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 crowdsourcing 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 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^i 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...
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-ups 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 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.

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

In this section, we present a bayesian model to infer the latent skills of 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 skill distributions 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.

The crowdsourced task \( t_j \) is represented as a bag of vocabularies \( v_{ij} \), where \( \#v_{ij} \) is the number of vocabulary \( v_j \) in task \( t_j \). We assume that the feedback scores for the workers are proportional to their skills (i.e., thumbs-up for their returned answers in Quora. We assume that the feedback scores of worker \( w_j \) can be interpreted as \( w_{ij}^1 \times c_1 + w_{ij}^2 \times c_2 = 4.7 \times 0.9 + 1.1 \times 0.1 \approx 4 \).
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 $w^i$ 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 $0 \leq s_{ij} \leq 1$ 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

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

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 $T = \{t_1, t_2, \ldots, t_M\}$ and $M$ workers $W = \{w^1, w^2, \ldots, w^M\}$.

Now, we denote a set of model parameters $\varphi = \{W, \Sigma_w, C, \Sigma_c, \tau, \beta\}$. The parameters $W$ and $\Sigma_w$ are the prior for latent worker skill while $C$ and $\Sigma_c$ are the prior for latent task category. The parameter $\tau$ is variance between the feedback score and the predictive performance of the workers. The parameter $\beta$ 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

$$w^i \sim \text{Normal}(\mu_w, \Sigma_w)$$

$$\sim \frac{1}{(2\pi)^{N/2} |\Sigma_w|^{1/2}} \exp\left(-\frac{1}{2} (w^i - \mu_w)^T \Sigma_w^{-1} (w^i - \mu_w)\right)$$

where $\mu_w$ is the mean of the skills on latent categories and $\Sigma_w$ 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, $\Sigma_w$ 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

$$c^j \sim \text{Normal}(\mu_c, \Sigma_c)$$

$$\sim \frac{1}{(2\pi)^{K/2} |\Sigma_c|^{1/2}} \exp\left(-\frac{1}{2} (c^j - \mu_c)^T \Sigma_c^{-1} (c^j - \mu_c)\right)$$

where $\mu_c$ is the latent category distribution for all the crowdsourced tasks. The parameter $\Sigma_c$ 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 $v^j_p$ in task $t_j$, we assume $z^j_p$ is generated from a discrete distribution, given by

$$z^j_p \sim \text{Discrete} (\text{logistic}(c^j))$$

$$\sim \frac{\exp(c^j_p)}{\sum_{k=1}^{K} \exp(c^j_k)}$$

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

Based on the latent category $z^j_p$, the vocabulary is generated from a discrete distribution, given by

$$v^j_p \sim \beta_{z^j_p} = p(v^j_p \mid \beta, z^j_p)$$

where $\beta$ is a vocabulary distribution over latent categories. The parameter $\beta$ 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

$$s_{ij} \sim \text{Normal}(w^i(c^j)^T, \tau)$$

$$\sim \frac{1}{\tau \sqrt{2\pi}} \exp\left(-\frac{(s_{ij} - w^i(c^j)^T)^2}{2\tau^2}\right)$$

where the parameter $\tau$ 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 $\mu_w$ and $\Sigma_w$ from Line 1 to Line 3. Next, Algorithm 1 generates latent categories for the crowdsourced tasks by Normal distribution with parameters $\mu_c$ and $\Sigma_c$ in Line 3. The latent category of the vocabulary $z^j_p$ 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 $\beta$ and the latent category for vocabularies $z^j_p$ 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 $\varphi \equiv \{W, \Sigma_w, C, \Sigma_c, \tau, \beta_i\}$

Output: Feedback scores $S$

1: for each worker $w^i \in W$ do
2: \hspace{1em} Choose skills $w^i$ by Equation 2.
3: \hspace{1em} end for
4: for each task $t_j \in T$ do
5: \hspace{1em} Choose latent task category $c^j$ by Equation 3
6: \hspace{1em} for each vocabulary $v^j_{p^j} \in t_j$ do
7: \hspace{2em} (a) Choose a latent category $z^j_{p^j}$ by Equation 4
8: \hspace{1em} (b) Choose the text for vocabulary $v^j_{p^j}$ by Equation 5
9: \hspace{1em} end for
10: end for
11: for each crowdsourced task $t^i \in T$ do
12: \hspace{1em} for each employed worker $w^i \in A_i$ do
13: \hspace{2em} Generate feedback score $s_{ij}$ by Equation 6
14: \hspace{1em} end for
15: \hspace{1em} end for
16: 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 $\varphi = \{\mu_w, \Sigma_w, \mu_c, \Sigma_c, \tau, \beta_i\}$, and task assignment $A$, we factorize the joint distribution over $V$ and $S$, given by

$$p(V, S | A, \varphi) = \int_C \int_W p(W | \mu_w, \Sigma_w) p(C | \mu_c, \Sigma_c) p(Z | C, \tau) dC dW$$

where

$$p(W | \mu_w, \Sigma_w) = \prod_{w^i \in W} p(w^i | \mu_w, \Sigma_w)$$

$$p(C | \mu_c, \Sigma_c) = \prod_{c^j \in C} p(c^j | \mu_c, \Sigma_c)$$

$$p(Z | C) = \prod_{t_j \in T} \prod_{c^j \in C} \text{discrete}(\text{logistic}(c^j))$$

$$p(V | Z) = \prod_{t_j \in T} \prod_{s_{ij} \in 1} \beta_{s^j_{p^j}}$$

$$p(S | WC^T, \tau) = \prod_{t_j \in T} \prod_{s^j_{p^j} \in 1} p(s_{ij} | w^i (c^j)^T, \tau)$$

and $p(w^i | \mu_w, \Sigma_w)$, $p(c^j | \mu_c, \Sigma_c)$, $\text{discrete}(\text{logistic}(c^j))$, $\beta_{s^j_{p^j}}$, 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

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

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 $C$,

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

where

$$p(W, C, Z, \varphi | V, S) = \frac{p(W, C, Z, \varphi, V, S) dW dC dZ d\varphi}{\int_{W,C,Z,\varphi} p(W, C, Z, \varphi, V, S) dW dC dZ d\varphi}$$

This distribution is intractable to compute due to the coupling between the model parameters in $\varphi$. 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:

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

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

$$q(W, C, Z | \lambda_w, \nu_w, \lambda_c, \nu_c, \phi)$$

$$= \prod_{i=1}^{M} q(w^i | \lambda_w, \text{diag}(\nu_w^2)) \prod_{j=1}^{N} (q(c^j | \lambda_c, \text{diag}(\nu_c^2)))$$

$$\times \prod_{p=1}^{L} q(s^j_{p^j} | \phi_{p^j}),$$

where

$$q(w^i | \lambda_w, \text{diag}(\nu_w^2)) = \text{Normal}(\lambda_w, \text{diag}(\nu_w^2)),$$

$$q(c^j | \lambda_c, \text{diag}(\nu_c^2)) = \text{Normal}(\lambda_c, \text{diag}(\nu_c^2)),$$

$$q(s^j_{p^j} | \phi_{p^j}) = \text{discrete}(\phi_{p^j}).$$

diag(\cdot) is a diagonal matrix where the entries outside the main diagonal are all zero. Here $\lambda_w, \text{diag}(\nu_w^2), \lambda_c, \text{diag}(\nu_c^2), \phi_{p^j}$ are variational parameters. For brevity, we denote the collection of variational parameters as $\varphi' = \{\lambda_w, \text{diag}(\nu_w^2), \lambda_c, \text{diag}(\nu_c^2), \phi\}$
in the rest of the paper. Thus, the inference for \(W, C, \) and \(Z\) in Equation 7 can be simplified as follows:

\[
(W^*, C^*, Z^*) = \arg \max_{W^*, C^*, Z^*} \arg \max_{w^1} q(w^1), \ldots, \arg \max_{w^N} q(w^M), \\
\arg \max_{c^1} q(c^1), \ldots, \arg \max_{c^N} q(c^N), \\
\arg \max_{z^1} q(z^1), \ldots, \arg \max_{z^N} q(z^N)
\]

\[
= \arg \max_{\lambda^*, \alpha^*, \beta^*} q(\alpha^*)q(\beta^*)q(\lambda^*)q(\nu^*), \\
\arg \max_{\lambda^*} q(\lambda^*), \ldots, \arg \max_{\lambda^*} q(\lambda^*) q(\nu^*), \\
\arg \max_{z^*} q(z^*) \ldots, \arg \max_{z^*} q(z^*)
\]

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 \(\varphi\) with respect to \(L(q)\). We take the derivative of the lower bound \(L(q)\) with respect to the variational parameters \(\lambda, \nu, \phi, \varphi\), and set these derivatives to zeros.

\[
\nabla_\varphi L(q) = \left( \frac{\partial L(q)}{\partial \varphi_1}, \ldots, \frac{\partial L(q)}{\partial \varphi_N} \right) = 0.
\]

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

\[
\lambda^*_w = \frac{1}{\tau} \sum_{t_j, a_{ij} = 1} (\lambda^*_w \nu^* + \text{diag}(\nu^*)), \quad \nu^* = \frac{1}{L} \sum_{j = 1}^{L} \sum_{i = 1}^{N} \log \beta_{s_{ij}}, \quad \tau = \sum_{i = 1}^{N} \exp(\lambda^*_w),
\]

\[
\epsilon_j = \exp(\lambda^*_w) \frac{(\nu^{*})^2}{2},
\]

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

However, \(L(q)\) is not amenable to analytic maximization with respect to \(\lambda^*\) and \(\nu^*\). Thus, we use the conjugate gradient algorithm with derivative

\[
\frac{\partial L(q)}{\partial \lambda^*} = (\lambda^*_w - 1) + \frac{1}{\tau} \sum_{t_j, a_{ij} = 1} (\lambda^*_w \nu^* + \text{diag}(\nu^*))
\]

\[+ \sum_{t_j, a_{ij} = 1} s_{ij} \lambda^*_w + L \phi^* - \frac{1}{\epsilon} \exp(\lambda^*_w) \frac{(\nu^{*})^2}{2}.
\]

\[
\frac{\partial L(q)}{\partial \nu^{*}} = (\lambda^*_w - 1) + \frac{1}{\tau} \sum_{t_j, a_{ij} = 1} (\lambda^*_w \nu^* + \text{diag}(\nu^*))
\]

\[+ \sum_{t_j, a_{ij} = 1} s_{ij} \lambda^*_w + L \phi^* - \frac{1}{\epsilon} \exp(\lambda^*_w) \frac{(\nu^{*})^2}{2}.
\]

5.3 Optimization Procedure

Based on the estimated stationary points, we optimize the model parameters \(\varphi = \{\mu_w, \Sigma_w, \mu_e, \Sigma_e, \tau, \beta_{s_{ij}}\}\) in this section.

We take the derivative of the lower bound \(L(q)\) with respect to the model parameters \(\varphi\), and set these derivatives to zeros.

\[
\nabla_\varphi L(q) = \left( \frac{\partial L(q)}{\partial \varphi_1}, \ldots, \frac{\partial L(q)}{\partial \varphi_N} \right) = 0.
\]

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 \( n_{\text{max}} \).

**Output:** variational parameters \( \varphi' \) and model parameters \( \varphi \).

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

\[
\mu_w = \frac{1}{M} \sum_{t=1}^{M} \lambda_w^n \tag{16}
\]
\[
\Sigma_w = \frac{1}{M} \sum_{t=1}^{M} \text{diag}(\nu_v^n)^2 + (\lambda_w - \mu_w)(\lambda_w - \mu_w)^T \tag{17}
\]
\[
\mu_c = \frac{1}{N} \sum_{j=1}^{N} \lambda_c^j \tag{18}
\]
\[
\Sigma_c = \frac{1}{N} \sum_{j=1}^{N} \text{diag}(\nu_v^n)^2 + (\lambda_c - \mu_c)(\lambda_c - \mu_c)^T \tag{19}
\]
\[
\tau^2 = \frac{1}{|A|} \sum_{a_{ij}=1} s_{ij} (\lambda_w^a)^2 \text{diag}(\nu_v^n)^2 \lambda_w^a + (\lambda_c^a)^2 \text{diag}(\nu_v^n)^2 \lambda_c^a - 2s_{ij} (\lambda_w^a)^T \lambda_c^a + (\text{diag}(\nu_v^n)^2)^2 \text{diag}(\nu_v^n)^2) \tag{20}
\]
\[
\beta_k,v \propto \sum_{j=1}^{T} \sum_{p=1}^{L} \phi_{p,k}^j (v_p^j = v) \tag{21}
\]

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 \( \varphi' \) and model parameters \( \varphi \) in Lines 3 and 4 until the objective function becomes convergent. The objective function \( L(q) \) can be persistently improved by variational algorithm, stated in [22]. Because the value of \( L(q) \) 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 \( \lambda_c^j \) and \( \nu_v^j \) 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 \( c \) and feedback scores are removed. Thus, we compute its latent category using conjugate gradient algorithm with derivative

\[
\frac{\partial L(q)}{\partial \lambda_c^j} = \Sigma_c^{-1} \lambda_c^j + \mu_c + L \phi^j - \frac{1}{\varepsilon_j} \exp\{\lambda_c^j + \frac{(\nu_v^j)^2}{2}\} \tag{22}
\]
\[
\frac{\partial L(q)}{\partial \nu_v^j} = \Sigma_v^{-1} \nu_v^j - \frac{1}{\varepsilon_j} \exp\{\lambda_c^j + \frac{(\nu_v^j)^2}{2}\} \tag{23}
\]

and the update for \( \phi^j \) and \( \varepsilon_j \) 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 \( \varphi \), Algorithm 3 selects top-\( k \) workers \( R \) for this task. In the first phrase, Algorithm 3 computes the variational parameters \( \lambda_c^j \) and \( \nu_v^j \) 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 \( \lambda_c^j \) and variance \( \nu_v^j \) 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.

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 responder 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\(^4\) through its API from Jan, 2012 onwards. In total, we collect 886,600+ unique questions. On the average, each question has around three respondents.

\(^4\)http://www.quora.com/

\(^5\)http://answers.yahoo.com/
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 8666,000+ 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. 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 as a bag of vocabularies for the given task defined in Section 4.1. Then, we define a bag of vocabularies of the task resolved by worker \( w \). The ranking score of worker \( w \) is given by

\[
s_{ij} = \frac{(t^j)^T t^i}{\sqrt{(t^j)^T t^j} \sqrt{(t^i)^T t^i}}
\]

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

- **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 \( w \) as well as the latent category of the task \( c \). Given a crowdsourced task \( t^i \), the crowd-selection of DRM on workers is proportional to the predictive performance \( w^i(c) \).

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

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. \( w^i_1 > w^i_2 \) ?). 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

\[
ACCU = \frac{|R| - R_{\text{best}} - 1}{|R| - 1}
\]

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

We propose to use Top\( K \) to measure the recall of the crowd-selection algorithm. The Top\( K \) 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 Top\( K \) is given by

\[
\text{Top}K = \frac{|\{t^j | R_{\text{best}} | K\}|}{N}
\]

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\(_1\) is a group of workers who solve more than three tasks in Quora. Quora\(_2\) consists of all the workers in Quora. In this experiment, we extract nine groups of workers for testing, denoted as Quora\(_1\), Quora\(_2\), ..., 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 from 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)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Quora1</th>
<th>Quora5</th>
<th>Quora1</th>
<th>Quora5</th>
<th>Quora1</th>
<th>Quora5</th>
<th>Quora1</th>
<th>Quora5</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSM</td>
<td>0.859</td>
<td></td>
<td>0.873</td>
<td></td>
<td>0.881</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSPM</td>
<td>0.935</td>
<td>0.936</td>
<td>0.936</td>
<td>0.935</td>
<td>0.945</td>
<td>0.946</td>
<td>0.947</td>
<td>0.946</td>
</tr>
<tr>
<td>DRM</td>
<td>0.936</td>
<td>0.935</td>
<td>0.936</td>
<td>0.935</td>
<td>0.945</td>
<td>0.946</td>
<td>0.945</td>
<td>0.948</td>
</tr>
<tr>
<td>TDPM</td>
<td>0.945</td>
<td>0.948</td>
<td>0.950</td>
<td>0.951</td>
<td>0.957</td>
<td>0.959</td>
<td>0.961</td>
<td>0.961</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Quora1</th>
<th>Quora5</th>
<th>Quora1</th>
<th>Quora5</th>
<th>Quora1</th>
<th>Quora5</th>
<th>Quora1</th>
<th>Quora5</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSPM</td>
<td>0.733</td>
<td>0.887</td>
<td>0.737</td>
<td>0.891</td>
<td>0.744</td>
<td>0.894</td>
<td>0.743</td>
<td>0.897</td>
</tr>
<tr>
<td>DRM</td>
<td>0.862</td>
<td>0.957</td>
<td>0.866</td>
<td>0.939</td>
<td>0.848</td>
<td>0.918</td>
<td>0.831</td>
<td>0.831</td>
</tr>
<tr>
<td>TDPM</td>
<td>0.8906</td>
<td>0.963</td>
<td>0.877</td>
<td>0.944</td>
<td>0.868</td>
<td>0.928</td>
<td>0.852</td>
<td>0.912</td>
</tr>
</tbody>
</table>

Figure 3: Statistics of the Crowd in Quora

(a) Task Coverage  (b) Group Size

7.3.2 Performance on Yahoo! Answer

For Yahoo! Answer dataset, we extract five groups of workers for testing, denoted as \(Yahuoe_{10}, Yahuoe_{15}, \ldots, Yahuoe_{20}\).

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 \(Yahuoe_{20}\) is around 0.93 while the number of workers in \(Yahuoe_{20}\) 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.

7.3.3 Performance on Stack Overflow

For Stack Overflow dataset, we extract five groups of workers for testing denoted as \(Stack_{1}, Stack_{3}, \ldots, Stack_{12}\).
Table 5: Precision of Crowd-Selection Algorithms in Yahoo! Answer (The Best Score in Bold)

<table>
<thead>
<tr>
<th>Algorithm /Category</th>
<th>Yahoo10</th>
<th>Yahoo15</th>
<th>Yahoo20</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSM</td>
<td>0.665</td>
<td>0.685</td>
<td></td>
</tr>
<tr>
<td>TSPM</td>
<td>0.855</td>
<td>0.884</td>
<td>0.877</td>
</tr>
<tr>
<td>DRM</td>
<td>0.846</td>
<td>0.877</td>
<td>0.877</td>
</tr>
<tr>
<td>TDPM</td>
<td>0.945</td>
<td>0.970</td>
<td>0.970</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Algorithm /TopK</th>
<th>Yahoo10</th>
<th>Yahoo15</th>
<th>Yahoo20</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSM</td>
<td>0.518</td>
<td>0.717</td>
<td>0.504</td>
</tr>
<tr>
<td>TSPM</td>
<td>0.695</td>
<td>0.810</td>
<td>0.637</td>
</tr>
<tr>
<td>DRM</td>
<td>0.655</td>
<td>0.815</td>
<td>0.628</td>
</tr>
<tr>
<td>TDPM</td>
<td>0.823</td>
<td>0.904</td>
<td>0.809</td>
</tr>
</tbody>
</table>

Figure 5: Statistics of the Crowd 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 Stack12 is around 0.9 while the number of workers in Stack12 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 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 of improved recall in the Yahoo! Answer dataset and the increased number of workers providing answers for popular questions.
Table 7: Precision of Crowd-Selection Algorithms in Stack Overflow (The Best Score in Bold)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Stack1</th>
<th>Stack6</th>
<th>...</th>
<th>Stack11</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSM</td>
<td>0.693</td>
<td>0.738</td>
<td>0.758</td>
<td></td>
</tr>
<tr>
<td>TSPM</td>
<td>0.711</td>
<td>0.706</td>
<td>0.704</td>
<td>0.702</td>
</tr>
<tr>
<td>DRM</td>
<td>0.710</td>
<td>0.708</td>
<td>0.706</td>
<td>0.708</td>
</tr>
<tr>
<td>TDPM</td>
<td>0.801</td>
<td>0.813</td>
<td>0.817</td>
<td>0.821</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Stack1</th>
<th>Stack5</th>
<th>Stack9</th>
<th>Stack12</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSM</td>
<td>0.433</td>
<td>0.685</td>
<td>0.465</td>
<td>0.713</td>
</tr>
<tr>
<td>TSPM</td>
<td>0.461</td>
<td>0.691</td>
<td>0.481</td>
<td>0.713</td>
</tr>
<tr>
<td>DRM</td>
<td>0.455</td>
<td>0.685</td>
<td>0.479</td>
<td>0.704</td>
</tr>
<tr>
<td>TDPM</td>
<td>0.581</td>
<td>0.819</td>
<td>0.634</td>
<td>0.835</td>
</tr>
</tbody>
</table>

Figure 7: Statistics of the Crowd in Stack Overflow

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

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


10. APPENDIX

10.1 Variational Parameter Estimation

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

$$\mathcal{L}'_\nu = E_q[\log p(W|\nu_w, \text{diag}(\nu_w^2))] + E_q[\log p(S|WC^T, \tau)]$$

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

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

Then, we set its derivative to zero and we obtain

$$\nu_w = \left( \left( \lambda_w^T \lambda_w + \text{diag}(\nu_w^2) \right)^{-1} \right) \lambda_w.$$

For $\mu_w$, we have

$$\mathcal{L}'_{\mu_w} = E_q[\log p(W|\mu_w, \text{diag}(\mu_w^2))] + E_q[\log p(S|WC^T, \tau)]$$

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

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

The derivations for $\Sigma_w$ and $\tau_w$ are similar to the above.