Cross-Task Crowdsourcing

Kaixiang Mo[1], Erheng Zhong[1] and Qiang Yang[1,2]
1:Hong Kong University of Science and Technology, Hong Kong;
2: Huawei Noah's Ark Lab, Science and Technology Park, Shatin, Hong Kong
{kxmo,ezhong,qyang}@cse.ust.hk

ABSTRACT

Crowdsourcing is an effective method for collecting labeled data for various data mining tasks. It is critical to ensure the veracity of the produced data because responses collected from different users may be noisy and unreliable. Previous works solve this veracity problem by estimating both the user ability and question difficulty based on the knowledge in each task individually. In this case, each single task needs large amounts of data to provide accurate estimations. However, in practice, budgets provided by customers for a given target task may be limited, and hence each question can be presented to only a few users where each user can answer only a few questions. This data sparsity problem can cause previous approaches to perform poorly due to the overfitting problem on rare data and eventually damage the data veracity. Fortunately, in real-world applications, users can answer questions from multiple historical tasks. For example, one can annotate images as well as label the sentiment of a given title. In this paper, we employ transfer learning, which borrows knowledge from auxiliary historical tasks to improve the data veracity in a given target task. The motivation is that users have stable characteristics across different crowdsourcing tasks and thus data from different tasks can be exploited collectively to estimate users' abilities in the target task. We propose a hierarchical Bayesian model, TLC (Transfer Learning for Crowdsourcing), to implement this idea by considering the overlapping users as a bridge. In addition, to avoid possible negative impact, TLC introduces task-specific factors to model task differences. The experimental results show that TLC significantly improves the accuracy over several state-of-the-art non-transfer-learning approaches under very limited budget in various labeling tasks.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – Data Mining

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

Keywords

Crowdsourcing, Transfer Learning

1. INTRODUCTION

Producing large-scale training, validation and test sets is vital for many machine learning and data mining applications. Most often this task has to be carried out "by hand" and thus it is delicate, expensive, and tedious. Crowdsourcing systems such as Amazon Mechanical Turk [1], reCAPTCHA [2] and the ESP game [3] have made it easy to distribute simple labeling tasks to hundreds of users(referred to as worker in Mechanical Turk). Crowdsourcing is widely used in tasks such as sentiment classification [12], object recognition [7], ranking [13] and clustering [6], etc.

A typical crowdsourcing pipeline can be divided into three main steps: 1: Task design, 2: Data distribution and 3: Answer aggregation. In the answer aggregation step, one needs to aggregate these noisy and unreliable responses into a true answer. Answer aggregation is most important step because it is closely related to the veracity of the produced answers and the final performance of machine learning algorithms.

An intuitive way to aggregate the responses is via "Majority Voting", in which one always selects the answer chosen by most users. However, this method is ineffective in practice. First, some users are more skilled and consistent in one specific task, as they may master more background knowledge, or may be more patient. This implies that when dealing with the same task, different users may have different abilities to annotate instances/answer questions. Second, different instances may have different levels of difficulty. Third, since typical crowdsourcing tasks are tedious and the reward is small, errors are common even among users who made an effort. Thus, in order to achieve higher veracity for answer aggregation, one needs to take both user ability and question difficulty into account.

Several approaches have been proposed based on the above motivation [14, 6, 7, 15, 10]. These works model the user ability and question difficulty as two latent attributes, which affect the probability that one user provides the correct answer to a question. These approaches require the collecting of large amounts of data to guarantee the veracity of the final answers. However, in practice, due to the limited budget of each task provided by customers, each question may be presented to a few users and each user may answer only

[1] `https://www.mturk.com/mturk/welcome`
[2] `http://www.google.com/recaptcha`
[3] `http://www.gwap.com/gwap/gamesPreview/espgame/`

Table 1: Related Works in Transfer Learning and Crowdsourcing

	Gold Truth	Crowdsourced Labels
Single-task	Traditional Machine Learning	Traditional Crowdsourcing [14, 6, 7, 15, 10]
Cross-task	Traditional Transfer Learning [8, 4, 9, 17, 3, 18]	Cross-task Crowdsourcing (CTC)

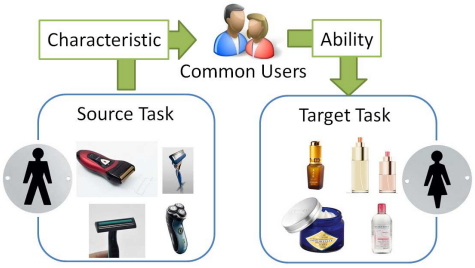


Figure 1: Cross Task Crowdsourcing

a few questions. We define this problem as data sparisity. This data sparseness may make existing learning algorithms overfit the few responses. For example, a highly reliable user who only answers two questions in a new task may accidentally make an obvious mistake in one question. This may cause the other answer from this user to be considered as useless due to overfitting, despite the fact that this user is an expert at doing similar tasks. Under this situation, previous approaches may fail to infer the user ability and question difficulty accurately and hence may provide incorrect aggregated answers.

Fortunately, in most crowdsourcing services, many users have answered questions in multiple historical tasks. According to a survey by Ross [11], 31% Mechanical Turk users have worked for more than half a year and 43% of users work more than 5 hours a week. The survey also stated that people often participate in various types of task such as image labeling and item comparison. In addition, the normal time span of a task is about 2 weeks, so we can infer that active users who have worked for more than half a year have answered questions in many different tasks with high probability. This phenomenon sheds light on solving the data sparsity problem: by considering the same users as a bridge, data from previous tasks can be borrowed to benefit the current task as most users may have relatively stable characteristics in completing related tasks.

However, due to the differences among tasks, merging the data from multiple tasks naively does not work well. For example, different tasks may require different background knowledge, such as the differences between image annotation and sentiment analysis tasks. There are also differences between different kinds of labels, e.g., binary vs. multi-class labels. Finally, we may have different display styles, e.g., texts vs. images. These differences may influence users' performance and misguide the estimation process on user ability and question difficulty. Can we utilize the data from multiple previously labeled tasks for labeling the data in the current task, while avoiding the negative effects stemming from task differences?

In this paper, we consider the problem of making the maximum use of the knowledge gained from previously solved crowdsourcing tasks to benefit the current crowdsourcing task. We propose a novel transfer-learning basd solution known as *Cross-Task Crowdsourcing* (CTC) for the problem. In particular, we exploit transfer learning to extract knowledge from auxiliary domains to help learning in a target domain [8]. As an example, in Figure 1, some new users in MTurk first answer many questions about images on razors and then accept a task to annotate a set of new images related to the topic of 'makeups'. By observing users' performance in the razor-recognition task, users' characteristics, such as gender, can be inferred automatically. In addition, these characteristics are closely related to users' abilities to answer questions in other tasks. For example, female users generally know about makeup products better than male users. Thus, in the target makeup task, users' abilities can be estimated more accurately, which can be used to improve the data veracity of answer aggregation. Knowledge transfer seems particularly useful for crowdsourcing service providers such as Mechanical Turk because they can identify same users in different tasks using a unified identification.

To enable the transfer of knowledge between crowdsourcing tasks, we propose a hierarchical Bayesian model known as Transfer Learning for Crowdsourcing (TLC). We notice that most users have relatively stable characteristics in completing related tasks, so the performance of these users in a target task can be estimated accurately. For each user, we introduce two shared variables, where one models the average performance of a user while the other models the users' variance in task performance in different tasks. For each task, we use another variable to model task-specific factors. Knowledge from different tasks is shared via these variables. To construct the TLC model, we propose a Markov chain Monte Carlo (MCMC) method to infer the latent variables. Our experiments confirm that TLC can effectively transfer knowledge from related auxiliary tasks while avoiding potential negative effects due to task differences.

We summarize the main contributions of this paper as follows:

1. We study a new transfer learning problem in crowdsourcing applications to reduce the cost of crowdsourcing operations. We address the problem of data veracity of crowdsourcing systems. To the best of our knowledge, this is the first work to utilize multiple tasks in crowdsourcing applications.

2. We propose a novel hierarchical Bayesian model, TLC, to solve the knowledge-transfer problem. TLC can solve the data-sparsity problem by exploiting knowledge from related tasks while avoiding the negative effect caused by task differences. We propose an effective inference algorithm to infer the model variables.

3. We conduct experiments on various real-world datasets. The results show that TLC outperforms the previous approaches significantly and reduces the costs of crowdsourcing customers. The results also show that TLC can effectively transfer knowledge from different but related tasks and avoid potential harm brought about by task difference.

2. RELATED WORKS

In this section, we review related works on learning in crowdsourcing context. We summarize the related works in Table 1.

2.1 Transfer Learning

Transfer learning (TL) solves the lack of class label problem in the target application by "borrowing" supervised knowledge from related problems [8]. It has been applied on classification [9], clustering [17], ranking [5], collaborative filtering [3] and social network analysis [18]. For example, Daume et al. [4] proposed a graphical model by applying shared priors to the auxiliary and target tasks to represent common sentiment knowledge. Pan et al. [9] surveyed a series of algorithms on text classification. In [17], Yang introduced auxiliary knowledge from text to reduce the data sparsity in the image clustering task by building translators from vocabulary to image features. Recently, Yahoo's learning-to-rank challenge [4] also promotes transfer learning's application in the ranking field. In addition, more and more transfer learning research works have been proposed to cope with relational data. Cao et al. [17] proposed a novel generative model based on Gaussian Process, which introduced a new kernel to describe the task relations. Recently, Zhong et al. [18], incorporated knowledge in multiple social networks to help infer users' behaviors by assigning social regulars in a topic model. However, these algorithms build models based on gold truths, which means the labeled data for training models are all reliable. This is not true for crowdsourcing tasks, where the labels provided by different users can be noisy and inaccurate. Thus, these previous approaches cannot be directly applied to solve the cross-task crowdsourcing problem studied in this paper.

2.2 Crowdsourcing

Crowdsourcing is a process that involves outsourcing tasks to a distributed group of people. Recently, several real and flexible systems have been launched, such as Amazon Mechanical Turk, which makes related applications practical. In machine learning and data mining research fields, many researchers use crowdsourcing services to solve the lack of labeled-data problem. Related applications range from sentiment classification [12], object recognition [7], ranking [13] to clustering [6]. To obtain labeled data from crowdsourcing systems, researchers commonly provide a certain budget and then distribute the data to different users/labelers according to certain principals or just randomly. Before utilizing crowdsourced labels, one needs to clean the data, since labels from the crowd are contaminated by errors and bias. To achieve this goal, several approaches have been proposed recently [14, 6, 7, 15, 10]. For example, Welinder et al. [15] presented a graphical model that discovers and represents groups of annotators with different sets of skills and knowledge, as well as groups of images that differ qualitatively. Wauthier et al. [14] focused on modeling users' bias in a Bayesian model. Raykar et al. [10] improves the estimation accuracy by distinguishing spammers and normal users. In summary, most of these methods infer the true label for a given instance by estimating both the user ability and the question difficulty. Then the probability that one user provides a correct answer to a question is based on these two factors. However, these methods require a large number of answers for each user and each question. That makes the method impractical in many real-world applications, since the budget for each crowdsourcing task is limited.

[4] http://learningtorankchallenge.yahoo.com

Table 2: Definition of Notations

Notation	Description	Number	Set Notation
		Data	
T_i	ith Task	K	$\mathcal{T} = \{T_i \vert 1 \le i \le K\}$
U_p	pth User	n	$\mathcal{U} = \{U_p \vert 1 \le p \le n\}$
Q_q^i	qth Question in T_i	$\Sigma_i^K m_i$	$\mathcal{Q}^i = \{Q_q^i \vert 1 \le q \le m_i\}$ $\mathcal{Q} = \{\mathcal{Q}^i \vert 1 \le i \le K\}$
A_q^i	True Answer to Q_q^i	$\Sigma_i^K m_i$	$\mathcal{A}^i = \{A_q^i \vert 1 \le q \le m_i\}$ $\mathcal{A} = \{\mathcal{A}^i \vert 1 \le i \le K\}$
L_{pq}^i	Answer to question Q_q^i given by user U_p		$\mathcal{L}^i = \{L_{pq}^i \vert I^i(p,q) > 0, 1 \le p \le n, 1 \le q \le m_i\}$ $\mathcal{L} = \{\mathcal{L}^i \vert 1 \le i \le K\}$
		Model Latent Variables	
B_i	Task-specific factor for T_i	K	$\mathcal{B} = \{B_i \vert 1 \le i \le K\}$
G_p	User dependent characteristic for U_p	n	$\mathcal{G} = \{G_p \vert 1 \le p \le n\}$
M_p	User averaged performance for U_p	n	$\mathcal{M} = \{M_p \vert 1 \le p \le n\}$
C_p^i	Performance of U_p in T_i	$K * n$	$\mathcal{C}^i = \{C_q^i \vert 1 \le p \le n\}$ $\mathcal{C}_p = \{C_q^i \vert 1 \le i \le K\}$ $\mathcal{C} = \{\mathcal{C}_p \vert 1 \le p \le n\}$
D_q^i	Difficulty of Q_q^i	$\Sigma_i^K m_i$	$\mathcal{D}^i = \{D_q^i \vert 1 \le q \le m_i\}$ $\mathcal{D} = \{\mathcal{D}^i \vert 1 \le i \le K\}$

3. PROBLEM FORMULATION

We describe the problem of cross-task crowdsourcing (CTC) in this section. The notations can be found in Table 2. There are n users, K tasks, where the i-th task have m_i questions.

- Let T_i be the i-th task, $\mathcal{T}$ denote the task set.
- Let U_p denote the p-th user, $\mathcal{U}$ denote the user set.
- Let Q_q^i denote the q-th question in T_i, $\mathcal{Q}^i$ denote the questions in T_i and $\mathcal{Q}$ denote the whole question set.
- In a typical crowdsourcing task, each user gives responses to several questions in T_i. Let L_{pq}^i denote the answer of question Q_q^i given by user U_p and then for each task, we obtain a sparse answer matrix $\mathcal{L}^i = \{L_{pq}^i \vert I^i(p,q) > 0, 1 \le p \le n, 1 \le q \le m_i\}$, where $I^i(p,q)$ is the indicator function for the i-th task. If U_p has answered Q_q^i, $I^i(p,q) = 1$, otherwise $I^i(p,q) = 0$. Let $\mathcal{L} = \{\mathcal{L}^i \vert 1 \le i \le K\}$.
- Finally, denote the true answer for the question Q_q^i as A_q^i. In parallel, we obtain $\mathcal{A}^i = \{A_q^i \vert 1 \le q \le m_i\}$ and $A = \{\mathcal{A}^i \vert 1 \le i \le K\}$.

We first consider single-task crowdsourcing problems. For the task T_i, we try to infer answers in $\mathcal{A}^i$ based only on the observations $\mathcal{L}^i$, which is referred to as $(\mathcal{A}^t) = STC(\mathcal{L}^t)$. However, as we stated in the introduction, as individual users can answer only a few questions, some $\mathcal{L}^i$ can be very sparse, and hence the estimation of $\mathcal{A}^i$ can be inaccurate. To solve this problem, the cross-task crowdsourcing (CTC) approach aims to exploit the knowledge in different source tasks T^i that has abundant data, where $i \neq t$ to estimate the

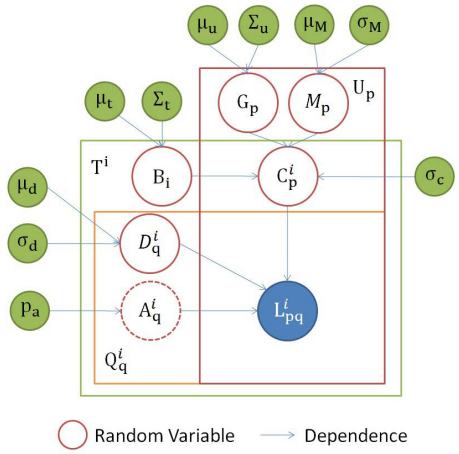


Figure 2: The Graphical Model of TLC
Blue: observed variable; Green: prior; Dashed: variable of interest.

answers $\mathcal{A}^i$ in the target task T^t collaboratively. Mathematically, suppose that t-th task is the target task, the problem can be formulated as $(\mathcal{A}^t) = CTC(\mathcal{L})$. The challenge in CTC is how to transfer the knowledge effectively given that each user may have different performance in different tasks due to certain task-specific factors; for example, different tasks may require different background knowledge. In this paper, we focus on the annotation task where each question is an instance to label, such as a document or an image, whereas the answer to a question is its correspondent label. To simplify the discussion, we assume that in the same task, each question has only limited answers and the number of answer candidates to each question is a constant. Then, for each entry in $\mathcal{L}^i$, $L^i_{pq} \in [1, Z^i]$, Z^i is the number of labels to the questions in T_i.

4. PROBABILISTIC MODEL

In this section, we describe our proposed model, TLC (Transfer Learning for Crowdsourcing). We first present the main idea of performing knowledge transfer, then state the generation process of TLC, and finally introduce an effective inference algorithm to construct TLC models.

4.1 Model Description

Our main idea is that the knowledge from related crowdsourcing tasks can be utilized to regularize the estimation of each user's performance by considering the overlapping users' averaged performance and characteristics, as well as tasks-specific factors as a bridge. This allows us to better know which user is more trustworthy and infer true answers more accurately with fewer user responses.

A graphical representation of TLC can be found in Figure 2. For each user U_p in all tasks, the average performance of U_p over all tasks is defined as M_p, M_p is generated from a single-dimension Gaussian distribution.

$$M_p \sim \mathcal{N}(\mu_M, \sigma_M) \tag{1}$$

μ_M is the mean of all users' averaged performance and σ_M is the corresponding variance. M_p is decided by task independent factors of a user which in practice, could be the user's IQ, ability to focus, education level, etc. The characteristics of a user U_p are defined as G_p, which is generated from a multivariate Gaussian distribution.

$$G_p \sim \mathcal{N}(\mu_u, \Sigma_u) \tag{2}$$

μ_u is the mean of all users' characteristics and Σ_u is the corresponding covariance matrix. G_p is decided by a task dependent factor of U_p, so that G_p accounts for the differences of user performance in different tasks. In practice, G_p can be multidimensional, each dimension may be related to the users' knowledge in a specific domain. We use B_i to denote the task-specific factors for task T_i, B_i is generated from a multivariate Gaussian distribution.

$$B_i \sim \mathcal{N}(\mu_t, \Sigma_t) \tag{3}$$

μ_t is the mean of all task-specific factors and Σ_t is the corresponding covariance matrix. In reality, B_i can be multidimensional, where each dimension can be considered as a factor, such as whether some specific knowledge is required. We use C^i_p to denote the ability of user U_p answering questions in T_i, and let $C \in \mathcal{R}^{n\times K}$ denote the whole performance matrix containing the performance of all users in all tasks. C^i_p is generated from a single-dimension Gaussian distribution.

$$C^i_p \sim \mathcal{N}(G_p * B_i + M_p, \sigma_c) \tag{4}$$

σ_c is the corresponding variance, which models how much the actual user ability will deviate from the expected user ability. A larger value in C^i_p means that U_p is more likely to give correct answers to questions in T_i, while a negative C^i_p means that U_p is adversary and always gives wrong answers on purpose. The dot product of G_p and B_i can be viewed as the interaction of the user's characteristics and how much this characteristic is needed to perform well in this specific task. For example, a male user may have sufficient knowledge about sports but little knowledge about how to apply makeups to oneself. If a task requires certain knowledge on sports but requires no knowledge on makeups, the user may perform very well; otherwise, he may perform badly. On the question side, we use D^i_q to denote the difficulty of the question Q^i_q, where $D^i_q > 0$.

$$D^i_q \sim \frac{1}{e^{\mathcal{N}(\mu_d, \sigma_d)}} \tag{5}$$

Here $D^i_q = \infty$ means Q^i_q is so difficult that even the best users can only guess the answers. $D^i_q = 0$ means that Q^i_q is so easy that even the worst users can always answer it correctly. Eq.(5) can model the fact that very few questions are extremely easy or extremely difficult. In addition, we use A^i_q to denote the true answer for Q^i_q. A^i_q is generated from a discrete distribution, which corresponds to a multiple-choice single answer question.

$$A^i_q \sim Dis(p_a) \tag{6}$$

p_a is a common prior to all questions. Finally, let L^i_{pq} $\{\forall p, q | I^i(p, q) > 0\}$ denote the response (label) given by U_p to Q^i_q. For a question with Z^i possible answers, the probability that a user can provide the correct answer to the question is computed through a logistic function as follows:

$$p(L^i_{pq} = A^i_q | C^i_p, D^i_q) = \frac{1}{1 + (Z^i - 1)e^{-\frac{C^i_p}{D^i_q}}} \tag{7}$$

Algorithm 1 Inference of TLC

Input:$\mathcal{L}$, K, n,$m = \{m_i\}_{i=1}^K$, $Z = \{Z^i\}_{i=1}^K$, Ω, ν, γ, μ_M, σ_M, μ_u, Σ_u, μ_t, Σ_t, σ_c, μ_d, σ_d, p_a, λ_u, λ_t, σ_{MH}, P_{MH}
Output: $\mathcal{A}$
Initialize every $M_p \sim \mathcal{N}(\mu_M, \sigma_M) * 0.01$, $G_p \sim \mathcal{N}(\mu_u, \Sigma_u) * 0.01$, $B_i \sim \mathcal{N}(\mu_t, \Sigma_t) * 0.01$, with a small variance.
Initialize every $A_q^i \sim Dis(p_a)$, $C_p^i \sim \mathcal{N}(G_p * B_i + M_p, \sigma_c)$, $D_q^i \sim \frac{1}{e^{\mathcal{N}(\mu_d, \sigma_d)}}$ randomly
Initialize every $\hat{A}_q^i = 0$, $\hat{C}_p^i = 0$, $\hat{D}_q^i = 0$,$\hat{M}_p = 0$ the current number of iterations $\zeta = 1$.
Calculate W with Eq.(14)
for $\zeta = 1$ to ν **do**
  Update every true answer A_q^i using Eq.(9)
  Update every C_p^i using posterior distribution Eq.(11) with Metropolis Hastings algorithm
  Update every D_q^i using posterior distribution Eq.(10) with Metropolis Hastings algorithm
  Update every M_p using posterior distribution Eq.(12)
  $\hat{A}_q^i = \frac{\hat{A}_q^i * (\zeta-1) + A_q^i}{\zeta}$, $\hat{C}_p^i = \frac{\hat{C}_p^i * (\zeta-1) + C_p^i}{\zeta}$, $\hat{D}_q^i = \frac{\hat{D}_q^i * (\zeta-1) + D_q^i}{\zeta}$, $\hat{M}_p = \frac{\hat{M}_p * (\zeta-1) + M_p}{\zeta}$.
  IF ζ **mod** $\Omega = 0$, Update G_p, B_i for Ω times as
  for $\omega = 1$ to Ω **do**
    $G_p = G_p - \frac{\partial \mathcal{J}(G,B)}{\partial G_p} * \gamma$ with Eq.(15)
    $B_i = B_i - \frac{\partial \mathcal{J}(G,B)}{\partial B_i} * \gamma$ with Eq.(16)
  end for
  ENDIF
end for
Return $\mathcal{A}$

When a user has a larger C_p^i, the answer he/she provides has a higher probability to be the same as the true answer to the question. If a user is adversarial, he may have a negative C_p^i, so that the answer that he gives is more likely to disagree with the true answer. When the C_p^i is near 0, the user has around $\frac{1}{Z^i}$ probability to answer the questions correctly. After inferring the latent variables, the true answer to each question can be calculated by Bayesian theorem:

$$p(A_q^i|L_{pq}^i, D_q^i, C_p^i) \propto p(L_{pq}^i|A_q^i, D_q^i, C_p^i)p(A_q^i, D_q^i, C_p^i) \quad (8)$$

4.2 Inference

We need to infer a total number of six variables G_p^i, M_p, C_p^i, B_i, D_q^i, A_q^i. To construct the model, TLC, we propose a mixture method of Markov Chain Monte Carlo (MCMC) and Gradient Descent (GD), where the MCMC is utilized to infer the low-level variables, user specific ability C_p^i, user averaged performance M_p, problem difficulty D_q^i and true answer A_q^i, while the GD is to find the optimal G_p and B_i, user characteristics and task-specific factors respectively. In order to infer the latent variables from the observed variable L_{pq}^i, we first calculate the posterior distribution of each variable as follows:

$$p(A_q^i|L_{pq}^i, D_q^i, C_p^i) = \frac{p(L_{pq}^i|A_q^i, D_q^i, C_p^i) * p(A_q^i)}{\Sigma_{\hat{A}_q^i=\{T,F\}} p(L_{pq}^i|\hat{A}_q^i, D_q^i, C_p^i) * p(\hat{A}_q^i)} \quad (9)$$

$$p(D_q^i|L_{pq}^i, A_q^i, C_p^i) = \frac{p(L_{pq}^i|A_q^i, D_q^i, C_p^i) * p(D_q^i)}{\int p(L_{pq}^i|A_q^i, \hat{D}_q^i, C_p^i)\,\mathrm{d}\,p(\hat{D}_q^i)} \quad (10)$$

$$p(C_p^i|L_{pq}^i, A_q^i, D_q^i) = \frac{p(L_{pq}^i|A_q^i, D_q^i, C_p^i) * p(C_p^i)}{\int p(L_{pq}^i|A_q^i, D_q^i, \hat{C}_p^i)\,\mathrm{d}\,p(\hat{C}_p^i)} \quad (11)$$

$$p(M_p|\mathcal{C}_p) = \frac{p(\mathcal{C}_p|M_p, \sigma_c) * p(M_p)}{\int p(\mathcal{C}_p|\hat{M}_p, \sigma_c)\,\mathrm{d}\,p(\hat{M}_p)} \quad (12)$$

$$= \mathcal{N}\Big((\frac{\mu_m}{\sigma_m^2} + \frac{\Sigma_{i=1}^K C_p^i}{\sigma_c^2})/(\frac{1}{\sigma_m^2} + \frac{K}{\sigma_c^2}), (\frac{1}{\sigma_m^2} + \frac{K}{\sigma_c^2})^{-1}\Big) \quad (13)$$

Given the observed variable L_{pq}^i, we iteratively sample A_q^i using Eq.(9), sample D_q^i using Eq.(10), sample C_p^i using Eq.(11) and sample M_p using Eq.(12). Since the prior distribution of C_p^i and D_q^i are not conjugate with the likelihood function $p(L_{pq}^i|A_q^i, D_q^i, C_p^i)$, we utilize a Metropolis Hastings algorithm when sampling C_p^i and D_q^i. Take the sampling process of C_p^i as an example. We firstly randomly initialize C_p^i and then generate a new sample $\bar{C}_p^i$ near the previous sample, $\bar{C}_p^i \sim \mathcal{N}(C_p^i, \sigma_{\text{MH}})$, we use Gaussian distribution as the jumping distribution, σ_{MH} is the variance of the jumping distribution. Consequently, we compare the posterior probability density of the new sample with the previous sample. If $p(\bar{C}_p^i|L_{pq}^i, A_q^i, D_q^i) > p(C_p^i|L_{pq}^i, A_q^i, D_q^i)$, the new sample obtains a higher posterior probability density, we accept the new sample and perform update: $C_p^i = \bar{C}_p^i$. Otherwise, we accept the new sample randomly with probability P_{MH}. The above process is repeated until convergence when C_p^i does not change too much. The sampling process of D_q^i is similar to C_p^i, which firstly randomly initialize D_q^i, and then draw a new sample $\bar{D}_q^i$ near D_q^i. Then, we compare the posterior probability density of $\bar{D}_q^i$ and D_q^i and decide whether to accept the new sample or not.

For the user characteristic G_p and task-specific factor B_i, we exploit gradient descent to find the optimal solution via C_p^i. Since each user may give different numbers of answers in each task, we assign a weight to each C_p^i during the inference process. We denote the weight of C_p^i as W_{ip}, $W = \{W_{ip}|1 \le i \le K, 1 \le p \le n\}$. The weight matrix can model our confidence on users' performance C_p^i in each task: the more answers one user gives in a task, the higher confidence we can obtain from the estimation of user's performance in this particular task. If W_{ip} is small, we can predict C_p^i with the knowledge learned from related auxiliary task, which may have sufficient data.

The relation between the weight W_{ip} and the number of answers in T_i may not be linear. After a user answered a number of questions in a task, we gain high confidence in our performance estimation. That is, C_p^i can be trusted and W_{ip} should be high. Thus, we make $W_{ip} \propto \log(r_{i,p})$ where $r_{i,p}$ is the number of responses given by U_p to all questions in T_i. We also normalize all W_{ip} to make the maximum weight 1 and minimum weight 0. Let $\mathbf{MAX} = \text{MAX}(log(r_{i,p}))$ and $\mathbf{MIN} = \text{MIN}(log(r_{i,p}))$,

$$W_{ip} = \frac{log(r_{i,p}) - \mathbf{MIN}}{\mathbf{MAX} - \mathbf{MIN}} \quad (14)$$

The objective function is defined as follows: we minimize the following quantity

$$\mathcal{J}(\mathcal{G}, \mathcal{B}) = \Sigma_{i=1}^K \Sigma_{p=1}^n (W_{ip} * (G_p * B_i + M_p - C_p^i)^2)$$
$$+\lambda_t \Sigma_{i=1}^K \parallel B_i \parallel_f + \lambda_u \Sigma_{p=1}^n \parallel G_p \parallel_f$$

where $\parallel \dot{\parallel}$ is the Frobenius norm, λ_t and λ_u are two trade-off parameters, and μ_c is the mean of all C_p^i. The partial derivative of the objective function with respect to G_p and

Table 3: Summary of Data Characteristics

Collections	# of Users	# of Tasks	# of Questions	# of Responses
Synthetic	40	2	100	3040
Affective Text Analysis	38	6	100	6000
Gender Hobby	42	2	204	3252

B_i are

$$\frac{\partial \mathcal{J}(\mathcal{G},\mathcal{B})}{\partial G_p} = \Sigma_{1\le i\le K} 2(G_p * B_i + M_p - C_p^i)B_i W_{ip} + 2\lambda_u G_p \quad (15)$$

$$\frac{\partial \mathcal{J}(\mathcal{G},\mathcal{B})}{\partial B_i} = \Sigma_{1\le p\le n} 2(G_p * B_i + M_p - C_p^i)G_p W_{ip} + 2\lambda_t B_i \quad (16)$$

Then, these two equations are used to iteratively optimize G_p and B_i until convergence.

Framework The whole inference process can be found in Algorithm 1. Overall, it is an iterative process. In each iteration, MCMC is utilized to infer the low-level variables, user-specific ability C_p^i, user-averaged performance M_p, problem difficulty D_q^i and true answer A_q^i, while the GD is to find the optimal user characteristics G_p and task-specific factors B_i, respectively. ν is the number of iterations, Ω controls how often to update hyper parameters, and γ is the learning rate. Our algorithm is a kind of MCMC algorithm, which convergence guarantee is proved in [1].

Time Complexity We analyze the time complexity of TLC. In the algorithm, the part for updating A and D takes $O(\Sigma_i^K m_i)$ time, and for updating C takes $O(K * n)$ time. Furthermore, the part for updating G and M takes $O(n * \Omega)$ time, and for updating B takes $O(K * \Omega)$ time. In addition, we update the parameters with ν iterations. Thus the overall time complexity is $O(\nu * [(\Sigma_i^K m_i + n * k) + (n + K) * \Omega])$. The baseline model GLAD has a complexity of $O(\nu * (\Sigma_i^K m_i + n * k))$. TLC has a slightly higher complexity than the traditional aggregation model because TLC uses another two variables to model the relationship of user abilities in different tasks. However, the running time of the aggregation algorithm is not our major concern in the whole crowdsourcing process, since most of the time is spent on collecting responses from the crowdsourcing users. This process takes weeks or even months to complete. By exploiting knowledge from related tasks, we can reduce the number of responses needed, which can also reduce the time of the whole process.

5. EXPERIMENTS

In this section, we experimentally verify that our TLC algorithm performs well. We first present some observations and findings in our preliminary empirical study, which motivates us to design the proposed TLC algorithm. Consequently, we give a synthetic example to illustrate the intrinsic properties of TLC. In addition, we compare TLC with several state-of-the-art methods on two realworld data collections, where the proposed method TLC improves the baselines significantly.

For evaluation metrics, we introduce two evaluation metrics Root Mean Square Error (RMSE) and accuracy to measure how accurate the inferred results match the ground truths. RMSE is used in the synthetic dataset as the task is regression, and the smaller the RMSE the better. Accuracy is used in two realworld data collections as our task is classification where higher accuracy is better. We compare TLC with three baselines: (1) Majority Vote: A popular heuristic which does not model user ability and question difficulty(denoted as "Majority" or "MV"). (2) GLAD [16] model considers user ability and question difficulty. (3) DARE [2] model considers user ability, question difficulty and user's advantage to a question. For GLAD and DARE, we implement them in two different ways. One is to build models on only the target task(denoted as "GLAD" and "DARE"), and the other is a naive transfer model where we put questions from different tasks together directly and do not model task differences (denoted as "NaiveTL GLAD" and "NaiveTL DARE").

5.1 Dataset Description

We evaluate the effectiveness of TLC on three data collections. The first one is synthetic. The second one is about affective text analysis [12]. The third is Gender Hobby Dataset, which is collected by ourselves from Amazon Mechanical Turk. The characteristics of the datasets are summarized in Table 3. The datasets are public [5].

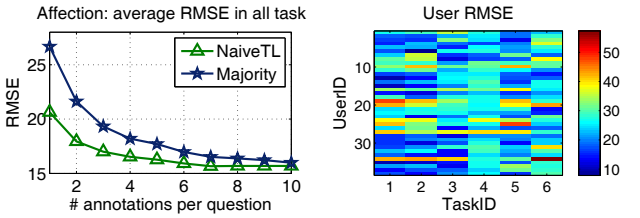
We describe the data-generation process of the synthetic datasets as follows. We build two tasks, where each task has 100 questions and each question has 2 possible answers. There are 40 users who have rated both tasks. The motivation to generate such a dataset is that we can control the performance of each user and hence we can test whether the proposed method TLC can recover these latent variables or not. In addition, we can adjust users' performance in different tasks such that the task differences can be simulated.

1. In order to simulate two related tasks whose task-specific factors are completely different, we set $B_0 = -1$, $B_1 = 1$. We randomly generate $G_p \sim \mathcal{N}(\mu_u = 0, \Sigma_u = 1)$ for each user U_p.
2. We generate $C_p^i \sim \mathcal{N}(G_p * B_i + M_p, \sigma_c)$ and generate $D_q^i \sim \frac{1}{e^{\mathcal{N}(\mu_d,\sigma_d)}}$ randomly, where all $M_p = 1$, $\mu_d = 1$,$\sigma_d = 1$. We also randomly generate gold answer $A_q^i \in \{0, 1\}$ for each question Q_q^i in each tasks.
3. We sample aLL possible responses L_{pq}^i by U_p to Q_q^i, using Eq.(7). A part of all generated L_{pq}^i is used to make up $\mathcal{L}$.

The affective text analysis datasets [12] are related to sentiment analysis. After reading a news headline, users are asked to give an integer rating in [0,100] for each of six emotions to indicate how strong the emotion is. The six emotions are anger, disgust, fear, joy, sadness and surprise. Each emotion corresponds to one task. In total, there are 38 users and 100 headlines, where each user has given emotion ratings to all 6 tasks. Each emotion of each headline is rated by 10 users. A ground truth rating for each headline of each emotion is provided by expert labelers. Our goal is to infer the ground truth ratings with the possibly noisy and biased ratings provided by the users.

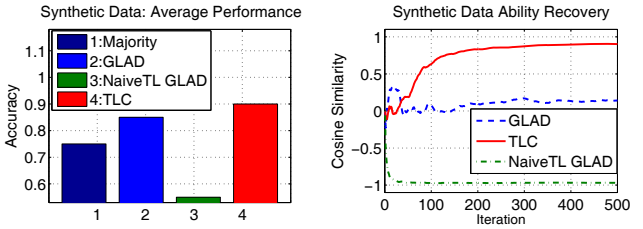
The Gender Hobby dataset is collected by us; this dataset is about different hobby of different genders. When a user

[5] http://www.cse.ust.hk/~kxmo/materials/GenderHobbyDataSet.rar



(a) NaiveTL VS non-TL (b) User Performance

Figure 3: Preliminary Study



(a) Results in Accuracy (b) User ability recovery test

Figure 4: Synthetic Experiments

accepted one of our tasks in Mechanical Turk, he/she is asked to answer 10 questions on 2 topics, half are about sports and half are about makeup and cooking. No user can answer the same question more than once. The user can choose from 2 answers if he knows the answer, or he can choose "I don't know". 57 users gave 1400 responses to our 112 different questions. We choose a set of users with distinct gender related actions to conduct our experiment, these users are either male sports fans, or female housewives. Our goal is to infer the correct answer with the noisy answers provided by the users.

5.2 Preliminary Study

As we stated in the introduction, users' responses to a single task can be very sparse. Our solution is to exploit the knowledge from related auxiliary tasks. We designed a simple strategy to show knowledge transfer based on the affective text analysis datasets. We compare two methods based on voting. The first one is majority voting that always chooses the option chosen by most users as the ground truth. The second one is called Naive Transfer, which exploits knowledge from other tasks. It first estimates the user ability based on users' responses in other tasks. Specifically, it uses majority voting on other tasks to produce "pseudo" truths. Then, each user's common ability in all auxiliary tasks is calculated according to the distance between their responses to these "pseudo" truths. Finally, the normalized common ability of each user is utilized as the weights to combine users' responses in the target task. The result is shown in Figure 3(a). The experiment is repeated 10 times and we show the average performance on all tasks. We can observe that, using the same number of answers per question, the Naive Transfer method can achieve significantly lower RMSE than the non-weighted version. Alternatively, in order to achieve the same RMSE, the weighted majority voting saves up to 50% of user responses per headline. For example, the RMSE of majority voting is 16 when there are 10 ratings. On the other hand, weighted majority voting achieves the same RMSE with only 5 ratings! That means that if tasks are related, knowledge can be transferred across these tasks. However, we also observe that users' performance levels are also different in different tasks, as shown in Figure 3(b). If the difference is too large, directly using users' common ability may harm the final result. However, if we can predict the users' performance in the target task more accurately, we may achieve better results.

5.3 Performance on Synthetic Dataset

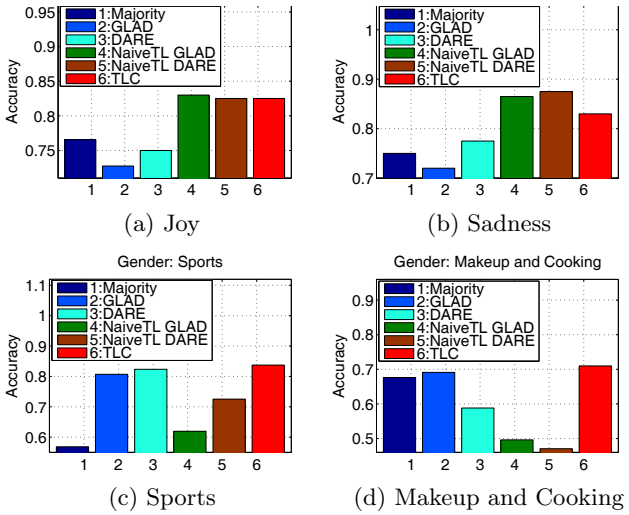
We first describe a synthetic experiment to answer two questions: (1) Can TLC improve the performance of a target task even when the source task and the target task are completely different? (2) Can TLC accurately recover hidden parameters, e.g., users' abilities?

We divided the T_1 and T_2 tasks into 4 tasks. $\mathcal{Q}_t^2$ and $\mathcal{Q}_s^2$ are two disjoint sets of questions in tasks T_2 and $\mathcal{Q}^1$ is the set of questions in T_1. $\mathcal{L}_t^2$, $\mathcal{L}_s^2$, $\mathcal{L}_t^1$ and $\mathcal{L}_s^1$ are the sets of responses given by $\mathcal{U}_t$ to $\mathcal{Q}_t^2$, $\mathcal{U}_s$ to $\mathcal{Q}_s^2$, $\mathcal{U}_t$ to $\mathcal{Q}^1$ and $\mathcal{U}_s$ to $\mathcal{Q}^1$ respectively. $\mathcal{L}_t^2$ is the set of responses for target task, the other $\mathcal{L}_s^2$, $\mathcal{L}_t^1$ and $\mathcal{L}_s^1$ are the set of responses for auxiliary tasks. $\mathcal{L}_s^2$ and $\mathcal{L}_s^1$ are dense and contain information about how the user's performance in source task and in target tasks are related. Specifically for our experiment, all users in three auxiliary source tasks $\mathcal{L}_s^2$, $\mathcal{L}_s^1$ and $\mathcal{L}_t^1$ have answered 50 questions and each question in $\mathcal{Q}_t^2$ have received 2 ratings from users in $\mathcal{U}_t$. Our goal is to infer the true answer for each question in $\mathcal{Q}_t^2$. We introduced three baselines: majority voting, single-task model, e.g., GLAD on $\mathcal{L}_t^2$; naive transfer model: using GLAD on $\mathcal{L}_t = \{\mathcal{L}_t^1, \mathcal{L}_t^2\}$. The result is shown in Fig.(4). The majority voting has about 75% accuracy and the single-task model, GLAD, achieves about 85% accuracy. Due to large task differences, the naive transfer model that combines data from different tasks simply does not work. However, by modeling users' characteristics across tasks and tasks' specific factors, TLC improves them to about 90%. The experiment result shows TLC can transfer knowledge from related but different auxiliary tasks and avoid potentially harm from task differences.

To answer the second question on how well TLC can recover user ability C_p^i in the sparse target task, we compare TLC with baselines. For evaluation, we compute the similarity between the predicted user ability C_p^i and the ground truth $\bar{C}_p^i$, with a normalized cosine similarity which is defined as in Eq.(17). The results are shown in Fig.(4). Note that the majority vote does not model user ability, so we do not take it into consideration.

$$S = Cosine(\frac{X - \bar{X}}{std(X)}, \frac{Y - \bar{Y}}{std(Y)}) \tag{17}$$

We observe that, as the number of iterations increases, the similarity of TLC goes steadily towards 1. That means TLC recovers user ability very well. On the other hand, the similarity of GLAD goes up before the 25-th iteration, but falls quickly and stays around 0 due to severe over-fitting. In addition, the similarity of naive transfer method goes down directly and stays close to -1. This is because the naive transfer method does not model task differences, thus it relates users' abilities in the source tasks to that in the target task in the opposite way. This means the naive transfer method trusts the users with the lowest ability in the target task. This analysis again shows that our proposed TLC model can effectively transfer knowledge from completely different yet related tasks, and the TLC model can recover users' abilities very accurately.



(a) Joy (b) Sadness

(c) Sports (d) Makeup and Cooking

Figure 5: Real-world Experiments
(a)-(b)Result of Affective Text Dataset, (c)-(d) Result of Gender Hobby Dataset

5.4 Performance on Real-world Applications

Here we present the performance of TLC in two real-world applications. Our experiment is divided into two parts. The first part is on the Affective Text Analysis dataset, which shows the effectiveness of TLC on similar tasks. The second part is on the Gender Hobby dataset, which shows that when users show a strong performance correlation in different tasks, we can use the auxiliary data from related tasks to improve performance in target task.

Firstly, we focus on the situation when source tasks and target tasks are very similar. We first convert the emotion score in the original Affection dataset to a true/false question. Emotion score in $[0, 50)$ is converted to false while score in $[50, 100]$ is converted to true. Let T_i denote the i task. For each experiment, we pick a part of the questions $\mathcal{Q}_t^i$ in task T_i to be the target task, while the other questions $\mathcal{Q}_s^i$ in T_i and all questions $\mathcal{Q}^j$ in T_j where $j \neq i$ are used as source tasks. We left 2 answers to each question in the target task and stimulated more responses from each user in source tasks, while keeping the accuracy of each user unchanged. Specifically, users in target task $\mathcal{U}_t$ answered 5 questions in target task $\mathcal{Q}_t^i$ and answered 50 questions in source task $\mathcal{Q}_s^j$ where $j \neq i$; other users $\mathcal{U}_s$ answer 50 questions both in $\mathcal{Q}_s^i$ and in each of $\mathcal{Q}^j$ where $j \neq i$. The performance of TLC and all other baselines are presented in Table 4 and Figure 5. We ran each model 10 times and took the average performance. Due to the difference between tasks, the performances of the compared methods are different, but on the whole, Transfer Models have higher accuracy than Non-Transfer Models. Specifically when using Joy as the target task, the proposed model TLC achieves 82.5% accuracy, which improved the best non-transfer model by 6%. The results show that the information in auxiliary tasks can actually help the target task. TLC has comparable accuracy with Naive Transfer Models, which shows that the TLC model can effectively transfer knowledge from auxiliary tasks when source tasks and target tasks are very similar.

Secondly, we focus on the situation where users show strong performance correlation in different tasks using Gender Hobby datasets. The original dataset contains 400 responses given by 21 users related to 102 questions, where each task has

y

Algorithm	Single-Task			Naive-Transfer		TLC
	MV	GLAD	DARE	GLAD	DARE	
Anger	0.7660	0.8250	0.8250	**0.8400**	0.8000	0.8375
Disgust	0.7635	0.6550	0.7750	**0.8550**	0.8250	0.8100
Fear	0.7562	0.9300	0.9250	0.9350	**0.9500**	0.9475
Joy	0.7655	0.7275	0.7500	**0.8300**	0.8250	0.8250
Sadness	0.7500	0.7200	0.7750	0.8650	**0.8750**	0.8300
Surprise	0.7687	0.7425	**0.8250**	0.7475	0.7500	0.8075

Table 5: Accuracy on Gender Hobby Collections

Algorithm	Single-Task			Naive-Transfer		TLC
	MV	GLAD	DARE	GLAD	DARE	
Sports	0.5686	0.8069	0.8235	0.6196	0.7255	**0.8373**
Makeup/	0.6765	0.6912	0.5882	0.4961	0.4706	**0.7098**

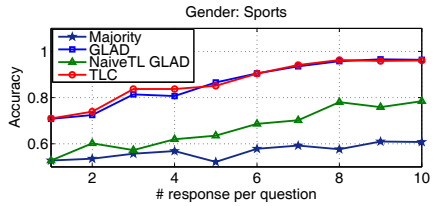
51 questions. We enlarged the dataset by up-sampling: we sampled and added another 21 users with the same accuracy distribution as well as 51 questions with the same difficulty distribution to each task. We left 4 answers to each question in target task and stimulated more responses from each user in the source tasks, while keeping the accuracy of each user unchanged. Denote T_1 to be the sports task, T_2 to be the makeup and cooking task, respectively. We used both tasks as source and target. When we pick a part of the questions $\mathcal{Q}_t^i$ in task T_i as the target task, the other questions $\mathcal{Q}_s^i$ in T_i along with all questions $\mathcal{Q}^{2-i}$ in T_{2-i} are used in source tasks. A part of the users, $\mathcal{U}_t$ answer 50 questions in $\mathcal{Q}^{2-i}$ and about 5 questions in $\mathcal{Q}_t^i$, and the other users $\mathcal{U}_s$ answer 50 questions in both $\mathcal{Q}^{2-i}$ and $\mathcal{Q}_s^i$. Finally, we got 3,252 responses, 204 questions in 2 tasks, 42 users in total. Our goal is to infer the true answer for each question in target task $\mathcal{Q}_t^i$. The result is shown in Fig.(5) and Table 5. We ran each model 10 times and took the average performance. Due to large differences between tasks, the two naive transfer models that do not model task differences do not work well. When using Sports as the target task, the proposed TLC model achieves 83.7% accuracy and outperformed all the baseline models by at least 1.5%. When using Makeup and Cooking tasks as target tasks, the TLC model achieves 70.9% accuracy, improves the best baseline model by about 2%. This experiment demonstrates that the proposed TLC model can transfer knowledge from related but different tasks, and avoid possible negative affects brought about by task differences.

5.5 Parameter Analysis

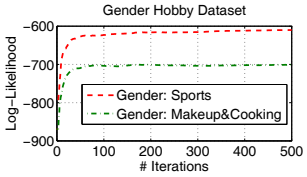
We answer three questions in the subsection to analyze the robustness of TLC: 1. Does TLC converge? 2. How does the data sparsity level affect the TLC performance? 3. Is TLC sensitive to model parameters? We answer these questions via an empirical test, where the results are the average from 10 repeated tests (see Fig.6).

We first plot the likelihood of the model in a target domain. As the model iterates, the log-likelihood of the model becomes larger and larger, until it becomes stable. This shows that our model is able to converge. We then study the impact of target domain sparsity. We use different numbers of responses to each question in the target domain, while keeping the source domain the same. In this experiment, TLC performs better than the non-transfer model in situations where there are fewer responses in the target task. This experiment shows that TLC can effectively transfer knowledge when the target task data is sparse.

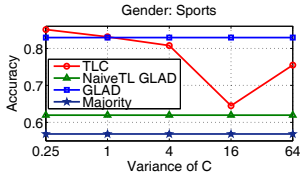
We also experiment on the parameters of the model. A smaller σ_c makes prediction more accurate, because a smaller σ_c better regularizes the difference between the inferred user



(a) Data Sparsity level



(b) Likelihood in target task



(c) σ_c

Figure 6: Impact of parameters

ability C_p^i in the target task and the predicted user ability from the auxiliary task. This shows that the model's predictive ability is accurate. σ_M and σ_d do not affect the result very much, the result is omitted due to page limitations. In reality, we can select the most appropriate parameters by testing the model performance on a validation set. We can keep a small set with gold labels as validation set and then select the parameters which achieve the best performance on it.

5.6 Discussion

In this section, we briefly discuss situations where the TLC model is most suitable. We list a number of situations. First, some of the users in the target tasks have finished plenty of questions in source tasks in the history. Second, users in source and target tasks show strong performance correlation. TLC model is particularly suitable for crowdsourcing service-provider websites such as Mechanical Turk, as well as big crowdsourcing requesters who post many related tasks to a group of stable users. When most of the users in target tasks are new or when the target tasks have no correlation with source tasks, TLC still performs as well as single-task baseline models.

6. CONCLUSION

In this paper, we studied a new problem on how to transfer knowledge under the context of crowdsourcing, where the labels on data instances provided by various crowdsourcing users can be sparse, noisy and unreliable. We consider this problem as cross-task crowdsourcing (CTC), where we exploit transfer learning to learn from related auxiliary tasks. Although the shared user may perform relatively stably in similar tasks, the task-specific differences may degrade the performance. In response, we proposed a hierarchical Bayesian model to transfer the knowledge from related tasks adaptively. To the best of our knowledge, this is the first work to utilize multiple tasks in crowdsourcing applications via transfer learning. The proposed model is flexible in that it can be extended to any number of tasks. We conducted empirical studies on two real datasets: Affective Text Analysis dataset and Gender Hobby dataset collected from Amazon Mechanical Turk, where the proposed algorithm TLC outperforms several state-of-the-art non-transfer models by as high as 6% on accuracy.

Acknowledgement

We thank the support of Hong Kong CERG Grants 621211 and 620812. We thank Rong Pan for discussions, and Shauna Dalton for revisions.

7. REFERENCES

[1] C. Andrieu, N. De Freitas, A. Doucet, and M. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

[2] Y. Bachrach, T. Graepel, T. Minka, and J. Guiver. How to grade a test without knowing the answers—a bayesian graphical model for adaptive crowdsourcing and aptitude testing. *Arxiv preprint arXiv:1206.6386*, 2012.

[3] B. Cao, N. N. Liu, and Q. Yang. Transfer learning for collective link prediction in multiple heterogenous domains. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 159–166, Haifa, Israel, June 2010. Omnipress.

[4] H. Daumé, III and D. Marcu. Domain adaptation for statistical classifiers. *J. Artif. Int. Res.*, 26(1):101–126, May 2006.

[5] K. Duh and A. Fujino. Flexible sample selection strategies for transfer learning in ranking. *Inf. Process. Manage.*, 48(3):502–512, May 2012.

[6] R. G. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 558–566. 2011.

[7] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1953–1961. 2011.

[8] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

[9] W. Pan, E. Zhong, and Q. Yang. Transfer learning for text mining. In C. C. Aggarwal and C. Zhai, editors, *Mining Text Data*, pages 223–257. Springer, 2012.

[10] V. C. Raykar and S. Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *J. Mach. Learn. Res.*, 13:491–518, Mar. 2012.

[11] J. Ross, A. Zaldivar, L. Irani, and B. Tomlinson. Who are the turkers? worker demographics in amazon mechanical turk. *Department of Informatics, University of California, Irvine, USA, Tech. Rep*, 2009.

[12] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[13] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis. Max algorithms in crowdsourcing environments. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 989–998, New York, NY, USA, 2012. ACM.

[14] F. L. Wauthier and M. I. Jordan. Bayesian bias mitigation for crowdsourcing. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1800–1808. 2011.

[15] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2424–2432. 2010.

[16] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in Neural Information Processing Systems*, 22:2035–2043, 2009.

[17] Q. Yang, Y. Chen, G.-R. Xue, W. Dai, and Y. Yu. Heterogeneous transfer learning for image clustering via the social web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 1–9, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[18] E. Zhong, W. Fan, J. Wang, L. Xiao, and Y. Li. Comsoc: adaptive transfer of user behaviors over composite social network. In *KDD'12*, pages 696–704. ACM, 2012.