware worker recommendation is reflected in Figure 6 (a), where Tri gives rise to the highest Acc in each testing group (with K varied from 3 to 15). The reason about its superiority is quite straightforward, as Tri finds the workers who are not only probable to fulfill the given tasks on time, but also able to solve them correctly; while, all the baseline methods simply emphasize on one type of workers' character. Such an explanation can be further verified through the following observations.

Table 2: Experiment II-1: Recall@K (K = 10, 50, 100) and time cost comparisons between Positive-only Strategy and Binary-class Collaborative Filtering, where "δ-x" represents the Bcf with parameter δ equal to x.

	Recall@K			Time (s)
Methods	10	50	100	
Pos	**0.1090**	**0.3386**	**0.4906**	**49.48**
δ-0.5	0.0685	0.2391	0.3421	36.46
δ-0.2	0.0695	0.2457	0.3683	53.16
δ-0.1	0.0806	0.2597	0.3968	84.15
δ-0.05	0.0814	0.2649	0.4080	141.60
δ-0.02	0.0847	0.2705	0.4183	359.90

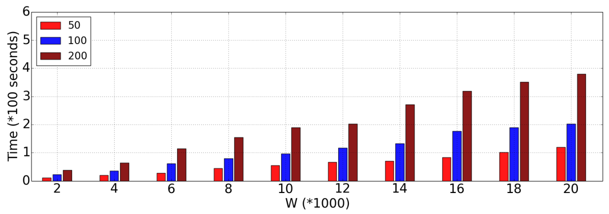


Figure 7: Experiment II-2. Time cost for Worker Characterization with variant workers' scales; red, blue and dark red bars represent settings of 50, 100 and 200 topics.

As demonstrated in Figure 6 (b), Tri leads to the largest number of acquired answers. Although Pre also tries to maximize the acquisition of answers (by recommending tasks to those workers who have the top preferences), it leaves out the factor of activeness, thus resulting in slightly lower #Ans; while Exp and Act's #Ans are significantly lower due to their complete neglect of workers' preferences towards different types of tasks. What's more, Tri's average scores are comparable to those of Exp (which recommends the workers with the top expertises), and considerably higher than Pre and Act (in Figure 6 (c)). Such a finding implies that Tri excludes the low-expertise workers, who are unlikely to produce high quality answers.

7.3 Positive-only Inference Evaluation

Comparison With Bcf. We measure Recall@K for a total of 1000 randomly chosen tasks and record the time consumption of worker characterization for all the comparison methods, whose results are presented in Table 2. It can be observed that Pos (standing for the positive-only inference) generates the largest Recall@K values and incurs comparatively low time cost (only greater than Bcf δ-0.5). That is to say, the adoption of Pos leads to the best inference quality, while preserving high running efficiency.

In addition, Bcf's inference quality can be improved to some extent with moderate decrement of δ (as Bcf shows the highest Recall@K when δ equals to 0.02). This is because a smaller value of δ will incorporate more training samples (as Bcf with parameter δ takes $1/\delta$ pseudo negative instances). However, the decrement of δ will inevitably result in two side effects: 1) the time consumption will grow considerably as more training instances need to be processed; 2) the pseudo negative instances are likely to be erroneous, which prevents Bcf's further improvement on inference quality (as Bcf δ-0.02's Recall@K is still much lower than that of Pos).

Scalability. We further evaluate Positive-only Inference's temporal scalability in terms of time cost with variant workers' scales. Particularly, workers are sampled through bootstrapping from the original dataset, whose size ranges from 2000 to 20,000. (In case of possible duplication, any pair

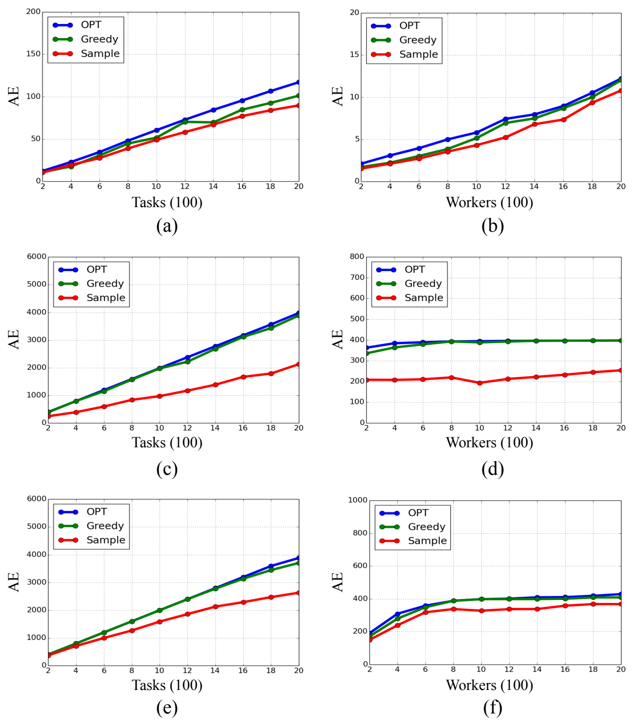


Figure 8: Experiment III-1. (a), (b): AE (Real) on variant number of tasks and workers; (c), (d) AE (Uniform) on variant number of tasks and workers; (e), (f): AE (Bernoulli) on variant number of tasks and workers.

of duplicated workers will be treated as different entities, i.e., different workers with the same answering record.) We select 3 candidate latent topic dimensions and present the results in Figure 7.

The overall time cost increases linearly with the growth of workers' scale, given the fixed topic dimension; besides, the overall time cost is roughly proportional to the topic dimension. Both observations are consistent with our theoretical analysis in Section 5.3. As such, the time cost won't be too large given a reasonable scope of workers' scale. In addition, two alternative methods can be adopted for further acceleration, in case of large-scale workers are presented. Firstly, as each worker's latent characters are updated independently (as presented in Alg. 1), worker characterization can be parallel conducted, where each worker's updating process is executed by a unique thread (similar manner to [34]). When adequate computing devices are provided, such an execution could cut down the time cost by orders. Secondly, because workers' characters will probably change steadily, incremental learning approaches (e.g., [7]) can be applied, which will omit the cost of inferring workers' characters all over again.

7.4 Sampling-based Recommendation Evaluation

The experiments are carried out with: (1) variant scales of tasks (200 to 2000) and a fixed number of workers (2000); (2) variant scales of workers (200 to 2000) and a fixed number of tasks (200). Both real and synthetic (Uniform and Bernoulli) datasets are employed. The resulted AE is reported in Figure 8 and the time cost is shown in Figure 9, where the following interesting points can be observed.

Firstly, the approximations' performances are clearly above their theoretical lower bounds (1/2 for Greedy and 1/4 for Sample). A necessary condition for the worst cases of Greedy and Sample is that all the tasks share a similar group of favorable workers, where tasks recommended in the initial stage will probably affect the subsequent ones. However, such a condition is almost impossible to hold in neither real nor synthetic settings: in real settings, tasks' recommendation utilities (i.e., EFP) are greatly diversified over the workers because of their differences on semantic vectors; while, in synthetic settings, tasks' EFP over workers are also significantly discrepant owing to their independent generation.

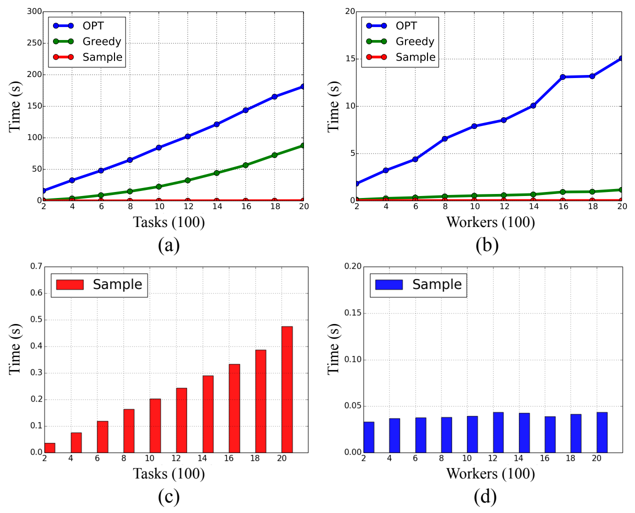


Figure 9: Experiment III-2. (a), (b): different methods' time costs on variant number of tasks and workers; (c), (d): Sample's time costs on variant number of tasks and workers.

Secondly, Sample's performance is close to Greedy and the optimal result (denoted as OPT) on the real (Figure 8 (a), (b)) and Bernoulli-synthesized (Figure 8 (e), (f)) data. For both real and Bernoulli-synthesized datasets, a task's total recommendation utility (over the whole workers) is densely concentrated on a small group of "favorable workers". (Such a "concentration" property is common in practice, as it is probable that only a tiny group of workers are highly appropriate to be recommended for a specific task.) As a result, such workers can be selected with significantly higher probabilities, which reduces the performance gap between Sample and Greedy. However, such a property does not hold on Uniform-synthesized data, as a task's generated EFP follows uniform distribution, thus enjoying greater mutual similarities. As a result, Sample's performance is comparable to Greedy and OPT under the real and Bernoulli-synthesized settings, rather than the Uniform-synthesized setting.

Thirdly, the performance ratio between Sample and Greedy becomes smaller with the tasks' scale's increase (Figure 8 (a), (c), (e)). In Alg 2, Sample randomly selects tasks and their candidate workers for recommendation with probabilities proportional to the corresponding EFP; meanwhile, Greedy can be regarded as the deterministic variation of Sample, as it sequentially makes recommendation with the descending order of EFP. When there are more tasks being presented, Sample's recommendation sampling operation (line 9, Alg. 2) becomes more randomized; thus, it resembles Greedy less and its resulted AE diverges from that of Greedy. Despite such a deficiency, Sample's performance is still comparable to Greedy and OPT on real data when tasks' scale reaches 2000, owing to the "concentration" property of real world data discussed in the second point.

At the same time, the time cost of Sample is considerably lower than the baselines (shown in Figure 9 (a), (b)); as Sample only requires O(|T|) rounds of sampling operation, and each of which can be completed in O(1) with Alias method. In addition, Sample's time cost is linear to the tasks' scale (Figure 9 (c)), and independent of workers' scale (Figure 9 (d)), which is consistent our theoretical analysis. Such a property is desirable for real applications where the recommendation could involve tremendous number of workers (e.g., a total of 6 million registered users in Stack Overflow).

7.5 Summary of Experiments

The major findings of the experimental study can be summarized with the following points.

- The triple-factor aware worker recommendation provides the given tasks with workers who will not only offer timely answers with large probabilities, but will also be capable of solving the tasks with high qualities, therefore leading to the highest total EFP of all the comparison methods.
- The adoption of positive-only inference results in the best inference quality, while preserving high running efficiency.
- The sampling-based method is able to make near optimal recommendation under the real world settings, with greatly smaller time cost. Besides, the sampling-based method shows satisfactory temporal scalability, as its running time is linear to the tasks' scale, and independent of the workers' scale.

8. CONCLUSION AND FUTURE WORK

In this work, we propose the Triple-factor Aware Worker Recommendation framework, which comprehensively considers the workers' expertises, preferences and activenesses to maximize the production of high quality answers. In this framework, the LHFM is constructed to accurately estimate the tasks' underlying categories and infer the workers' latent characters. Thanks to the adoption of positive-only inference, the model's parameters are learned with higher efficiency and better quality. Besides, the sampling-based algorithm is proposed to generate the near optimal worker recommendation in a greatly accelerated and temporally scalable way. Extensive experimental studies are carried out on both real and synthetic datasets, whose results verify the effectiveness and efficiency of our proposed methods. In the future work, we will consider TriRec's adaption to better fit users' individualized requests, which will make the system more interactive and user-friendly; for example, how to guarantee the acquisition of qualified answers within a certain specified time window.

Acknowledgment. The work is supported in part by the Hong Kong RGC Project 16202215, Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006, NSFC Grant No. 6172920161232018, Microsoft Research Asia Collaborative Grant and NSFC Guang Dong Grant No. U1301253.

9. APPENDIX

Proof of Theorem 1. According to the discussed probabilistic formulation, we are dealing with the following sequential process: *1*) a total of b_w candidates are browsed, where $b_w \sim$ Poisson(a_w); *2*) the selected candidates will be finally confirmed through Bernoulli selection Ber$(\rho_w \theta_t)$. Since we have a total of $|T|$ tasks, the probability of an arbitrary candidate being confirmed equals to $\sum_T \rho_w \theta_t / |T|$.

Now we transform such a process with the following equivalent form. According to the definition of Poisson distribution, the candidates' generation can be regarded as "M independent trials with success rate of a_w/M ($M \to \infty$)", where the probability of b_w successful trials is presented as:

$$P(b_w|a_w) = \binom{M}{b_w}(1 - \frac{a_w}{M})^{M-b_w}(\frac{a_w}{M})^{b_w}. \qquad (22)$$

To get confirmed in the end, a candidate trial also needs to pass the Bernoulli selection Ber$(\sum_T \rho_w \theta_t / |T|)$. Because the candidate generation and the final confirmation are independent, therefore, a trial success rate in such a sequential process is $\frac{a_w}{M} * \frac{\sum_T \rho_w \theta_t}{|T|}$. In other words, the number of successful trials for the whole sequential process follows:

$$P(b|a_w, \rho_w, \theta_t, T) = \binom{M}{b_w}(1 - \frac{\mathrm{U}}{M})^{M-b_w}(\frac{\mathrm{U}}{M})^{b_w}, \qquad (23)$$

where $\mathrm{U} = a_w \sum_T \rho_w \theta_t / |T|$. Therefore, the number of answers (i.e., Ψ_w) follows Poisson$(a_w \sum_T \rho_w \theta_t / |T|)$.

Proof of Theorem 2. The proof is made with the following two steps. Firstly, the incident of "task t's being answered by worker w (i.e., $A_t^w = 1$) during one unit time slot τ_0" is equivalent to the occurrence of following sequential events: *1*) w browses as least one task during one unit time slot τ_0; *2*) t is within w's browsed tasks; and 3) t is accepted by w. As such, "$A_t^w = 1$" is the specific instance of the sequential process discussed in the last proof, where each trial's success rate is $a_w \rho_w \theta_t / N$ and at least one trial will succeed in the end. Thus, the probability of "$A_t^w = 1$" equals to $P(b_w \geq 1 | a_w / \sum_T \rho_w \theta_t)$, which means:

$$P(A_t^w = 1|a_w, \rho_w, \theta_t, T) = 1 - e^{-a_w \rho_w \theta_t / |T|}. \qquad (24)$$

Secondly, given an arbitrary time period τ, it can be verified that the expected number of browsed tasks within such a time period will be $\frac{a_w |\tau|}{|\tau_0|}$. Replacing a_w in Eq. 24 with such a value, we have:

$$P(A_t^w = 1|a_w, \rho_w, \theta_t, T) = 1 - e^{-a_w \rho_w \theta_t |\tau| / |T||\tau_0|}. \qquad (25)$$

Therefore, Theorem 2 is justified.

Proof of Theorem 3. We sketch the proof with the following three parts. Firstly, given the optimal worker recommendation following the form of weighted bipartite assignment problem (with positive integral capacity constraints), the greedy solution, which sequentially picks edges with the descending order of weights, leads to an approximation result no less than 1/2 of the optimal one. Secondly, Alg. 2 makes recommendation with three sequential randomization steps, i.e., type sampling, candidates generation and recommendation sampling; for an arbitrary pair of task and worker (e.g., t and w), it is straightforward that the probability of w's being recommended to t, denoted as $\mathrm{P}_{t,w}$, follows $\mathrm{P}_{t,w} \propto \mathrm{EFP}(t, w)^{2+\epsilon}$, where ϵ is a positive number. Thirdly, in each repetition of the recommendation sampling (line 9, Alg. 2), the expected EFP of the newly added recommendation follows:

$$\begin{aligned}\sum_{\mathrm{T}'_{\mathrm{rec}}} \mathrm{EFP}(t, \kappa_t) * \mathrm{P}_{t,\kappa_t} &= \sum_{\mathrm{T}'_{\mathrm{rec}}} \frac{\mathrm{EFP}(t, \kappa_t) * \mathrm{EFP}(t, \kappa_t)^{2+\epsilon}}{\sum_{\mathrm{T}'_{\mathrm{rec}}} \mathrm{EFP}(t, \kappa_t)^{2+\epsilon}} \\ \geq \sum_{\mathrm{T}'_{\mathrm{rec}}} \frac{\mathrm{EFP}(t, \kappa_t) * \mathrm{EFP}(t, \kappa_t)}{\sum_{\mathrm{T}'_{\mathrm{rec}}} \mathrm{EFP}(t, \kappa_t)} &\geq \frac{\max_{\mathrm{T}'_{\mathrm{rec}}} \mathrm{EFP}(t, \kappa_t)}{2}.\end{aligned} \qquad (26)$$

i.e., in each repetition, the sampled recommendation's EFP is no less than 1/2 of the one selected by the greedy algorithm. Thus, the approximation ratio is above 1/4.

10. REFERENCES

[1] J. Bian, Y. Liu, D. Zhou, E. Agichtein, and H. Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *WWW 2009*, pages 51–60.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[3] A. C. Cameron and P. K. Trivedi. *Regression analysis of count data*, volume 53. Cambridge university press, 2013.

[4] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, Y. Tong, and C. J. Zhang. gmission: A general spatial crowdsourcing platform. *PVLDB*, 7(13):1629–1632, 2014.

[5] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, J. Han, and J. Zhao. Reliable diversity-based spatial crowdsourcing by moving workers. *PVLDB*, 8(10):1022–1033, 2015.

[6] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[7] E. Hazan, A. Rakhlin, and P. L. Bartlett. Adaptive online gradient descent. In *NIPS 2007*, pages 65–72.

[8] J. L. Herlocker, J. A. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53, 2004.

[9] N. Houlsby, J. M. Hernández-Lobato, and Z. Ghahramani. Cold-start active learning with robust ordinal matrix factorization. In *ICML 2014*, pages 766–774.

[10] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *GIS 2012*, pages 189–198.

[11] P. Mavridis, D. Gross-Amblard, and Z. Miklós. Using hierarchical skills for optimized task assignment in knowledge-intensive crowdsourcing. In *WWW 2016*, pages 843–853.

[12] J. D. Mcauliffe and D. M. Blei. Supervised topic models. In *NIPS 2008*, pages 121–128.

[13] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM 2008*, pages 502–511.

[14] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.

[15] M. Qu, G. Qiu, X. He, C. Zhang, H. Wu, J. Bu, and C. Chen. Probabilistic question recommendation for question answering communities. In *WWW 2009*, pages 1229–1230.

[16] F. Riahi, Z. Zolaktaf, M. Shafiei, and E. Milios. Finding expert users in community question answering. In *WWW 2012*, pages 791–798.

[17] J. San Pedro and A. Karatzoglou. Question recommendation for collaborative question answering systems with rankslda. In *RecSys 2014*, pages 193–200.

[18] G. Shani, A. Gunawardana, G. Shani, and A. Gunawardana. and asela gunawardana. evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297, 2011.

[19] M. Taddy. On estimation and selection for topic models. In *International Conference on Artificial Intelligence and Statistics*, pages 1184–1193, 2012.

[20] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Sharing clusters among related groups: Hierarchical dirichlet processes. In *NIPS 2005*, pages 1385–1392.

[21] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE 2016*, pages 49–60.

[22] M. D. Vose. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on Software Engineering*, 17(9):972–975, 1991.

[23] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD 2011*, pages 448–456.

[24] H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In *KDD 2015*, pages 1235–1244.

[25] G. C. Wei and M. A. Tanner. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990.

[26] H. Wu, Y. Wang, and X. Cheng. Incremental probabilistic latent semantic analysis for automatic question recommendation. In *RecSys 2008*, pages 99–106.

[27] L. Yang, M. Qiu, S. Gottipati, F. Zhu, J. Jiang, H. Sun, and Z. Chen. Cqarank: jointly model topics and expertise in community question answering. In *CIKM 2013*, pages 99–108.

[28] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing twitter and traditional media using topic models. In *ECIR 2011*, pages 338–349.

[29] Z. Zhao, L. Zhang, X. He, and W. Ng. Expert finding for question answering via graph regularized matrix completion. *TKDE*, 27(4):993–1004, 2015.

[30] L. Zheng and L. Chen. Mutual benefit aware task assignment in a bipartite labor market. In *ICDE 2016*, pages 73–84.

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

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

[33] G. Zhou, S. Lai, K. Liu, and J. Zhao. Topic-sensitive probabilistic model for expert finding in question answer communities. In *CIKM 2012*, pages 1662–1666.

[34] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. *AAIM 2008*, 5034:337–348.

[35] H. Zhu, E. Chen, H. Xiong, H. Cao, and J. Tian. Ranking user authority with relevant knowledge categories for expert finding. *World Wide Web*, 17(5):1081–1107, 2014.