Crowd-Selection Query Processing in Crowdsourcing Databases: A Task-Driven Approach

Zhou Zhao[†*], Furu Wei[§], Ming Zhou[§], Weikeng Chen[†], Wilfred Ng[†]
[†]Department of Computer Science and Engineering, Hong Kong University of Science and Technology
[§]Microsoft Research Asia, Beijing, China
[†]{zhaozhou, wilfred}@cse.ust.hk, [†]wchenad@connect.ust.hk,
[§]{fuwei,mingzhou}@microsoft.com

ABSTRACT

Crowd-selection is essential to crowdsourcing applications, since choosing the right workers with particular expertise to carry out specific crowdsourced tasks is extremely important. The central problem is simple but tricky: given a crowdsourced task, who is the right worker to ask? Currently, most existing work has mainly studied the problem of crowd-selection for simple crowdsourced tasks such as decision making and sentiment analysis. Their crowd-selection procedures are based on the trustworthiness of workers. However, for some complex tasks such as document review and question answering, selecting workers based on the latent category of tasks is a better solution.

In this paper, we formulate a new problem of task-driven crowd-selection for complex tasks. We first develop a bayesian generative model to exploit "who knows what" for the workers in the crowdsourcing environment. The model provides a principle and natural framework for capturing the latent skills of workers as well as the latent categories of crowdsourced tasks. The inference of the latent skills of workers is based on past resolved crowdsourced tasks with feedback scores. We assume that the feedback scores can illustrate the performance of the workers for the tasks. We then devise a variational algorithm that transforms the latent skill inference with the proposed model into a standard optimization problem, which can be solved efficiently. We verify the performance of our method through extensive experiments on the data collected from three well-known crowdsourcing platforms for question answering tasks such as Quora, Yahoo ! Answer and Stack Overflow.

1. INTRODUCTION

In recent years, crowdsourcing techniques [18] have attracted a lot of attention due to their effectiveness in real-life applications. They tackle the tasks including image tagging [19], decision making [12, 16] and natural language processing, which are hard for computers, but relatively easy for human workers. Some successful crowdsourcing examples for question answering tasks that appear include Quora [23], Yahoo ! Answer [5] and Stack Overflow [1], where users submit questions and get answers from the crowd. Using crowdsourcing techniques, these tasks can be solved well by human workers.

Despite the success of crowdsoucing techniques, the crowd-selection still remains challenging. Earlier approaches usually focus on the problem of crowd-selection for simple tasks where the selection procedure is based on the trustworthiness of workers [12, 7, 16, 31, 2]. However, in many cases, the task-driven crowd-selection is a better solution. Consider a question answering task t, "What are the advantages of B+ Tree over B Tree?". The existing crowd-selection procedure may not work in this case. Since the trustworthy workers may not be skilled in the area of computer science, therefore, they may not be able to answer this question.

In this paper, we formulate a new problem of task-driven crowd-selection. We focus on addressing the following three challenging issues.

- **Crowd Modeling.** We extend the crowd modeling from single-dimensional trustworthiness to multi-dimensional skills for task-driven crowd-selection. Thus, we model the strengths and weaknesses of workers on latent category of tasks. However, the existing latent skill models [28, 33] based on *Multinomial distribution*[1] are difficult to apply to our problem. The skill values on the latent categories are normalized to one for the property of Multinomial distribution. Thus, the skills of workers on specific latent categories cannot be comparable. For example, the latent skills of worker w^i is (CS 0.9, Math 0.1) and the latent skills of worker w^j is (CS 0.8, Math 0.2). Given a CS-based task above, the existing models select w^i since its skill value on CS is higher. However, it might be that w^j is better on CS while w^j solves more Math-based tasks. Thus, the skill value of their models cannot infer the strengths and weaknesses of workers on specific latent categories. We propose a novel crowd model to tackle this problem.
- **Latent Skill Inference.** The probabilistic inference for the latent skills of workers is based on the past resolved tasks. The existing inference approaches are either based on the content of tasks [28, 33, 32] or answer consistency with other workers [19]. However, we argue that the skills of workers are not necessarily related to the number of their resolve tasks or the difference between their answers and others. We consider the feedback score of the resolved tasks to illustrate the

*The work was done when the first author was visiting Microsoft Research, Beijing, China.

[1]http://en.wikipedia.org/wiki/Multinomial_distribution

10.5441/002/edbt.2015.35

latent skills of workers. We argue that the feedback score is a better quality measure for the job done by workers. Nowadays, the feedback score has been widely used in crowdsourcing platforms such as the satisfactory rate in Amazon Mechanical Turk [2] and Crowdflower [3] , and the thumbs-up in Question Answering System like Quora, Yahoo ! Answer and Stack Overflow. We propose a novel latent skill inference method based on resolved tasks with feedback scores.

- **Incremental Crowd-Selection.** The crowdsourced tasks are always in quantity and arriving in high speed. Therefore, it is time consuming for latent category inference of the crowdsourced tasks in batch. In this work, we first devise a bayesian model that builds a latent category space and infers the skills of workers on that space based on the past resolved tasks. Next, we propose an incremental latent category inference algorithm that projects the newly coming tasks into the existing latent category space. Then, we can select the workers who are skilled in these categories to solve the tasks.

In this paper, we propose a bayesian model for task-driven crowd-selection. Our contributions are summarized as follows:

- We first propose the problem of task-driven crowd-selection for crowdsourced tasks.
- We propose a novel crowd model for modeling the skills on latent category space that enables the task-driven crowd-selection. The model also makes the skills of workers on specific latent task categories become comparable.
- We make the use of the feedback scores of the resolved tasks for latent skill inference. We consider the the feedback scores as the quality measure of the job done by the workers.
- We devise a variational algorithm that transforms the complex latent skill inference problem into a high-dimensional optimization problem, which can be solved efficiently.
- We develop an incremental crowd-selection algorithm that chooses the right workers for coming crowdsourced tasks in the real time.
- We evaluate our algorithm on the data collected from three well-known crowdsourcing applications Quora, Yahoo ! Answer, and Stack Overflow. For all datasets tested, we show that the quality of the selected workers based on our crowd model is more superior to that of other existing worker models including TSPM [8] and DRM [28].

Organizations. The rest of the paper is organized as follows. Section 2 gives an overview of the architecture of our method. Section 3 surveys the related works. Section 4 presents our bayesian model for task-driven crowd-selection. Section 5 introduces a variational algorithm for our proposed model. We present the incremental crowd-selection procedure in Section 6. We report the experimental results in Section 7 and conclude the paper in Section 8.

2. OVERVIEW

In this section, we give an overview of the architecture for task-driven crowd-selection.

We illustrate the architecture of our task-driven crowd-selection system in Figure 1. The core component of our system is crowd manager. The main functionalities of crowd manager are latent skill inference for workers and choose the right crowd for given crowdsourced tasks. The crowd model is stored in the crowd databases which support crowd insertion, crowd update and crowd retrieval. The red lines show the process of latent skill inference for workers as well as build latent category space for tasks, which is based on resolved tasks with feedback scores. The crowd databases are then updated. The blue line show the process of task-driven crowd-selection. Given a coming crowdsourced task, the crowd manager first projects it into the built latent category space. Next, the crowd manager returns the workers online as the candidate crowd for this task. The crowd manager then ranks the workers who are skilled in this task. The top ranked workers are chosen for solving the task. After that, the task dispatcher distributes this task to the selected workers. Finally, the system keeps collect the answers return by the selected workers.

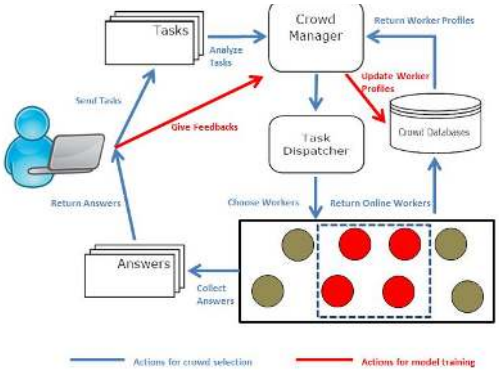


Figure 1: An Architecture for Task-Driven Crowd-Selection

In summary, our task-driven crowd-selection system can automatically ask the right crowd to process the crowdsourced tasks. The system is able to incrementally project the coming tasks to the existing latent category space such that the workers can be chosen in the real time. In the following sections, we present the idea and the methods of implementing this task-driven crowd-selection system.

3. RELATED WORK

Crowdsourcing has been widely used to solve challenging problems by human intelligence in comprehensive areas. Some successful applications that appear include CrowdDB [6], Qurk [14], CrowdSearch [29], HumanGS [15] and CDAS [12].

Recently, the crowdsourcing techniques have been applied in several research areas such as database management, machine learning and information retrieval. The crowdsourcing techniques on entity resolution were studied in [24, 26]. CrowdScreen [16] applied the crowdsourcing techniques in decision making. Guo et al. [9] and Venetis et al. [21] studied the problem of finding maximum element in the crowdsourcing databases. In [20], Trushkowsky et al proposed a method for crowdsourced enumerated query. In [4], Davidson et al proposed the top-k and group-by queries on crowdsourcing databases. Kaplan et al [11] aimed to select the right question for planing queries. Zhang et al [30] reduced the uncertainty for schema matching using crowdsourcing techniques. Marcus et al [13] studied the count query with the crowd. Park et al [17] aimed to find a best query query plan for crowdsourced data. Welinder et al [25] and Gao et al [7] proposed crowdsourcing based online algorithm to find the ground truth. Wu et al [27] studied the query

[2] https://www.mturk.com/
[3] http://crowdflower.com/

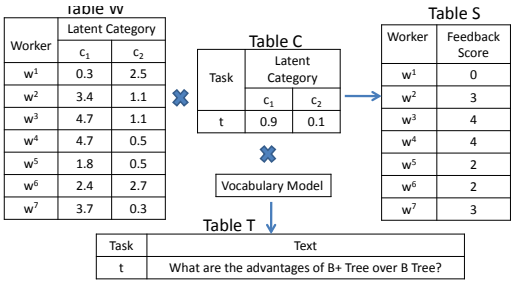


Figure 2: An Example of Generating Feedback Scores

processing over sensitive crowdsourced data.

The problem of crowd-selection is still the key component in these crowdsourcing applications, which has been studied in [12, 2, 3, 7]. However, these existing methods choose the crowd based on the trustworthiness of the workers, which may not be effective for complex crowdsourced tasks. In this paper, we study the problem of task-driven crowd-selection. We extend the crowd model from single-dimensional trustworthiness to multi-dimensional skills and propose a bayesian model for the skill variance of workers.

The most related works are TSPM [8] and DRM [28] for modeling the skills of workers. However, these methods model the skills of workers based on *Multinomial Distribution*, which we argue its limitation that it cannot distinguish the strengths and weaknesses of workers on specific latent category of the crowdsourced tasks in Section 1. In our experimental studies, we also show that the crowd-selection based on our novel crowd model outperforms both TSPM and DRM.

4. A BAYESIAN MODEL FOR TASK-DRIVEN CROWD-SELECTION

In this section, we present a bayesian model to infer the latent skills for workers as well as build the latent category of tasks for task-driven crowd-selection. The model exploits "who knows what" based on resolved tasks with feedback scores. Specifically, we calculate the posterior distribution of all possible worker skills, and find the most probable one with the maximum probability. Intuitively, this model best explains the feedback scores for the resolved tasks.

We start by illustrating an example of generating feedback scores on the jobs for a crowdsourced task, illustrated in Figure 2. After that, we introduce some basic notions and notations in Section 4.1, and define the problem in Section 4.2. Then, we present a generative process for task-driven crowd-selection in Section 4.3 and define the bayesian model in Section 4.4. We give a summary of notions and notations in Table 1.

Consider a question answering task "What are the advantages of B+ Tree over B Tree?" from Quora in Figure 2. We give the latent categories of this task in Table C. We assume that the vocabularies of this task are generated by its latent categories and the language model. The language model is vocabulary distribution over latent task categories. We can see that seven workers answered this question task and returned their answers. The latent skills of the workers are given in Table W and the feedback scores for their answers are given in Table S. The feedback scores are the number of thumbs-up for their returned answers in Quora. We assume that the feedback scores for the workers are proportional to their skills (i.e. $S \approx WC^T$). For instance, the feedback score of worker w_3 can be interpreted as $w_1^3 \times c_1 + w_2^3 \times c_2 = 4.7 \times 0.9 + 1.1 \times 0.1 \approx 4$.

Table 1: Summary of Notations

Notation	Meaning
T	A collection of N tasks $\{t^1, \ldots, t^N\}$
W	Latent skills of M workers $\{w^1, \ldots, w^M\}$
C	Latent categories on N tasks $\{c^1, \ldots, c^N\}$
A	Task assignment $\{a_{11}, \ldots, a_{ij}, \ldots, a_{MN}\}$
S	Feedback scores $\{s_{11}, \ldots, s_{ij}, \ldots, s_{MN}\}$
L	Number of vocabularies in the task
v_p^j	The p-th vocabulary in task t_j
z_p^j	Latent category of vocabulary v_p^j in task t_j
w^i	Latent skills of a worker $w^i = \{w_1^i, \ldots, w_K^i\}$
c^j	Latent categories of a task $c^j = \{c_1^j, \ldots, c_K^j\}$
Σ_c,ν_c	Prior for latent category
Σ_w,ν_w	Prior for worker skill
$\beta_{z_p^j}$	Prior for vocabulary

4.1 Notation

In this section, we introduce the notation used in the paper.

4.1.1 Crowdsourced Task T

The crowdsourced task t_j is represented as a bag of vocabularies $t_j = \{(v_1, \#v_1), (v_2, \#v_2), \ldots, (v_L, \#v_L)\}$ where $\#v_p$ is the number of vocabulary v_p in task t_j and L is the number of vocabularies. For example, the task t_j in Figure 2 can be represented as $t_j = \{(advantage, 1), (B, 1), (B+, 1), (over, 1), (tree, 2), (what, 1)\}$. We denote a collection of N crowdsourced tasks as T.

4.1.2 Latent Task Category C

The latent category of a task t_j is considered as a probability distribution $c^j = \{c_1^j, c_2^j, \ldots, c_K^j\}$ where K is the number of latent categories. We consider that c_k^j is the probability of task t_j in category c_k. The sum of the probability of a task t_j on the latent categories is $c_1^j + c_2^j \ldots + c_K^j = 1$. For example, the latent category of task t_j in Figure 2 is $c^j = \{c_1^j, c_2^j\} = \{0.9, 0.1\}$. We consider a collection of latent category of N crowdsourced tasks as C.

4.1.3 Latent Worker Skill W

The latent skills of worker w^i on K categories is $w^i = \{w_1^i, w_2^i, \ldots, w_K^i\}$ where w_k^i is a positive real number illustrating the ability of the worker on the latent category c_k. For example, the latent skills of workers w^2 is $\{w_1^2, w_2^2\} = \{3.4, 1.1\}$ and the latent skills of worker w^1 is $\{w_1^1, w_2^1\} = \{0.3, 2.5\}$. For latent category c_1 based task, employing w^2 is a better choice. On the other hand, the performance of w^2 is better on c_2 based task. We denote a collection of the latent skills of M workers as W. We explain the inference of W in Section 4.3.

4.1.4 Task Assignment A

The task assignment A is a $N \times M$ binary matrix where the entry a_{ij} indicates the assignment of task t_j to worker w^i (i.e. a_{ij} can be either 0 or 1). In this paper, we assume that a task can be assigned to more than one worker and a worker can have multiple tasks. For example, the assignment of task t_j and worker w^1 is $a_{1j} = 1$ in Figure 2. The task assignment A can be incremented when new tasks are resolved or new workers are involved in the crowdsourcing environment.

4.1.5 Task Feedback Score S

The task feedback score S is a $N \times M$ matrix where the entry s_{ij} indicates the score of the job done by worker w^i on task t_j. For example, the feedback score of worker w^2 on task t_j is 3 in Figure 2. The range of feedback score depends on the specific crowdsourcing applications. Here, we introduce two types of feedback scores used in this work.

- **Best Answer.** We consider the best answer as the feedback to illustrate the quality of answers in Yahoo ! Answer. The best answer is given by the question asker. Consider a resolved question answering task t_j. For the worker w^i receives best answer, the feedback score on the work is $s_{ij} = 1$. For other workers, we define feedback scores for them based on Jaccard distance between their answers and the best answer where $0 \leq s_{kj} \leq 1$ and $k \neq i$.

- **Thumbs-up.** We regard thumbs-up as the feedback to illustrate the quality of answers in both Quora and Stack Overflow. The thumbs-up for answers are given by the users of Quora and Stack Overflow. We consider the number of thumbs-up as the feedback score s_{ij}.

4.2 Problem Definition

We now formulate the problem of task-driven crowd-selection as follows:

Given a task t_j, how to choose the right online workers who are skilled in solving the task? We build a bayesian model based on resolved crowdsourced task (T, A, S) such that the following computational issues are effectively addressed: (1) For the coming task t_j, it can be projected to the latent category space c^j of our model. (2) After solving the task, the skills of workers involved can be updated. Thus, the task-driven crowd-selection is based on the strengths of workers on the latent category of the task $w^i(c^j)^T$. The top-k crowd-selection is to find a subset R of k workers such that

$$R = \arg\max_{|R|=k} \sum_{i \in R} w^i (c^j)^T \tag{1}$$

where $(c^j)^T$ is a transposed vector of latent category c^j.

4.3 A Generative Process

In this section, we illustrate the generative process for the feedback scores on the task-driven crowd-selection. For brevity, we show the generative process of the feedback scores $S = \{s_{11}, \ldots, s_{NM}\}$ based on N crowdsourced tasks $T = \{t_1, t_2, \ldots, t_N\}$ and M workers $W = \{w^1, w^2, \ldots, w^M\}$.

Now, we denote a set of model parameters $\varphi = \{W, \Sigma_w, C, \Sigma_c, \tau, \beta_c\}$. The parameters W and Σ_w are the prior for latent worker skill while C and Σ_c are the prior for latent task category. The parameter τ is variance between the feedback score and the predictive performance of the workers. The parameter β_c is the language model that generates the vocabularies of the task based on latent category c. We outline the generative process in Algorithm 1.

4.3.1 Generating Latent Worker Skill W

For the latent skills of worker $w^i \in W$, we assume that the skills on the latent categories are generated from a Normal distribution, given by

$$\begin{aligned} w^i &\sim Normal(\mu_w, \Sigma_w) \\ &\sim \frac{1}{(2\pi)^{K/2}|\Sigma_w|^{1/2}} \exp\{-\frac{1}{2}(w^i - \mu_w)^T \Sigma_w^{-1}(w^i - \mu_w)\} \end{aligned} \tag{2}$$

where μ_w is the mean of the skills on latent categories and Σ_w is the correlation of skills on latent categories. It is a generalized way to model the skills on latent categories while a special way is to assume the independence of skills on latent categories. In that case, Σ_w is a diagonal matrix.

4.3.2 Generating Latent Task Category C

For the latent category of task $t_j \in T$, we assume that c^j is generated from a Normal distribution, given by

$$\begin{aligned} c^j &\sim Normal(\mu_c, \Sigma_c) \\ &\sim \frac{1}{(2\pi)^{K/2}|\Sigma_c|^{1/2}} \exp\{-\frac{1}{2}(c^j - \mu_c)^T \Sigma_c^{-1}(c^j - \mu_c)\} \end{aligned} \tag{3}$$

where μ_c is the latent category distribution for all the crowdsourced tasks. The parameter Σ_c is the correlation of latent category distribution for all the crowdsourced tasks.

4.3.3 Generating Task Vocabulary V

For the latent category of vocabulary v_p^j in task t_j, we assume z_p^j is generated from a discrete distribution, given by

$$\begin{aligned} z_p^j &\sim Discrete(logistic(c^j)) \\ &\sim \frac{\exp(c_k^j)}{\sum_{k=1}^K \exp(c_k^j)}. \end{aligned} \tag{4}$$

where $logistic(c^j)$ is a logistic function that transforms the latent category c^j to a discrete distribution.

Based on the latent category z_p^j, the vocabulary is generated from a discrete distribution, given by

$$v_p^j \sim \beta_{z_p^j} = p(v_p^j | \beta, z_p^j) \tag{5}$$

where β is a vocabulary distribution over latent categories. The parameter β is used as the language model to generate the vocabularies of all the crowdsourced tasks.

4.3.4 Generating Task Feedback Score S

For the feedback score s_{ij} on the task t_j assigned to the worker w^i, we assume that s_{ij} is generated from a Normal distribution, given by

$$\begin{aligned} s_{ij} &\sim Normal(w^i (c^j)^T, \tau) \\ &\sim \frac{1}{\tau\sqrt{2\pi}} \exp\{-\frac{(s_{ij} - w^i (c^j)^T)^2}{2\tau^2}\} \end{aligned} \tag{6}$$

where the parameter τ is the variance of the Normal distribution. The product $w^i(c^j)^T$ is the predictive performance of worker w^i on the task t^j.

We now present the details of the generative process for the feedback scores S on the crowdsourced task T in Algorithm 1. Algorithm 1 generates the skills of workers by Normal distribution with parameters μ_w and Σ_w from Line 1 to Line 3. Next, Algorithm 1 generates latent categories for the crowdsourced tasks by Normal distribution with parameters μ_c and Σ_c in Line 5. The latent category of the vocabulary z_p^j is generated by a discrete distribution with logistic function based on c^j in Line 7. Then, the vocabularies of task t_j is generated by the language model β and the latent category for vocabularies z^j in Line 8. After that, Algorithm 1 generates the feedback scores S on the workers for the tasks T from Line 1 to Line 15. The feedback score is generated by Normal distribution based on the predictive performance of workers on the task $w^i(c^j)^T$. Finally, Algorithm 1 returns the feedback score S in Line 16.

Algorithm 1 Generating feedback scores for resolved tasks

Input: A set of tasks T, task assignment A, a set of model parameters $\varphi = \{W, \Sigma_w, C, \Sigma_c, \tau, \beta_c\}$
Output: Feedback scores S

for each worker $w^i \in W$ **do**
  Choose skills w^i by Equation 2.
end for
for each task $t_j \in T$ **do**
  Choose latent category c^j by Equation 3
  for each vocabulary $v_p^j \in t_j$ **do**
    (a) Choose a latent category z_p^j by Equation 4
    (b) Choose the text for vocabulary v_p^j by Equation 5
  end for
end for
for each crowdsourced task $t^j \in T$ **do**
  for each employed worker $w^i \in A_j$ **do**
    Generate feedback score s_{ij} by Equation 6
  end for
end for
return feedback scores S

4.4 Model Definition

In the previous discussion, we described a generative process for the feedback scores on the crowdsourced tasks. We now formally define a bayesian model that represents the underlying joint distribution over the vocabularies of tasks V and the feedback scores of tasks S.

Given a set of model parameters $\varphi = \{\mu_w, \Sigma_w, \mu_c, \Sigma_c, \tau, \beta\}$, and task assignment A, we factorize the joint distribution over V and S, given by

$$\begin{aligned} p(V,S|A,\varphi) &= \int_C \int_W p(W|\mu_w,\Sigma_w)p(C|\mu_c,\Sigma_c)p(Z|C) \\ &\times\ p(V|Z,\beta)p(S|WC^T,\tau)dCdW \end{aligned}$$

where

$$\begin{aligned} p(W|\mu_w,\Sigma_w) &= \prod_{w^i\in W} p(w^i|\mu_w,\Sigma_w), \\ p(C|\mu_c,\Sigma_c) &= \prod_{c^j\in C} p(c^j|\mu_c,\Sigma_c), \\ p(Z|C) &= \prod_{t_j\in T}\prod_{p=1}^{L} discrete(logistic(c^j)), \\ p(V|Z) &= \prod_{t_j\in T}\prod_{p=1}^{L} \beta_{z_p^j}, \\ p(S|WC^T,\tau) &= \prod_{t_j\in T}\prod_{a_{ij}=1} p(s_{ij}|w^i(c^j)^T,\tau), \end{aligned}$$

and $p(w^i|\mu_w,\Sigma_w)$, $p(c^j|\mu_c,\Sigma_c)$, $discrete(logistic(c_j))$, $\beta_{z_p^j}$, and $p(s_{ij}|w^i(c^j)^T,\tau)$ are defined from Equation 2 to Equation 6, respectively. For brevity, we omit the conditional part of the joint distribution $p(V,S|A,\varphi)$ and abbreviate it to $p(V,S)$ in the rest of this paper.

Based on the model, the problem of worker skill estimation problem can be transformed into a probabilistic inference problem, namely, finding the maximum a posterior (MAP) configuration of the worker skill W and latent task category C conditioning on the resolved tasks (V,S). That is to find

$$(W^*, C^*, Z^*) = \arg\max_{W,C,Z} p(W,C,Z|V,S) \tag{7}$$

where $p(W,C,Z|V,S)$ is the posterior distribution of W and C given collected resolved tasks (V,S). However, it is difficult to compute the joint posterior distribution of W and C,

$$p(W,C,Z|V,S) = \int_\varphi p(W,C,Z,\varphi|V,S)d\varphi \tag{8}$$

where

$$p(W,C,Z,\varphi|V,S) = \frac{p(W,C,Z,\varphi,V,S)}{\int_{W,C,Z,\varphi} p(W,C,Z,\varphi,V,S)dWdCdZd\varphi}.$$

This distribution is intractable to compute due to the coupling between the model parameters in φ. To tackle this problem, we develop an efficient and effective approximation in the next section.

5. A VARIATIONAL ALGORITHM

In this section, we propose a variational algorithm to approximate the distribution $p(W,C,Z|V,S)$ defined in Equation 8. The basic idea of our variational algorithm is to approximate the distribution $p(W,C,Z|V,S)$ using a variational distribution $q(W,C,Z)$ that is tractable for the maximization over W, C and Z in Equation 7.

5.1 Family of Variational Distributions

We restrict the variational distribution to a family of distributions that factorize as follows:

$$q(W,C,Z) = \prod_{i=1}^{M} q(w^i)\prod_{j=1}^{N}(q(c^j)\prod_{p=1}^{L} q(z_p^j)).$$

Then, we further require the distribution in this family to take the following parametric form:

$$\begin{aligned} & q(W,C,Z|\lambda_w,\nu_w,\lambda_c,\nu_c,\phi) \\ =\ & \prod_{i=1}^{M} q(w^i|\lambda_w^i, diag((\nu_w^i)^2))\prod_{j=1}^{N}(q(c^j|\lambda_c^j, diag((\nu_c^j)^2)) \\ \times\ & \prod_{p=1}^{L} q(z_p^j|\phi_p^j)), \end{aligned}$$

where

$$\begin{aligned} q(w^i|\lambda_w^i, diag((\nu_w^i)^2)) &= Normal(\lambda_w^i, diag((\nu_w^i)^2)), \\ q(c^j|\lambda_c^j, diag((\nu_c^j)^2)) &= Normal(\lambda_c^j, diag((\nu_c^j)^2)), \\ q(z_p^j|\phi_p^j) &= discrete(\phi_p^j). \end{aligned}$$

$diag(\cdot)$ is a diagonal matrix where the entries outside the main diagonal are all zero. Here λ_w^i, $diag((\nu_w^i)^2)$, λ_c^j, $diag((\nu_c^j)^2)$, ϕ_p^j are variational parameters. For brevity, we denote the collection of variational parameters as $\varphi' = \{\lambda_w, diag(\nu_w^2), \lambda_c, diag(\nu_c^2), \phi\}$

in the rest of the paper. Thus, the inference for W, C, and Z in Equation 7 can be simplified as follows:

$$\begin{aligned}
& (W^*, C^*, Z^*) \\
= \; & [\arg\max_{w^1} q(w^1), \ldots, \arg\max_{w^M} q(w^M), \\
& \arg\max_{c^1} q(c^1), \ldots, \arg\max_{c^N} q(c^N), \\
& \arg\max_{z^1} q(z^1), \ldots, \arg\max_{z^N} q(z^N)] \\
= \; & [\arg\max_{\lambda_w^1, diag((\nu_w^1)^2)} q(w^1), \ldots, \arg\max_{\lambda_w^M, diag((\nu_w^M)^2)} q(w^M), \\
& \arg\max_{\lambda_c^1, diag((\nu_c^1)^2)} q(c^1), \ldots, \arg\max_{\lambda_c^N, diag((\nu_c^N)^2)} q(c^N), \\
& \arg\max_{\phi^1} q(z^1), \ldots, \arg\max_{\phi^N} q(z^N)].
\end{aligned}$$

5.2 Stationary Points of $L(q)$

The goal of the variational algorithm is to find the variational distribution that is close to the true posterior $p(W, C, Z|V, S)$. This is equivalent to optimizing the variational parameters φ' with respect to some distance measure, given by

$$\varphi' = \arg\max_{\varphi'} D(q(\varphi')||p(W, C, Z|V, S))). \quad (9)$$

In this work, we adopt the Kullback-Leibler (KL) divergence which is commonly used to measure the difference between two distributions. It is defined as

$$KL(q||p) = \int_{\varphi'} q(\varphi') \log \frac{q(\varphi')}{p(W, C, Z|V, S)} d\varphi'$$

where KL divergence is a function of the variational parameters λ_w, $diag(\nu_w^2)$, λ_c, $diag(\nu_c^2)$, ϕ. However, directly optimizing the KL divergence is infeasible because the KL divergence involves the term $p(W, C, Z|V, S)$, which is intractable.

Instead, we solve an equivalent maximization problem, whose objective function is defined as

$$\begin{aligned}
\mathcal{L}(q) &= \int_{\varphi'} q(\varphi') \log \frac{p(W, C, Z, V, S)}{q(\varphi')} d\varphi' \\
&= E_q[\log p(W|\mu_w, diag(\nu_w^2))] + E_q[\log p(V|Z, \beta)] \\
&+ E_q[\log p(C|\mu_c, diag(\nu_c^2))] + E_q[\log p(Z|C)] \\
&- E_q[\log p(W|\lambda_w, diag(\nu_w^2))] - E_q[\log p(Z|\phi)] \\
&- E_q[\log p(C|\lambda_c, diag(\nu_c^2)] + E_q[\log p(S|WC^T, \tau)].
\end{aligned}$$

The expectations are taken with respect to the variational distribution $q(\cdot|\cdot)$ and subscripts denote the variational parameters involved in the expressions.

The equivalent between these two optimization problem can easily be seen as their objective functions sum up to a constant

$$KL(q||p) + \mathcal{L}(q) = \log p(V, S).$$

However, it is difficult to compute the expected log probability of $p(Z|C)$ in Equation 5. To preserve the lower bound of $\mathcal{L}$(q), we upper bound the log normalize with a Taylor expansion, given by

$$\begin{aligned}
E_q[\log p(Z|C)] &= E_q[Z^T C] - E_q[\log(\sum_{k=1}^{K} \exp\{C_k\})] \\
&\geq E_q[Z^T C] - \varepsilon^{-1}(\sum_{k=1}^{K} E_q[\exp\{C_k\}]) \\
&+ 1 - \log\varepsilon = E'_q[\log p(Z|C)].
\end{aligned}$$

where ε is a new variational parameter. Thus, we replace the term $E_q[\log p(Z|C)]$ with $E'_q[\log p(Z|C)]$ in $\mathcal{L}(q)$ and obtain a lower bound of $\mathcal{L}(q)$, denoted as $\mathcal{L}'(q)$.

In order to maximize the objective function $\mathcal{L}(q)$, we take the derivatives of its lower bound $\mathcal{L}'(q)$ with respect to the variational parameters λ_w, ν_w, λ_c, ν_c, ϕ, ε, and set these derivatives to zeros.

$$\nabla_{\varphi'} \mathcal{L}'(q) = (\frac{\partial \mathcal{L}'(q)}{\partial \lambda_w}, \frac{\partial \mathcal{L}'(q)}{\partial \nu_w}, \frac{\partial \mathcal{L}'(q)}{\partial \lambda_c}, \frac{\partial \mathcal{L}'(q)}{\partial \nu_c}, \frac{\partial \mathcal{L}'(q)}{\partial \phi}, \frac{\partial \mathcal{L}'(q)}{\partial \varepsilon}) = \overrightarrow{0}.$$

For clarity, we put all the derivations in Appendix 10.1. We report the solutions to the optimization problem by

$$\begin{aligned}
\lambda_w^i &= (\Sigma_w^{-1} + \frac{1}{\tau^2} \sum_{t_j : a_{ij}=1} ((\lambda_c^j)^T \lambda_c^j + diag((\nu_c^j)^2)))^{-1} \\
&\times (\Sigma_w^{-1} \mu_w + \frac{1}{\tau^2} \sum_{t_j : a_{ij}=1} s_{ij} \lambda_c^j), \quad (10) \\
(\nu_{w,k}^i)^2 &= (\sum_{t_j : a_{ij}=1} \frac{(\lambda_{c,k}^j)^2 + (\nu_{c,k}^j)^2}{\tau^2} + \Sigma_{w,kk}^{-1})^{-1}, \quad (11) \\
\phi_p^j &\propto \exp(\lambda_{c,v}^j + \frac{1}{L} \sum_{j=1}^{T} \sum_{p=1}^{L} 1[v_p^j = v] \log \beta_{c,v}), \quad (12) \\
\varepsilon_j &= \sum_{k=1}^{K} \exp(\lambda_{c,k}^j + \frac{(\nu_{c,k}^j)^2}{2}), \quad (13)
\end{aligned}$$

for all $i = 1, \ldots, M$, $k = 1, \ldots, K$, and $v = 1, \ldots, V$.

However, $\mathcal{L}'(q)$ is not amenable to analytic maximization with respect to λ_c^j and ν_c. Thus, we use the conjugate gradient algorithm with derivative

$$\begin{aligned}
\frac{\partial \mathcal{L}'(q)}{\partial \lambda_c^j} &= (\Sigma_c^{-1} + \frac{1}{\tau^2} \sum_{w^i : a_{ij}=1} (\lambda_w^i (\lambda_w^i)^T + diag((\nu_w^i)^2))) \lambda_c^j \\
&+ \Sigma_c^{-1} \mu_c + \frac{1}{\tau^2} \sum_{w^i : a_{ij}=1} s_{ij} \lambda_w^i + L\phi^j \\
&- \frac{1}{\varepsilon_j} \exp\{\lambda_c^j + \frac{(\nu_c^j)^2}{2}\} \quad (14)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}'(q)}{\partial \nu_{c,k}^j} &= (\sum_{w^i : a_{ij}=1} \frac{(\lambda_{w,k}^i)^2 + (\nu_{w,k}^i)^2}{\tau^2} + \Sigma_{c,kk}^{-1})(\nu_c^j)^2 \\
&- \frac{1}{\varepsilon_j} \exp\{\lambda_c^j + \frac{(\nu_c^j)^2}{2}\} \quad (15)
\end{aligned}$$

for all $j = 1, \ldots, N$ and $k = 1, \ldots, K$.

5.3 Optimization Procedure

Based on the estimated stationary points, we optimize the model parameters $\varphi = \{\mu_w, \Sigma_w, \mu_c, \Sigma_c, \tau, \beta_{c,v}\}$ in this section.

We take the derivative of the lower bound $\mathcal{L}'(q)$ with respect to the model parameters φ, and set these derivatives to zeros.

$$\nabla_{\varphi} \mathcal{L}'(q) = (\frac{\partial \mathcal{L}'(q)}{\partial \mu_w}, \frac{\partial \mathcal{L}'(q)}{\partial \Sigma_w}, \frac{\partial \mathcal{L}'(q)}{\partial \mu_c}, \frac{\partial \mathcal{L}'(q)}{\partial \Sigma_c}, \frac{\partial \mathcal{L}'(q)}{\partial \tau}, \frac{\partial \mathcal{L}'(q)}{\partial \beta_{c,v}}) = \overrightarrow{0}.$$

For clarity, we put all the derivations in Appendix 10.2. We report the solutions to the optimization problem by

Algorithm 2 Iterative Optimization Algorithm

Input: a set of task assignments A, a set of resolved tasks (W, T, R),a limit on the number of iterations n_{max}
Output: variational parameters φ' and model parameters φ

1: $n \leftarrow 0$
2: **repeat**
3: (a) Given φ, update φ' according to Equations (10)-(15)
4: (b) Given φ', update φ according to Equations (16)-(21)
5: $n \leftarrow n + 1$
6: **until** $\mathcal{L}'(q^{(n)}) - \mathcal{L}'(q^{(n-1)}) \leq \epsilon$ or $n > n_{max}$
7: **return** φ' and φ

$$\mu_w = \frac{1}{M}\sum_{i=1}^{M}\lambda_w^i \quad (16)$$

$$\Sigma_w = \frac{1}{M}\sum_{i=1}^{M}(diag((\nu_w^i)^2) + (\lambda_w - \mu_w)(\lambda_w - \mu_w)^T) \quad (17)$$

$$\mu_c = \frac{1}{N}\sum_{j=1}^{N}\lambda_c^j \quad (18)$$

$$\Sigma_c = \frac{1}{N}\sum_{j=1}^{N}(diag((\nu_c^j)^2) + (\lambda_c - \mu_c)(\lambda_c - \mu_c)^T) \quad (19)$$

$$\begin{aligned}\tau^2 &= \frac{1}{|A|}\sum_{a_{ij}=1}\{s_{ij}^2 + (\lambda_w^i)^T diag((\nu_c^j)^2)\lambda_w^i \\ &+ (\lambda_c^j)^T diag((\nu_w^i)^2)\lambda_c^j - 2s_{ij}(\lambda_w^i)^T\lambda_c^j \\ &+ ((\lambda_w^i)^T\lambda_c^j)^2 + Tr(diag((\nu_w^i)^2)diag((\nu_c^j)^2))\} \quad (20)\end{aligned}$$

$$\beta_{k,v} \propto \sum_{j=1}^{T}\sum_{p=1}^{L}\phi_{p,k}^j 1(v_p^j = v) \quad (21)$$

for all $s = 1, \ldots, K$ and $v = 1, \ldots, V$.

We present our optimization method in Algorithm 2. Algorithm 2 iteratively updates the variational parameters φ' and model parameters φ in Lines 3 and 4 until the objective function becomes convergent. The objective function $\mathcal{L}'(q)$ can be persistently improved by variational algorithm, stated in [22]. Because the value of $\mathcal{L}'(q)$ is finite, Algorithm 2 is guaranteed to converge with a finite number of iterations.

6. CROWD-SELECTION ALGORITHM

In this section, we introduce our crowd-selection algorithm based on bayesian model.

Given a new crowdsourced task t_j, we want to estimate its latent category such that the workers skilled in c^j can be selected. We first compute its variational parameters λ_c^j and ν_c^j under our bayesian model with parameters $\{\mu_c, \Sigma_c, \beta_{k,v}\}$. The estimation procedure is similar to Algorithm 2, but the terms depending on worker skill and feedback scores are removed. Thus, we compute its latent category using conjugate gradient algorithm with derivative

$$\frac{\partial\mathcal{L}'(q)}{\partial\lambda_c^j} = \Sigma_c^{-1}\lambda_c^j + \Sigma_c^{-1}\mu_c + L\phi^j - \frac{1}{\varepsilon_j}\exp\{\lambda_c^j + \frac{(\nu_c^j)^2}{2}\} \quad (22)$$

$$\frac{\partial\mathcal{L}'(q)}{\partial\nu_{c,k}^j} = \Sigma_{c,kk}^{-1}(\nu_c^j)^2 - \frac{1}{\varepsilon_j}\exp\{\lambda_c^j + \frac{(\nu_c^j)^2}{2}\} \quad (23)$$

Algorithm 3 Task-Driven Crowd-Selection Algorithm

Input: A task t_j, worker skill W, model parameters φ, number of workers k, a limit on the number of iterations n_{max}
Output: A set of selected workers R

1: $n \leftarrow 0$
2: **repeat**
3: (a) Update variational λ_c^j and ν_c^j by Equations (22)-(23)
4: (b) Update ϕ^j and ε_j by Equations (12)-(13)
5: **until** $n > n_{max}$
6: Sample $c^j \sim Normal(\lambda_c^j, \nu_c^j)$
7: Choose selected workers R by Equation 1
8: **return** selected workers R

and the update for ϕ^j and ε_j are given in Equations 12 and 13.

We now present the details of the task-driven crowd-selection in Algorithm 3. Given a new task t_j and model parameters φ, Algorithm 3 selects top-k workers R for this task. In the first phrase, Algorithm 3 computes the variational parameters λ_c^j and ν_c^j for the latent category of task t_j from Line 2 to Line 5. The latent category for task t_j is sampled by a Normal distribution with mean λ_c^j and variance ν_c^j in Line 6. In the second phrase, Algorithm 3 chooses a set of workers R based on worker skill W and latent category c^j by Equation 1 in Line 7. Finally, Algorithm 3 returns the selected workers R in Line 8.

7. EXPERIMENTAL STUDY

In this section, we evaluate the performance of our algorithms. All the algorithms, including those we compared within the experiments, were implemented in Java (we will release all our source codes if the paper is published) and tested on machines with Windows OS, Intel(R) Core(TM2) Quad CPU 2.66Hz, and 60GB of RAM memory.

7.1 Datasets

We collect the data from three well-known crowdsourcing applications Quora, Yahoo ! Answer and Stack Overflow. Some statistics of the datasets are reported in Table 2.

7.1.1 Quora

We gathered our Quora dataset through web-based crawls between August and early September 2012. We limited these crawls to 10 requests/second to minimize the impact on Quora[4].

We start our crawls using 100 randomly selected questions. The crawls follow a BFS pattern through the related questions link for each question. In total, we collect 444,000+ unique questions. Each question page contains a complete list of answers, and the respondent and voters for each answer. As shown in Table 2, this question-based crawl produced 444,000+ unique questions, 887,000+ unique answers, and 95,000+ unique users who answered a question. For each answer, we consider the number of thumbs-up voted by the crowd as the quality measure.

7.1.2 Yahoo ! Answer

We collect our Yahoo ! Answer dataset[5] through its API from Jan, 2012 onwards. In total, we collect 8866,000+ unique questions. On the average, each question has around three respondents.

[4] http://www.quora.com/
[5] http://answers.yahoo.com/

Table 2: Statistics of Real Datasets

Dataset	Total Questions	Total Users	Total Answers
Quora	444k	95k	887k
Yahoo ! Answer	8866k	1004k	26903k
Stack Overflow	83k	15k	236k

We also crawl the best answer for each question from Yahoo ! Answer, which is used to mark the best answerer. As shown in Table 2, the API-based crawl produced 8866,000+ unique questions, 26903,000+ unique answers, and 1004,000+ unique users who answered a question.

7.1.3 Stack Overflow

We download the Stack Overflow dataset from the website[6]. This dataset containing the questions and the related answerers posted between February 18, 2009 and June 7, 2009. For each answer of a question, the Stack Overflow provides the score of the answer. We consider the answer with the highest score as the best answer. The statistics of the dataset is given in Table 2.

7.2 Experimental Settings

We detail the experimental settings in this subsection, including the algorithm for comparison, the measures we use to assess the performance of the algorithms.

7.2.1 Algorithms for Comparison

We compare our task-driven crowd-selection algorithm, denoted as **TDPM** (Task-Driven Probabilistic Model), to several state-of-the-art algorithms such as Vector Space Model (VSM), Dual Role Model (DRM) [28] and Topic Sensitive Probabilistic Model (TSPM) [8].

- **VSM.** The VSM algorithm selects the workers based on the cosine similarity between the crowdsourced task and the historical tasks resolved by the workers. Consider t^j as a bag of vocabularies for the given task defined in Section 4.1. Then, we define t_w^i as a bag of vocabularies of the task resolved by worker w^i where $t_w^i = \bigcup_{t^j:a_{ij}=1} t^j$. The ranking score of worker w^i is given by

$$\hat{s_{ij}} = \frac{(t^j)^T t_w^i}{\sqrt{(t^j)^T t^j}\sqrt{(t_w^i)^T t_w^i}}.$$

  where the numerator $(t^j)^T t_w^i$ is a dot product of vocabulary vectors t^j and t_w^i.

- **DRM.** The DRM algorithm models the skills of workers as a Multinomial distribution. Then it carries out the estimation using **Probabilistic Latent Semantic Analysis** [10] for skills of worker w^i as well as the latent category of the task c^j. Given a crowdsourced task t^j, the crowd-selection of DRM on workers is proportional to the predictive performance $w^i(c^j)^T$.

- **TSPM.** The TSPM algorithm also models the skills of workers as a Multinomial distribution. The estimation of worker skill w^i and latent category of the task c^j are based on **Latent Dirichlet Allocation**. Similarly, the crowd-selection of TSPM on workers is based on the predictive score of workers on task t^j which is $w^i(c^j)^T$.

[6] http://www.ics.uci.edu/~duboisc/stackoverflow/

However, the sum of weights on the skills of all workers are normalized to one (i.e. $\sum_{k=1}^{K} w_k^i = 1$) because the property of Multinomial distribution. We argue that their models can not compare the skills of workers on a specific latent category (i.e. $w_k^i > w_k^j$?). We devise another model-based approach to tackle this problem.

7.2.2 Job Quality Assessment

We assess the performance of our algorithm by two measurements: *precision* and *recall*.

We employ the formula *ACCU* to measure the precision of the crowd-selection algorithms, which is also used in DRM [28] and TSPM [33]. ACCU is defined as the ratio of the rank of the right worker to the total number of candidate workers. The formula of ACCU is given by

$$ACCU = \frac{|R| - R_{best} - 1}{|R| - 1}$$

where R is the set of selected workers and R_{best} is the rank of right worker in R. We consider the rank of the best answerer as R_{best} in Yahoo ! Answer while we regard the rank of the worker with highest score as R_{best} in Quora and Stack Overflow.

We propose to use *TopK* to measure the recall of the crowd-selection algorithm. The TopK is defined as the ratio of the number of times that the rank of right worker is less than K to the total number of crowdsourced tasks N. The formula of TopK is given by

$$TopK = \frac{|\{t^j | R_{best}^j \leq K\}|}{N}.$$

In this experiment, we evaluate the recall of the crowd-selection algorithms using Top1 and Top2.

7.3 Performance Results

We report and discuss the performance results of our TDPM, with VSM, TSPM and DRM, for each of the three datasets as follows.

7.3.1 Performance on Quora

For Quora dataset, we first extract the group of workers based on their participation in solving tasks. We denote the group of workers who solve more than n tasks in Quora as $Quora_n$. For example, $Quora_3$ is a group of workers who solve more than three tasks in Quora. $Quora_1$ consists of all the workers in Quora. In this experiment, we extract nine groups of workers for testing, denoted as $Quora_1, Quora_2, \ldots, Quora_9$.

To analyze the extracted groups, we define task coverage of a group to be the ratio of the number of distinct task solved to the total number of tasks. We illustrate task coverage of the groups by varying the task participation threshold in Figure 3(a). We also show the size of the groups by varying the task participation threshold in Figure 3(b). We can see that the task coverage of $Quora_5$ is above 0.92 while the number of workers in $Quora_5$ is around 30,000+ (only one third of the total workers). We can conclude the size of the group with high task participation threshold is small and the crowd-selection from these groups can achieve high task coverage.

Then, we test the performance of the crowd-selection algorithms on different groups. For fairness, we randomly choose 10k questions for each group where the right worker for each testing question must be in the group.

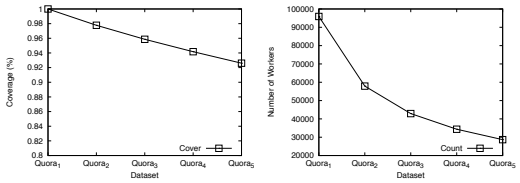
The running time of the algorithms in Quora for Top1 and Top2 crowd-selection are illustrated in Figures 4(a) and 4(b). We can see that the running time of all the algorithms increase with respect

Table 3: Precision of Crowd-Selection Algorithms in Quora (The Best Score in Bold)

Algorithm	$Quora_1$					$Quora_5$					$Quora_9$				
/Category	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50
VSM	0.859					0.873					0.881				
TSPM	0.935	0.936	0.936	0.935	0.935	0.945	0.946	0.947	0.946	0.946	0.953	0.953	0.952	0.953	0.953
DRM	0.936	0.935	0.936	0.936	0.935	0.945	0.946	0.945	0.946	0.948	0.952	0.952	0.954	0.952	0.953
TDPM	**0.945**	**0.948**	**0.950**	**0.951**	**0.951**	**0.957**	**0.959**	**0.961**	**0.961**	**0.962**	**0.962**	**0.965**	**0.966**	**0.966**	**0.966**

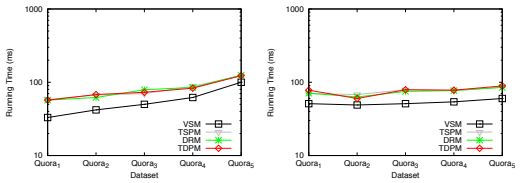
Table 4: Recall of Crowd-Selection Algorithms in Quora (The Best Score in Bold)

Algorithm	$Quora_1$		$Quora_2$		$Quora_3$		$Quora_4$		$Quora_5$	
/TopK	1	2	1	2	1	2	1	2	1	2
VSM	0.733	0.887	0.737	0.891	0.74	0.894	0.743	0.897	0.745	0.899
TSPM	0.882	0.957	0.866	0.939	0.848	0.918	0.831	0.9	0.814	0.882
DRM	0.882	0.956	0.866	0.937	0.844	0.916	0.829	0.9	0.815	0.883
TDPM	**0.8906**	**0.963**	**0.877**	**0.944**	**0.868**	**0.928**	**0.852**	**0.912**	**0.852**	**0.912**



(a) Task Coverage (b) Group Size

Figure 3: Statistics of the Crowd in Quora



(a) Top1 Crowd-Selection (b) Top2 Crowd-Selection

Figure 4: Running Time of Crowd-Selection Algorithms in Quora

to the groups of workers who participate more. This is because the questions answered by active workers are usually popular and attract more workers. Thus, the running cost for selecting right workers increases. We also observe that the running time increases slowly for DRM, TSPM and our TDPM since the estimation of latent category of the given task is the main computation cost.

Now, we investigate the effectiveness of the crowd-selection algorithms. For the Quora dataset, we set the number of latent task category $k = 10$, 20, 30, 40, and 50, respectively. We demonstrate the precision of the crowd-selection algorithms in Table 5 and the recall in Table 6.

We show the precision of the crowd-selection algorithm on three groups $Quora_1$, $Quora_5$ and $Quora_9$ by varying the number of latent categories from 10 to 50 in Table 5. We can observe that the precision of our algorithm is more superior to the other three algorithms on all the number of latent categories set. We can also find that the precision of all the algorithms increases when we select the crowd from more active workers (i.e. we select the workers from the group $Quora_n$). Then, we conclude that the active workers are usually the providers of the best answers. For all the algorithms based on the latent category, we notice that the precision increases and then becomes convergent when we add the number of latent categories.

We illustrate the recall of the algorithms for Top1 and Top2 Crowd-Selection on five groups in Table 6. We can see that both Top1 and Top2 recall of our algorithm is superior than other algorithms for all groups tested. We can also notice that the TopK recall of all the algorithms decreases with respect to the groups of workers who participate more. As mentioned, the questions answered by active workers are popular and the number of workers increases. Thus, the Topk crowd-selection becomes uncertain and the TopK recall of all algorithms decreases in Table 6.

7.3.2 *Performance on Yahoo ! Answer*

For Yahoo ! Answer dataset, we extract five groups of workers for testing, denoted as $Yahoo_{10}$, $Yahoo_{15}$, ..., $Yahoo_{30}$.

We illustrate task coverage of the groups by varying the task participation threshold in Figure 5(a). We also demonstrate the size of the groups by varying the task participation threshold in Figure 5(b). We can see that the task coverage of $Yahoo_{30}$ is around 0.93 while the number of workers in $Yahoo_{30}$ is around 100k (only one tenth of the total workers).

We also randomly choose 10k questions for testing. The running time of the algorithms in Yahoo ! Answer for Top1 and Top2 crowd-selection are illustrated in Figures 6(a) and 6(b). We can see that the running time of all algorithms increase for the questions answered by more active workers.

Now, we validate the precision of the crowd-selection algorithms in Table 5 and the recall in Table 6. We report the precision of the crowd-selection algorithms on $Yahoo_{10}$, $Yahoo_{15}$ and $Yahoo_{20}$ with respect to the number of latent categories. Compared with Quora, we find the precision of the crowd-selection algorithms on Yahoo ! Answer converges faster on the number of latent categories. The precision of VSM is much lower in Table 5. This is because the questions in Yahoo ! Answer are very short compared with the questions in Quora. We show the Top1 and Top2 recall of the crowd-selection algorithms on $Yahoo_{10}$, $Yahoo_{15}$, ..., $Yahoo_{30}$ in Table 6. Both Top1 and Top2 recall of the crowd-selection algorithm decrease for the questions selected for more active workers.

7.3.3 *Performance on Stack Overflow*

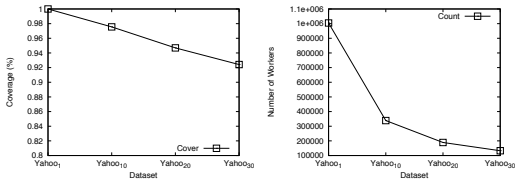
For Stack Overflow dataset, we extract five groups of workers for testing denoted as $Stack_1$, $Stack_3$, ..., $Stack_{12}$.

Table 5: Precision of Crowd-Selection Algorithms in Yahoo ! Answer (The Best Score in Bold)

Algorithm	$Yahoo_{10}$					$Yahoo_{15}$					$Yahoo_{20}$				
/Category	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50
VSM	0.665					0.676					0.685				
TSPM	0.855	0.849	0.848	0.847	0.847	0.884	0.879	0.878	0.877	0.877	0.905	0.900	0.899	0.899	0.899
DRM	0.846	0.847	0.847	0.847	0.847	0.877	0.878	0.877	0.877	0.877	0.898	0.900	0.8992	0.900	0.898
TDPM	**0.945**	**0.947**	**0.948**	**0.949**	**0.949**	**0.965**	**0.969**	**0.970**	**0.969**	**0.970**	**0.981**	**0.984**	**0.984**	**0.984**	**0.985**

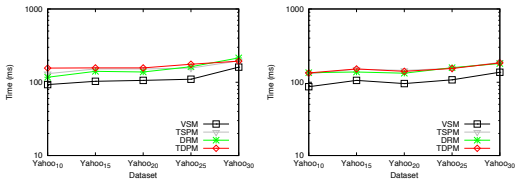
Table 6: Recall of Crowd-Selection Algorithms in Yahoo ! Answer (The Best Score in Bold)

Algorithm	$Yahoo_{10}$		$Yahoo_{15}$		$Yahoo_{20}$		$Yahoo_{25}$		$Yahoo_{30}$	
/TopK	1	2	1	2	1	2	1	2	1	2
VSM	0.518	0.721	0.515	0.717	0.511	0.708	0.504	0.702	0.496	0.693
TSPM	0.695	0.833	0.655	0.810	0.644	0.805	0.637	0.802	0.626	0.795
DRM	0.655	0.823	0.636	0.815	0.629	0.811	0.628	0.809	0.626	0.809
TDPM	**0.823**	**0.908**	**0.815**	**0.904**	**0.811**	**0.903**	**0.809**	**0.901**	**0.809**	**0.901**



(a) Task Coverage (b) Group Size

Figure 5: Statistics of the Crowd in Yahoo ! Answer



(a) Top1 Crowd-Selection (b) Top2 Crowd-Selection

Figure 6: Running Time of Crowd-Selection Algorithms in Yahoo ! Answer

We show task coverage of the groups by varying the task participation threshold in Figure 7(a). We illustrate the number of workers in the groups by varying the task participation threshold in Figure 7(b). We can find that the task cover of $Stack_{12}$ is around 0.9 while the number of workers in $Stack_{12}$ is less than 5k (only one sixth of the total workers).

We randomly choose 1k questions for testing. The running time of the algorithms in Stack Overflow for Top1 and Top2 crowd-selection are illustrated in Figures 8(a) and 8(b). We can see that the running time of all algorithm increase more rapidly than the running time in Quora and Yahoo ! Answer. This is because that there are more workers providing answers for popular questions in Stack Overflow.

We now show the precision of the crowd-selection algorithms in Table 7. We observe that the precision of crowd-selection algorithms increases quickly when we select active workers. We conclude that this is because the users in Stack Overflow usually trust the workers with high reputation such that the active workers are more likely to get best score. We also find that the precision of VSM algorithm is competitive in Table 7. This is because that we use the wisely labeled tags of the questions for vocabularies of the tasks. We illustrate the Top1 and Top2 recall of the crowd-selection algorithms in Table 8. We can notice that the Top1 and Top2 recall of crowd-selection algorithms in Stack Overflow is lower than the recall in other two datasets when we select the workers for more popular questions. We argue that this is because the popular questions in Stack Overflow attract more workers to solve them.

7.3.4 Conclusion on Performance Comparison

In conclusion, the results in Sections 7.3.1, 7.3.2, and 7.3.3 show that TDPM consistently attains high crowd-selection quality in terms of both precision and recall. Compared with the state-of-the-art algorithms, our TDPM algorithm benefits from two aspects new worker skill model and task feedback score. The new worker skill model makes the skills of workers on latent task categories become comparable. We also utilize the task feedback score which can be widely collected in many crowdsourcing platforms to further improve the estimation of the skills of workers on latent categories. In this experimental study, we also find that the crowd-selection from active workers also can greatly improve the precision.

8. CONCLUSION

We studied the problem of task-driven crowd-selection for crowdsourced tasks. Unlike the existing works based on the trustworthiness, our work is to devise a bayesian model that exploits "who knows what" for the workers in crowdsourcing system. The model builds the latent category space of the crowdsourced tasks as well as infers the latent skills of workers on the space. The probabilistic inference for the proposed bayesian model is based on the feedback scores of the past resolved tasks. We then develop a variational algorithm that transforms the probabilistic inference into a standard optimization problem, which can be solved efficiently. We also devise an incremental crowd-selection algorithm that projects the coming tasks into the existing latent category space and choose the highly skilled workers for the tasks. We validate the performance of our algorithm based on the data collected from three well-known crowdsourcing applications: Quora, Yahoo ! Answer and Stack Overflow.

ACKNOWLEDGEMENTS This work is partially supported by GRF under grant number HKUST 617610.

9. REFERENCES

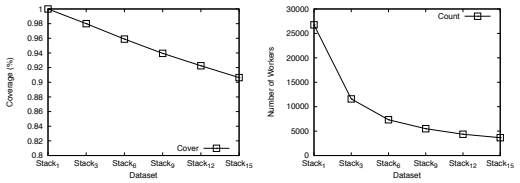
[1] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft. Building reputation in stackoverflow: an

Table 7: Precision of Crowd-Selection Algorithms in Stack Overflow (The Best Score in Bold)

Algorithm	$Stack_1$					$Stack_6$					$Stack_{12}$				
/Category	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50
VSM	0.693					0.738					0.758				
TSPM	0.711	0.706	0.704	0.702	0.702	0.851	0.847	0.849	0.849	0.847	0.935	0.932	0.932	0.931	0.931
DRM	0.710	0.708	0.706	0.708	0.707	0.854	0.850	0.852	0.850	0.849	0.936	0.934	0.933	0.931	0.932
TDPM	**0.801**	**0.813**	**0.817**	**0.821**	**0.824**	**0.930**	**0.941**	**0.946**	**0.950**	**0.954**	**0.997**	**1**	**1**	**1**	**1**

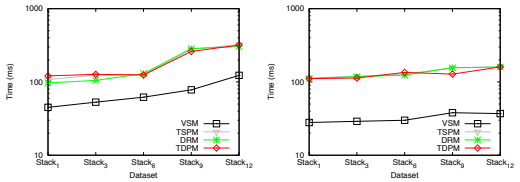
Table 8: Recall of Crowd-Selection Algorithms in Yahoo ! Answer (The Best Score in Bold)

Algorithm	$Stack_1$		$Stack_3$		$Stack_6$		$Stack_9$		$Stack_{12}$	
/TopK	1	2	1	2	1	2	1	2	1	2
VSM	0.433	0.685	0.465	0.713	0.483	0.727	0.493	0.739	0.502	0.749
TSPM	0.461	0.69	0.481	0.713	0.492	0.711	0.488	0.693	0.485	0.684
DRM	0.455	0.685	0.479	0.704	0.483	0.700	0.482	0.694	0.480	0.677
TDPM	**0.581**	**0.819**	**0.634**	**0.835**	**0.651**	**0.826**	**0.651**	**0.806**	**0.64**	**0.782**



(a) Task Coverage (b) Group Size

Figure 7: Statistics of the Crowd in Stack Overflow



(a) Top1 Crowd-Selection (b) Top2 Crowd-Selection

Figure 8: Running Time of Crowd-Selection Algorithms in S-tack Overflow

empirical investigation. In *Proceedings of the Tenth International Workshop on Mining Software Repositories*, pages 89–92. IEEE Press, 2013.

[2] C. C. Cao, J. She, Y. Tong, and L. Chen. Whom to ask?: jury selection for decision making tasks on micro-blog services. *Proceedings of PVLDB*, 5(11):1495–1506, 2012.

[3] C. C. Cao, Y. Tong, L. Chen, and H. Jagadish. Wisemarket: a new paradigm for managing wisdom of online social users. In *Proceedings of SIGKDD*, pages 455–463. ACM, 2013.

[4] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries. In *Proceedings of ICDT*, pages 225–236. ACM, 2013.

[5] G. Dror, Y. Koren, Y. Maarek, and I. Szpektor. I want to answer; who has a question?: Yahoo! answers recommender system. In *Proceedings of SIGKDD*, pages 1109–1117. ACM, 2011.

[6] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of SIGMOD*, pages 61–72. ACM, 2011.

[7] J. Gao, X. Liu, B. C. Ooi, H. Wang, and G. Chen. An online cost sensitive decision-making method in crowdsourcing systems. In *SIGMOD*, 2013.

[8] J. Guo, S. Xu, S. Bao, and Y. Yu. Tapping on the potential of q&a community by recommending answer providers. In *CIKM*, pages 921–930. ACM, 2008.

[9] S. Guo, A. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *Proceedings of SIGMOD*, pages 385–396. ACM, 2012.

[10] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57. ACM, 1999.

[11] H. Kaplan, I. Lotosh, T. Milo, and S. Novgorodov. Answering planning queries with the crowd.

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

[13] A. Marcus, D. Karger, S. Madden, R. Miller, and S. Oh. Counting with the crowd. In *VLDB*, pages 109–120. VLDB Endowment, 2012.

[14] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller. Human-powered sorts and joins. *Proceedings of PVLDB*, 5(1):13–24, 2011.

[15] A. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. *Proceedings of PVLDB*, 4(5):267–278, 2011.

[16] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: Algorithms for filtering data with humans. In *Proceedings of SIGMOD*, pages 361–372. ACM, 2012.

[17] H. Park and J. Widom. Query optimization over crowdsourced data, 2012.

[18] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *JMRL*, 99:1297–1322, 2010.

[19] Y. Tian and J. Zhu. Learning from crowds in the presence of schools of thought. In *Proceedings of SIGKDD*, pages 226–234. ACM, 2012.

[20] B. Trushkowsky, T. Kraska, M. J. Franklin, and P. Sarkar. Crowdsourced enumeration queries. In *ICDE*, 2013.

[21] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis. Max algorithms in crowdsourcing environments. In *Proceedings of WWW*, pages 989–998. ACM, 2012.

[22] M. J. Wainwright and M. I. Jordan. Graphical models,

exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[23] G. Wang, K. Gill, M. Mohanlal, H. Zheng, and B. Y. Zhao. Wisdom in the social crowd: an analysis of quora. In *Proceedings of WWW*, pages 1341–1352. International World Wide Web Conferences Steering Committee, 2013.

[24] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of PVLDB*, 5(11):1483–1494, 2012.

[25] P. Welinder and P. Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *CVPRW*, pages 25–32. IEEE, 2010.

[26] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. 2013.

[27] S. Wu, Z. Zhang, A. K. Tung, X. Wang, and S. Wang. K-anonymity for crowdsourcing database. *IEEE Transactions on Knowledge and Data Engineering*, page 1, 2013.

[28] F. Xu, Z. Ji, and B. Wang. Dual role model for question recommendation in community question answering. In *Proceedings of SIGIR*, pages 771–780. ACM, 2012.

[29] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 77–90. ACM, 2010.

[30] C. J. Zhang, L. Chen, H. Jagadish, and C. C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *VLDB*, 6(9), 2013.

[31] Z. Zhao, W. Ng, and Z. Zhang. Crowdseed: query processing on microblogs. In *Proceedings of EDBT*, pages 729–732. ACM, 2013.

[32] Z. Zhao, D. Yan, W. Ng, and S. Gao. A transfer learning based framework of crowd-selection on twitter. In *SIGKDD*, pages 1514–1517. ACM, 2013.

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

10. APPENDIX

10.1 Variational Parameter Estimation

We derive the partial derivatives one by one, starting with $\frac{\partial \mathcal{L}'}{\partial \lambda_w}$. For simplicity, we collect the terms involving λ_w in $\mathcal{L}'$ as

$$\begin{aligned}\mathcal{L}'_{\lambda_w^i} &= E_q[\log p(W|\mu_w, diag(\nu_w^2))] + E_q[\log p(S|WC^T,\tau)] \\ &- E_q[\log p(W|\lambda_w, diag(\nu_w^2))] \\ &= \Sigma_w^{-1}\mu_w\lambda_w^i - \frac{1}{2}\Sigma_w^{-1}(\lambda_w^i)^T\lambda_w^i + \sum_{t^j:a_{ij}=1}\{\frac{1}{\tau^2}(\lambda_w^i)^T\lambda_c^j s_{ij} \\ &+ (\lambda_c^j)^T\lambda_w^i(\lambda_w^i)^T\lambda_c^j] - \frac{1}{2\tau^2}[(\lambda_w^i)^T\lambda_w^i diag((\nu_w^j)^2)\}.\end{aligned}$$

Then, we set its derivative to zero and we obtain

$$\begin{aligned}\lambda_w^i &= (\Sigma_w^{-1} + \frac{1}{\tau^2}\sum_{t^j:a_{ij}=1}((\lambda_c^j)^T\lambda_c^j + diag((\nu_c^j)^2)))^{-1} \\ &\times (\Sigma_w^{-1}\mu_w + \frac{1}{\tau^2}\sum_{t^j\in a_{ij}=1} s_{ij}\lambda_c^j).\end{aligned}$$

For ν_w^i, we have

$$\begin{aligned}\mathcal{L}_{\nu_{w,k}^i} &= E_q[\log p(W|\mu_w, diag(\nu_w^2))] + E_q[\log p(S|WC^T,\tau)] \\ &- E_q[\log p(W|\lambda_w, diag(\nu_w^2))] \\ &= -\frac{1}{2}\Sigma_w^{-1}diag((\nu_w^i)^2) - \sum_{t^j:a_{ij}=1}\{\frac{1}{2\tau^2}((\lambda_c^j)^T\lambda_c^j diag((\nu_w^i)^2) \\ &+ diag((\nu_w^i)^2)diag((\nu_c^j)^2)) - \frac{1}{2}\ln|diag((\nu_w^i)^2)|\}\end{aligned}$$

By setting $\frac{\partial \mathcal{L}'}{\partial \nu_{w,k}^i} = 0$, we have

$$(\nu_{w,k}^i)^2 = (\sum_{t^j:a_{ij}}\frac{(\lambda_{c,k}^j)^2 + (\nu_{c,k}^j)^2}{\tau^2} + \Sigma_{w,kk}^{-1})^{-1}.$$

For $\phi_{c,v}$, we have

$$\begin{aligned}\mathcal{L}'_{\phi_{c,v}} &= E_q[\log p(Z|C)] + E_q[\log p(V|Z,\beta)] - E_q[\log p(Z|\phi)]] \\ &= \lambda_c\phi + \phi\log\beta_{c,v} - \phi\log\phi.\end{aligned}$$

By setting $\frac{\partial \mathcal{L}'}{\partial \phi_{c,v}} = 0$, we have

$$\phi_{c,v} \propto \exp(\lambda_{c,v}^j + \frac{1}{L}\sum_{j=1}^{T}\sum_{p=1}^{L}1[v_p^j = v]\log\beta_{c,v}).$$

For ε, we have

$$\mathcal{L}'_\varepsilon = -\varepsilon^{-1}(\sum_{k=1}^{K}E_q[\exp\{C_k\}]) - \log\varepsilon.$$

By setting $\frac{\partial \mathcal{L}'}{\partial \varepsilon} = 0$, we have

$$\varepsilon_j = \sum_{k=1}^{K}\exp(\lambda_{c,k}^j + \frac{(\nu_{c,k}^j)^2}{2})$$

10.2 Model Parameter Estimation

We estimate the model parameters based on the derived variational parameters.

We first maximize $\mathcal{L}'(q)$ with respect to μ_w, given by

$$\mathcal{L}'_{\mu_w} = \sum_{i=1}^{M}\Sigma_w^{-1}\mu_w\lambda_w^i - M\frac{1}{2}\mu_w^T\Sigma_w^{-1}\mu_w$$

Then, we set its derivative to zero and we obtain

$$\mu_w = \frac{1}{M}\sum_{i=1}^{M}\lambda_w^i.$$

For Σ_w, we have

$$\begin{aligned}\mathcal{L}'_{\Sigma_w} &= \sum_{i=1}^{M}(\Sigma_w^{-1}\mu_w\lambda_w^i - \frac{1}{2}\Sigma_w^{-1}(diag((\nu_w^i)^2) + (\lambda_w^i)^T\lambda_w^i) \\ &- M(\frac{1}{2}\mu_w^T\Sigma_w^{-1}\mu_w + \frac{1}{2}\ln|\Sigma_w|)).\end{aligned}$$

Next, we set its derivative to zero, given by

$$\begin{aligned}\frac{\partial \mathcal{L}'}{\partial \Sigma_w} &= \sum_{i=1}^{M}(-\Sigma_w^{-T}\Sigma_w^{-T}\mu_w\lambda_w^i + \frac{1}{2}\Sigma_w^{-T}\Sigma_w^{-T}(diag((\nu_w^i)^2) + \lambda_w^i(\lambda_w^i)^T)) \\ &- M(-\frac{1}{2}\Sigma_w^T\mu_w\mu_w^T\Sigma_w^{-T} + \frac{1}{2}(\Sigma_w^T)^{-1}).\end{aligned}$$

By setting $\frac{\partial \mathcal{L}'}{\partial \Sigma_w} = 0$, we have

$$\Sigma_w = \frac{1}{M}\sum_{i=1}^{M}(diag((\nu_w^i)^2) + (\lambda_w - \mu_w)(\lambda_w - \mu_w)^T).$$

The derivations for μ_c and Σ_c are similar to the above.