A Regularized Competition Model for Question Difficulty Estimation in Community Question Answering Services

Quan Wang[†] **Jing Liu**[‡] **Bin Wang**[†] **Li Guo**[†]
[†]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, P. R. China
{wangquan,wangbin,guoli}@iie.ac.cn
[‡]Harbin Institute of Technology, Harbin, P. R. China
jliu@ir.hit.edu.cn

Abstract

Estimating questions' difficulty levels is an important task in community question answering (CQA) services. Previous studies propose to solve this problem based on the question-user comparisons extracted from the question answering threads. However, they suffer from data sparseness problem as each question only gets a limited number of comparisons. Moreover, they cannot handle newly posted questions which get no comparisons. In this paper, we propose a novel question difficulty estimation approach called Regularized Competition Model (RCM), which naturally combines question-user comparisons and questions' textual descriptions into a unified framework. By incorporating textual information, RCM can effectively deal with data sparseness problem. We further employ a K-Nearest Neighbor approach to estimate difficulty levels of newly posted questions, again by leveraging textual similarities. Experiments on two publicly available data sets show that for both well-resolved and newly-posted questions, RCM performs the estimation task significantly better than existing methods, demonstrating the advantage of incorporating textual information. More interestingly, we observe that RCM might provide an automatic way to quantitatively measure the knowledge levels of words.

1 Introduction

Recent years have seen rapid growth in community question answering (CQA) services. They have been widely used in various scenarios, including general information seeking on the web[1], knowledge exchange in professional communities[2], and question answering in massive open online courses (MOOCs)[3], to name a few.

An important research problem in CQA is how to automatically estimate the difficulty levels of questions, i.e., *question difficulty estimation* (QDE). QDE can benefit many applications. Examples include 1) Question routing. Routing questions to appropriate answerers can help obtain quick and high-quality answers (Li and King, 2010; Zhou et al., 2009). Ackerman and McDonald (1996) have demonstrated that routing questions by matching question difficulty level with answerer expertise level will make better use of answerers' time and expertise. This is even more important for enterprise question answering and MOOCs question answering, where human resources are expensive. 2) Incentive mechanism design. Nam et al. (2009) have found that winning point awards offered by reputation systems is a driving factor for user participation in CQA services. Assigning higher point awards to more difficult questions will significantly improve user participation and satisfaction. 3) Linguistics analysis. Researchers in computational linguistics are always interested in investigating the correlation between language and knowledge, to see how the language reflects one's knowledge (Church, 2011). As we will show in Section 5.4, QDE provides an automatic way to quantitatively measure the knowledge levels of words.

Liu et al. (2013) have done the pioneer work on QDE, by leveraging question-user comparisons extracted from the question answering threads. Specifically, they assumed that the difficulty level of a question is higher than the expertise level of the asker (i.e. the user who asked the question), but lower than that of the best answerer (i.e. the user who provided the best answer). A TrueSkill al-

[1]http://answers.yahoo.com/

[2]http://stackoverflow.com/

[3]http://coursera.org/

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1115–1126,
October 25-29, 2014, Doha, Qatar.

gorithm (Herbrich et al., 2006) was further adopted to estimate question difficulty levels as well as user expertise levels from the pairwise comparisons among them. To our knowledge, it is the only existing work on QDE. Yang et al. (2008) have proposed a similar idea, but their work focuses on a different task, i.e., estimating difficulty levels of tasks in crowdsourcing contest services.

There are two major drawbacks of previous methods: 1) *data sparseness problem* and 2) *cold-start problem*. By the former, we mean that under the framework of previous work, each question is compared only twice with the users (once with the asker and the other with the best answerer), which might not provide enough information and contaminate the estimation accuracy. By the latter, we mean that previous work only deals with well-resolved questions which have received the best answers, but cannot handle newly posted questions with no answers received. In many real-world applications such as question routing and incentive mechanism design, however, it is usually required that the difficulty level of a question is known instantly after it is posted.

To address the drawbacks, we propose further exploiting questions' textual descriptions (e.g., title, body, and tags) to perform QDE. Preliminary observations have shown that a question's difficulty level can be indicated by its textual description (Liu et al., 2013). We take advantage of the observations, and assume that if two questions are close in their textual descriptions, they will also be close in their difficulty levels, i.e., the *smoothness assumption*. We employ manifold regularization (Belkin et al., 2006) to characterize the assumption. Manifold regularization is a well-known technique to preserve local invariance in manifold learning algorithms, i.e., nearby points are likely to have similar embeddings (Belkin and Niyogi, 2001). Then, we propose a novel Regularized Competition Model (RCM), which formalizes QDE as minimizing a loss on question-user comparisons with manifold regularization on questions' textual descriptions. As the smoothness assumption offers extra information for inferring question difficulty levels, incorporating it will effectively deal with data sparsity. Finally, we adopt a K-Nearest Neighbor approach (Cover and Hart, 1967) to perform cold-start estimation, again by leveraging the smoothness assumption.

Experiments on two publicly available data sets collected from Stack Overflow show that 1) RCM performs significantly better than existing methods in the QDE task for both well-resolved and cold-start questions. 2) The performance of RCM is insensitive to the particular choice of the term weighting schema (determines how a question's textual description is represented) and the similarity measure (determines how the textual similarity between two questions is measured). The results demonstrate the advantage of incorporating textual information for QDE. Qualitative analysis further reveals that RCM might provide an automatic way to quantitatively measure the knowledge levels of words.

The main contributions of this paper include: 1) We take fully advantage of questions' textual descriptions to address data sparseness problem and cold-start problem which previous QDE methods suffer from. To our knowledge, it is the first time that textual information is introduced in QDE. 2) We propose a novel QDE method that naturally combines question-user comparisons and questions' textual descriptions into a unified framework. The proposed method performs QDE significantly better than existing methods. 3) We demonstrate the practicability of estimating difficulty levels of cold-start questions purely based on their textual descriptions, making various applications feasible in practice. As far as we know, it is the first work that considers cold-start estimation. 4) We explore how a word's knowledge level can be automatically measured by RCM.

The rest of the paper is structured as follows. Section 2 describes the problem formulation and the motivation of RCM. Section 3 presents the details of RCM. Section 4 discusses cold-start estimation. Section 5 reports experiments and results. Section 6 reviews related work. Section 7 concludes the paper and discusses future work.

2 Preliminaries

2.1 Problem Formulation

A CQA service provides a platform where people can ask questions and seek answers from others. Given a CQA portal, consider a specific category where questions on the same topic are asked and answered, e.g., the "C++ programming" category of Stack Overflow. When an asker u_a posts a question q in the category, there will be several answerers to answer the question. Among all the received answers, a best one will be chosen

by the asker or voted by the community. The answerer who provides the best answer is called the best answerer u_b. The other answerers are denoted by $O = \{u_{o_1}, u_{o_2}, \cdots, u_{o_M}\}$. A question answering thread (QA thread) is represented as a quadruplet (q, u_a, u_b, O). Collecting all such QA threads in the category, we get M users and N questions, denoted by $\mathcal{U} = \{u_1, u_2, \cdots, u_M\}$ and $\mathcal{Q} = \{q_1, q_2, \cdots, q_N\}$ respectively. Each user u_m is associated with an expertise score θ_m, representing his/her expertise level. A larger θ_m indicates a higher expertise level of the user. Each question q_n is associated with a difficulty score β_n, representing its difficulty level. A larger β_n indicates a higher difficulty level of the question. Difficulty scores (as well as expertise scores) are assumed to be comparable with each other in the specified category. Besides, each question q_n has a textual description, and is represented as a V-dimensional term vector $\boldsymbol{d}_n$, where V is the vocabulary size.

The *question difficulty estimation* (QDE) task aims to automatically learn the question difficulty scores (β_n's) by utilizing the QA threads $\mathcal{T} = \{(q, u_a, u_b, O) : q \in \mathcal{Q}\}$ as well as the question descriptions $\mathcal{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_N\}$ in the specified category. Note that in Section 2 and Section 3, we consider estimating difficulty scores of resolved questions, i.e., questions with the best answers selected or voted. Estimating difficulty scores of unresolved questions, e.g., newly posted ones, will be discussed in Section 4.

2.2 Competition-based Methods

Liu et al. (2013) have proposed a competition-based method for QDE. The key idea is to 1) extract pairwise competitions from the QA threads and 2) estimate question difficulty scores based on extracted competitions.

To extract pairwise competitions, it is assumed that question difficulty scores and user expertise scores are expressed on the same scale. Given a QA thread (q, u_a, u_b, O), it is further assumed that:

Assumption 1 (pairwise comparison assumption) *The difficulty score of question q is higher than the expertise score of the asker u_a, but lower than that of the best answerer u_b. Moreover, the expertise score of the best answerer u_b is higher than that of the asker u_a, as well as any answerer in O.*[4]

[4]The difficulty score of question q is not assumed to be lower than the expertise score of any answerer in O, since such a user may just happen to see the question and respond to it, rather than knowing the answer well.

Given the assumption, there are $(|O| + 3)$ pairwise competitions extracted from the QA thread, including 1) one competition between the question q and the asker u_a, 2) one competition between the question q and the best answerer u_b, 3) one competition between the best answerer u_b and the asker u_a, and 4) $|O|$ competitions between the best answerer u_b and each of the answerers in O. The question q is the winner of the first competition, and the best answerer u_b is the winner of the remaining $(|O| + 2)$ competitions. These pairwise competitions are denoted by

$$C_q = \{u_a \prec q, q \prec u_b, u_a \prec u_b, u_{o_1} \prec u_b, \cdots, u_{o_M} \prec u_b\},$$

where $i \prec j$ means that competitor j beats competitor i in a competition. Let

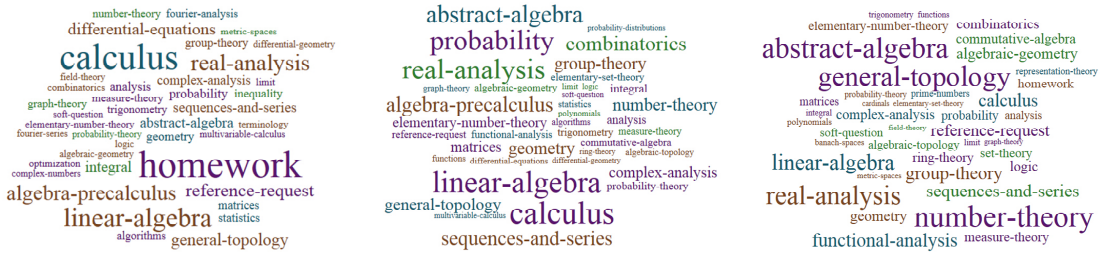
$$C = \bigcup_{q \in \mathcal{Q}} C_q \tag{1}$$

be the set containing all the pairwise competitions extracted from $\mathcal{T}$.

Given the competition set C, Liu et al. (2013) further adopted a TrueSkill algorithm (Herbrich et al., 2006) to learn the competitors' skill levels (i.e. the question difficulty scores and the user expertise scores). TrueSkill assumes that the practical skill level of each competitor follows a normal distribution $N\left(\mu, \sigma^2\right)$, where μ is the average skill level and σ is the estimation uncertainty. Then it updates the estimations in an online mode: for a newly observed competition with its win-loss result, 1) increase the average skill level of the winner, 2) decrease the average skill level of the loser, and 3) shrink the uncertainties of both competitors as more data has been observed. Yang et al. (2008) have proposed a similar competition-based method to estimate tasks' difficulty levels in crowdsourcing contest services, by leveraging PageRank (Page et al., 1999) algorithm.

2.3 Motivating Discussions

The methods introduced above estimate competitors' skill levels based solely on the pairwise competitions among them. The more competitions a competitor participates in, the more accurate the estimation will be. However, according to the pairwise comparison assumption (Assumption 1), each question participates in only two competitions, one with the asker and the other with the best answerer. Hence, there might be no enough information to accurately infer its difficulty score. We call this the *data sparseness problem*.



(a) Low difficulty. (b) Medium difficulty. (c) High difficulty.

Figure 1: Tag clouds of SO/Math questions with different difficulty levels.

Taking advantage of additional metadata has been demonstrated to be an effective way of dealing with data sparsity in various applications such as collaborative filtering (Claypool et al., 1999; Schein et al., 2002) and personalized search (Dou et al., 2007; Sugiyama et al., 2004). The rationale behind is to bridge the gap among users/items by leveraging their similarities based on the metadata. As for QDE, preliminary observations have shown that a question's difficulty level can be indicated by its textual description (Liu et al., 2013). As an example, consider the QA threads in the "mathematics" category of Stack Overflow. Divide the questions into three groups: 1) low difficulty, 2) medium difficulty, and 3) high difficulty, according to their difficulty scores estimated by TrueSkill. Figure 1 visualizes the frequency distribution of tags in each group, where the size of each tag is in proportion to its frequency in the group. The results indicate that the tags associated with the questions do have the ability to reflect the questions' difficulty levels, e.g., low difficulty questions usually have tags such as "homework" and "calculus", while high difficulty ones usually have tags such as "general topology" and "number theory". We further calculate the Pearson correlation coefficient (Rodgers and Nicewander, 1988) between 1) the gap between the averaged difficulty scores in each two groups and 2) the Euclidean distance between the aggregated textual descriptions in each two groups . The result is $r = 0.6424$, implying that the difficulty gap is positively correlated with the textual distance. In other words, the more similar two questions' textual descriptions are, the more close their difficulty levels are. Therefore, we take the textual information to bridge the difficulty gap among questions, by assuming that

Assumption 2 (smoothness assumption) *If two questions q_i and q_j are close in their textual descriptions $\boldsymbol{d}_i$ and $\boldsymbol{d}_j$, they will also be close in their difficulty scores β_i and β_j.*

The smoothness assumption brings us additional information about question difficulty scores by inferring textual similarities. It serves as a supplement to the pairwise competitions, and might help address the data sparseness problem which previous methods suffer from.

3 Modeling Text Similarities for QDE

This section presents a novel Regularized Competition Model (RCM) for QDE, which combines the pairwise competitions and the textual descriptions into a unified framework. RCM can alleviate the data sparseness problem and perform more accurate estimation.

3.1 Regularized Competition Model

We start with several notations. As question difficulty scores can be directly compared with user expertise scores, we take questions as pseudo users. Let $\bar{\boldsymbol{\theta}} \in \mathbb{R}^{M+N}$ denote the skill levels (i.e. the expertise scores and the difficulty scores) of all the (pseudo) users:

$$\bar{\theta}_i = \begin{cases} \theta_i, & 1 \le i \le M, \\ \beta_{i-M}, & M < i \le M+N, \end{cases}$$

where $\bar{\theta}_i$ is the i-th entry of $\bar{\boldsymbol{\theta}}$. The first M entries are the user expertise scores, denoted by $\bar{\boldsymbol{\theta}}_u \in \mathbb{R}^M$. The last N entries are the question difficulty scores, denoted by $\bar{\boldsymbol{\theta}}_q \in \mathbb{R}^N$. Let $\bar{\theta}_i^{(u)}$ and $\bar{\theta}_i^{(q)}$ denote the i-th entries of $\bar{\boldsymbol{\theta}}_u$ and $\bar{\boldsymbol{\theta}}_q$ respectively.

Exploiting Pairwise Competitions. We define a loss on each pairwise competition $i \prec j$:

$$\ell\left(\bar{\theta}_i, \bar{\theta}_j\right) = \max\left(0, \delta - \left(\bar{\theta}_j - \bar{\theta}_i\right)\right)^p, \qquad (2)$$

where p is either 1 or 2. The loss is defined on the skill gap between the two competitors, i.e., $\bar{\theta}_j - \bar{\theta}_i$,

measuring the inconsistency between the expected outcome and the actual outcome. If the gap is larger than a predefined threshold δ, competitor j would probably beat competitor i in the competition, which coincides with the actual outcome. Then the loss will be zero. Otherwise, there is a higher chance that competitor j loses the competition, which goes against the actual outcome. Then the loss will be greater than zero. The smaller the gap is, the higher the chance of inconsistency becomes, and the greater the loss will be. Note that the threshold δ can take any positive value since we do not pose a norm constraint on $\bar{\boldsymbol{\theta}}$.[5] Without loss of generality we take $\delta = 1$ throughout this paper. As we will show in Section 3.2, the loss defined in Eq. (2) has some similarity with the SVM loss (Chapelle, 2007). We name it hinge loss when $p = 1$, and quadratic loss when $p = 2$.

Given the competition set C, estimating skill levels of (pseudo) users then amounts to solving the following optimization problem:

$$\min_{\bar{\boldsymbol{\theta}}} \sum_{(i \prec j) \in C} \ell\left(\bar{\theta}_i, \bar{\theta}_j\right) + \frac{\lambda_1}{2} \bar{\boldsymbol{\theta}}^T \bar{\boldsymbol{\theta}}, \tag{3}$$

where the first term is the empirical loss measuring the total inconsistency; the second term is a regularizer to prevent overfitting; and $\lambda_1 \geq 0$ is a trade-off coefficient. It is also a competition-based QDE method, called Competition Model (CM).

Exploiting Question Descriptions. Manifold regularization is a well-known technique used in manifold learning algorithms to preserve local invariance, i.e., nearby points are likely to have similar embeddings (Belkin and Niyogi, 2001). In QDE, the smoothness assumption expresses similar "invariance", i.e., nearby questions (in terms of textual similarities) are likely to have similar difficulty scores. Hence, we characterize the assumption with the following manifold regularizer:

$$\begin{aligned} \mathcal{R} &= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\bar{\theta}_i^{(q)} - \bar{\theta}_j^{(q)}\right)^2 w_{ij} \\ &= \bar{\boldsymbol{\theta}}_q^T \mathbf{D} \bar{\boldsymbol{\theta}}_q - \bar{\boldsymbol{\theta}}_q^T \mathbf{W} \bar{\boldsymbol{\theta}}_q = \bar{\boldsymbol{\theta}}_q^T \mathbf{L} \bar{\boldsymbol{\theta}}_q, \end{aligned} \tag{4}$$

where w_{ij} is the textual similarity between question i and question j; $\mathbf{W} \in \mathbb{R}^{N \times N}$ is the similarity matrix with the (i, j)-th entry being w_{ij}; $\mathbf{D} \in \mathbb{R}^{N \times N}$ is a diagonal matrix with the i-th entry on the diagonal being $d_{ii} = \sum_{j=1}^{N} w_{ij}$; and $\mathbf{L} = \mathbf{D} - \mathbf{W} \in \mathbb{R}^{N \times N}$ is the graph Laplacian (Chung, 1997). Minimizing $\mathcal{R}$ results in the smoothness assumption: for any questions i and j, if their textual similarity w_{ij} is high, the difficulty gap $\left(\bar{\theta}_i^{(q)} - \bar{\theta}_j^{(q)}\right)^2$ will be small.

A Hybrid Method. Combining Eq. (3) and Eq. (4), we obtain RCM, which amounts to the following optimization problem:

$$\min_{\bar{\boldsymbol{\theta}}} \sum_{(i \prec j) \in C} \ell\left(\bar{\theta}_i, \bar{\theta}_j\right) + \frac{\lambda_1}{2} \bar{\boldsymbol{\theta}}^T \bar{\boldsymbol{\theta}} + \frac{\lambda_2}{2} \bar{\boldsymbol{\theta}}_q^T \mathbf{L} \bar{\boldsymbol{\theta}}_q. \tag{5}$$

Here $\lambda_2 \geq 0$ is also a trade-off coefficient. The advantages of RCM include 1) It naturally formalizes QDE as minimizing a manifold regularized loss function, which seamlessly integrates both the pairwise competitions and the textual descriptions. 2) By incorporating textual information, it can address the data sparseness problem which previous methods suffer from, and perform significantly better in the QDE task.

3.2 Learning Algorithm

Redefine the k-th pairwise competition (assumed to be carried out between competitors i and j) as $(\boldsymbol{x}_k, y_k)$. $\boldsymbol{x}_k \in \mathbb{R}^{M+N}$ indicates the competitors:

$$x_i^{(k)} = 1, x_j^{(k)} = -1, \text{and } x_l^{(k)} = 0 \text{ for any } l \neq i, j,$$

where $x_l^{(k)}$ is the l-th entry of $\boldsymbol{x}_k$. $y_k \in \{1, -1\}$ is the outcome: if competitor i beats competitor j, $y_k = 1$; otherwise, $y_k = -1$. The objective in Eq. (5) can then be rewritten as

$$\mathcal{L}\left(\bar{\boldsymbol{\theta}}\right) = \sum_{k=1}^{|C|} \max\left(0, 1 - y_k \left(\bar{\boldsymbol{\theta}}^T \boldsymbol{x}_k\right)\right)^p + \frac{1}{2} \bar{\boldsymbol{\theta}}^T \mathbf{Z} \bar{\boldsymbol{\theta}},$$

where $\mathbf{z} = \begin{pmatrix} \lambda_1 \mathbf{I}_M & \mathbf{0} \\ \mathbf{0} & \lambda_1 \mathbf{I}_N + \lambda_2 \mathbf{L} \end{pmatrix}$ is a block matrix; $\mathbf{I}_M \in \mathbb{R}^{M \times M}$ and $\mathbf{I}_N \in \mathbb{R}^{N \times N}$ are identity matrices; $p = 1$ corresponds to the hinge loss, and $p = 2$ the quadratic loss. It is clear that the loss defined in Eq. (2) has the same format as the SVM loss.

The objective $\mathcal{L}$ is differentiable for the quadratic loss but non-differentiable for the hinge loss. We employ a subgradient method (Boyd et al., 2003) to solve the optimization problem. The algorithm starts at a point $\bar{\boldsymbol{\theta}}_0$ and, as many iterations as needed, moves from $\bar{\boldsymbol{\theta}}_t$ to $\bar{\boldsymbol{\theta}}_{t+1}$ in the direction of the negative subgradient:

$$\bar{\boldsymbol{\theta}}_{t+1} = \bar{\boldsymbol{\theta}}_t - \gamma_t \nabla \mathcal{L}\left(\bar{\boldsymbol{\theta}}_t\right),$$

[5] Given any $\bar{\theta}_i$, $\bar{\theta}_j$, and δ, there always exists a linear transformation which keeps the sign of $\left(\delta - \left(\bar{\theta}_j - \bar{\theta}_i\right)\right)$ unchanged.

Algorithm 1 Regularized Competition Model

Require: competition set C and description set $\mathcal{D}$
1: $\bar{\boldsymbol{\theta}}_0 \leftarrow \mathbf{1}$
2: **for** $t = 0 : T-1$ **do**
3: $\quad \mathcal{K}_t \leftarrow \left\{k : 1 - y_k\left(\bar{\boldsymbol{\theta}}_t^T \boldsymbol{x}_k\right) > 0\right\}$
4: $\quad \nabla\mathcal{L}\left(\bar{\boldsymbol{\theta}}_t\right) \leftarrow$ calculated by Eq. (6)
5: $\quad \bar{\boldsymbol{\theta}}_{t+1} \leftarrow \bar{\boldsymbol{\theta}}_t - \gamma_t \nabla\mathcal{L}\left(\bar{\boldsymbol{\theta}}_t\right)$
6: $\quad \boldsymbol{\Theta}_{t+1} \leftarrow \left\{\bar{\boldsymbol{\theta}}_0, \bar{\boldsymbol{\theta}}_1, \cdots, \bar{\boldsymbol{\theta}}_{t+1}\right\}$
7: $\quad \bar{\boldsymbol{\theta}}_{t+1} \leftarrow \arg\min_{\bar{\boldsymbol{\theta}} \in \boldsymbol{\Theta}_{t+1}} \mathcal{L}\left(\bar{\boldsymbol{\theta}}\right)$
8: **end for**
9: **return** $\bar{\boldsymbol{\theta}}_T$

where $\gamma_t > 0$ is the learning rate. The subgradient is calculated as

$$\nabla\mathcal{L}\left(\bar{\boldsymbol{\theta}}_t\right) = \begin{cases} \mathbf{Z}\bar{\boldsymbol{\theta}}_t - \sum\limits_{k\in\mathcal{K}_t} y_k \boldsymbol{x}_k, & p=1, \\ \mathbf{Z}\bar{\boldsymbol{\theta}}_t + 2\sum\limits_{k\in\mathcal{K}_t} \boldsymbol{x}_k \boldsymbol{x}_k^T \bar{\boldsymbol{\theta}}_t - 2\sum\limits_{k\in\mathcal{K}_t} y_k \boldsymbol{x}_k, & p=2, \end{cases} \quad (6)$$

where $\mathcal{K}_t = \left\{k : 1 - y_k\left(\bar{\boldsymbol{\theta}}_t^T \boldsymbol{x}_k\right) > 0\right\}$. As it is not always a descent method, we keep track of the best point found so far (Boyd et al., 2003):

$$\bar{\boldsymbol{\theta}}_{t+1} = \arg\min_{\bar{\boldsymbol{\theta}} \in \boldsymbol{\Theta}_{t+1}} \mathcal{L}\left(\bar{\boldsymbol{\theta}}\right),$$

where $\boldsymbol{\Theta}_{t+1} = \left\{\bar{\boldsymbol{\theta}}_0, \bar{\boldsymbol{\theta}}_1, \cdots, \bar{\boldsymbol{\theta}}_{t+1}\right\}$. The whole procedure is summarized in Algorithm 1.

Convergence. For constant learning rate (i.e., $\gamma_t = \gamma$), Algorithm 1 is guaranteed to converge to within some range of the optimal value, i.e.,

$$\lim_{t\to\infty} \mathcal{L}\left(\bar{\boldsymbol{\theta}}_t\right) - \mathcal{L}^* < \epsilon,$$

where $\mathcal{L}^*$ denotes the minimum of $\mathcal{L}(\cdot)$, and ϵ is a constant defined by the learning rate γ. For more details, please refer to (Boyd et al., 2003). During our experiments, we set the iteration number as $T = 1000$ and the learning rate as $\gamma_t = 0.001$, and convergence was observed.

Complexity. For both the hinge loss and the quadratic loss, the time complexity (per iteration) and the space complexity of RCM are both $O\left(|C| + \eta N^2\right)$. Here, $|C|$ is the total number of competitions, M and N are the numbers of users and questions respectively, and η is the ratio of non-zero entries in the graph Laplacian $\mathbf{L}$.[6] In the analysis, we have assumed that $M \ll \eta N^2$ and $N \ll \eta N^2$.

[6]Owing to the sparse nature of questions' textual descriptions, the graph Laplacian $\mathbf{L}$ is usually sparse, with about 70% entries being zero according to our experiments.

4 Cold-Start Estimation

Previous sections discussed estimating difficulty scores of resolved questions, from which pairwise competitions could be extracted. However, for newly posted questions without any answers received, no competitions could be extracted and none of the above methods work. We call it the *cold-start problem*.

We heuristically apply a K-Nearest Neighbor (KNN) approach (Cover and Hart, 1967) to cold-start estimation, again by leveraging the smoothness assumption. The key idea is to propagate difficulty scores from well-resolved questions to cold-start ones according to their textual similarities. Specifically, suppose that there exists a set of well-resolved questions whose difficulty scores have already been estimated by a QDE method. Given a cold-start question q^*, we first pick K well-resolved questions that are closest to q^* in textual descriptions, referred to as the nearest neighbors. The difficulty score of question q^* is then predicted as the averaged difficulty scores of its nearest neighbors. The KNN method bridges the gap between cold-start and well-resolved questions by inferring their textual similarities, and might effectively deal with the cold-start problem.

5 Experiments

We have conducted experiments to test the effectiveness of RCM in estimating difficulty scores of both well-resolved and cold-start questions. Moreover, we have explored how a word's difficulty level can be quantitatively measured by RCM.

5.1 Experimental Settings

Data Sets. We obtained a publicly available data set of Stack Overflow between July 31, 2008 and August 1, 2012[7], containing QA threads in various categories. We considered the categories of "C++ programming" and "mathematics", and randomly sampled about 10,000 QA threads from each category, denoted by SO/CPP and SO/Math respectively. For each question, we took the title and body fields as its textual description. For both data sets, stop words in a standard list[8] and words whose total frequencies are less than 10 were removed. Table 1 gives the statistics of the data sets.

[7]http://blog.stackoverflow.com/category/cc-wiki-dump/

[8]http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

	# users	# questions	# competitions	# words
SO/CPP	14,884	10,164	50,043	2,208
SO/Math	6,564	10,528	40,396	2,009

Table 1: Statistics of the data sets.

For evaluation, we randomly sampled 600 question pairs from each data set, and asked annotators to compare the difficulty levels of the questions in each pair. We had two graduate students majoring in computer science annotate the SO/CPP questions, and two majoring in mathematics annotate the SO/Math questions. For each question, only the title, body, and tags were exposed to the annotators. Given a question pair (q_1, q_2), the annotators were asked to give one of the three labels: $q_1 \succ q_2$, $q_2 \succ q_1$, or $q_1 = q_2$, which respectively means that question q_1 has a higher, lower, or equal difficulty level compared with question q_2. We used Cohen's kappa coefficient (Cohen, 1960) to measure the inter-annotator agreement. The result is $\kappa = 0.7533$ on SO/CPP and $\kappa = 0.8017$ on SO/Math, indicating that the inter-annotator agreement is quite substantial on both data sets. After removing the question pairs with inconsistent labels, we got 521 annotated SO/CPP question pairs and 539 annotated SO/Math question pairs.

We further randomly split the annotated question pairs into development/test/cold-start sets, with the ratio of 2:2:1. The first two sets were used to evaluate the methods in estimating difficulty scores of resolved questions. Specifically, the development set was used for parameter tuning and the test set was used for evaluation. The last set was used to evaluate the methods in cold-start estimation, and the questions in this set were excluded from the learning process of RCM as well as any baseline method.

Baseline Methods. We considered three baseline methods: PageRank (PR), TrueSkill (TS), and CM, which are based solely on the pairwise competitions.

- PR first constructs a competitor graph, by creating an edge from competitor i to competitor j if j beats i in a competition. A PageRank algorithm (Page et al., 1999) is then utilized to estimate the relative importance of the nodes, i.e., question difficulty scores and user expertise scores. The damping factor was set from 0.1 to 0.9 in steps of 0.1.
- TS has been applied to QDE by Liu et al. (2013). We set the model parameters in the same way as they suggested.
- CM performs QDE by solving Eq. (3). We set λ_1 in $\{0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$.

We compared RCM with the above baseline methods. In RCM, both parameters λ_1 and λ_2 were set in $\{0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$.

Evaluation Metric. We employed accuracy (ACC) as the evaluation metric:

$$\text{ACC} = \frac{\text{\# correctly judged question pairs}}{\text{\# all question pairs}}.$$

A question pair is regarded as correctly judged if the relative difficulty ranking given by an estimation method is consistent with that given by the annotators. The higher the accuracy is, the better a method performs.

5.2 Estimation for Resolved Questions

The first experiment tested the methods in estimating difficulty scores of resolved questions.

Estimation Accuracies. We first compared the estimation accuracies of PR, TS, CM, and RCM on the test sets of SO/CPP and SO/Math, obtained with the best parameter settings determined by the development sets. Table 2 gives the results, where "H" denotes the hinge loss and "Q" the quadratic loss. In RCM, to calculate the graph Laplacian $\mathbf{L}$, we adopted Boolean term weighting schema and took Jaccard coefficient as the similarity measure. From the results, we can see that 1) RCM performs significantly better than the baseline methods on both data sets (t-test, p-value < 0.05), demonstrating the advantage of exploiting questions' textual descriptions for QDE. 2) The improvements of RCM over the baseline methods on SO/Math are greater than those on SO/CPP, indicating that the textual descriptions of the SO/Math questions are more powerful in reflecting their difficulty levels. The reason is that the SO/Math questions are much more heterogeneous, belonging to various subfields of mathematics. The difficulty gaps among different subfields are sometimes obvious (e.g., a question in topology in general has a higher difficulty level than a question in linear algebra), making the textual descriptions more powerful in distinguishing the difficulty levels.

Graph Laplacian Variants. We further investigated the performances of different term weighting schemas and similarity measures in the graph

	PR	TS	CM		RCM	
			H	Q	H	Q
SO/CPP	0.5876	0.6134	0.6340	0.6753	**0.7371**	0.7268
SO/Math	0.6067	0.6109	0.6527	0.6820	**0.7699**	**0.7699**

Table 2: ACC of different methods for well-resolved questions.

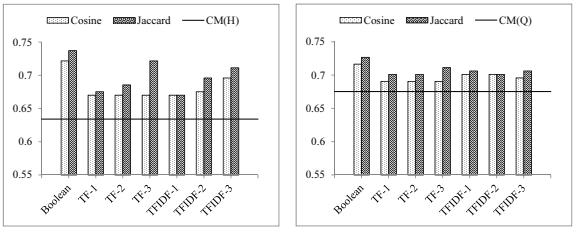
Notation	Definition
Boolean	$v(w,q) = \begin{cases} 1, & \text{if word } w \text{ occurs in question } q \\ 0, & \text{otherwise} \end{cases}$
TF-1	$v(w,q) = f(w,q)$, the number of occurrences
TF-2	$v(w,q) = \log(f(w,q)+1)$
TF-3	$v(w,q) = 0.5 + \frac{0.5 \times f(w,q)}{\max\{f(w,q) : w \in q\}}$
TFIDF-1	$v(w,q) = \text{TF-1} \times \log \frac{\lvert Q \rvert}{\lvert\{q \in Q : w \in q\}\rvert}$
TFIDF-2	$v(w,q) = \text{TF-2} \times \log \frac{\lvert Q \rvert}{\lvert\{q \in Q : w \in q\}\rvert}$
TFIDF-3	$v(w,q) = \text{TF-3} \times \log \frac{\lvert Q \rvert}{\lvert\{q \in Q : w \in q\}\rvert}$
Cosine	$\text{Sim}(\boldsymbol{d}_1, \boldsymbol{d}_2) = \frac{\boldsymbol{d}_1^T \boldsymbol{d}_2}{\lVert\boldsymbol{d}_1\rVert \times \lVert\boldsymbol{d}_2\rVert} \in [0,1]$
Jaccard	$\text{Sim}(\boldsymbol{d}_1, \boldsymbol{d}_2) = \frac{\boldsymbol{d}_1^T \boldsymbol{d}_2}{\lVert\boldsymbol{d}_1\rVert^2 + \lVert\boldsymbol{d}_2\rVert^2 - \lVert\boldsymbol{d}_1\rVert \times \lVert\boldsymbol{d}_2\rVert} \in [0,1]$

Table 3: Different term weighting schemas and similarity measures.

Laplacian. The term weighting schema determines how a question's textual description is represented. We explored a Boolean schema, three TF schemas, and three TFIDF schemas (Salton and Buckley, 1988). The similarity measure determines how the textual similarity between two questions is calculated. We explored the Cosine similarity and the Jaccard coefficient (Huang, 2008). Detailed descriptions are given in Table 3.

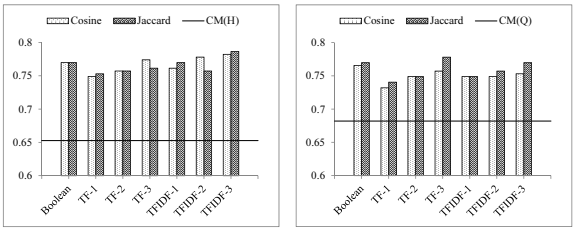
Figure 2 and Figure 3 show the estimation accuracies of the RCM variants on the test sets of SO/CPP and SO/Math respectively, again obtained with the best parameter settings determined by the development sets. The performance of CM is also given (the straight lines in the figures).[9] From the results, we can see that 1) All the RCM variants can improve over CM on both data sets, and most of the improvements are significant (t-test, p-value < 0.05). This further demonstrates that the effectiveness of incorporating textual descriptions is not affected by the particular choice of the term weighting schema or similarity measure. 2) Boolean term weighting schema performs the best, considering different similarity measures, loss types, and data sets collectively. 3) Jaccard

[9]CM performs better than PR and TS on both data sets.



(a) Hinge loss. (b) Quadratic loss.

Figure 2: ACC of RCM variants for well-resolved questions on SO/CPP.



(a) Hinge loss. (b) Quadratic loss.

Figure 3: ACC of RCM variants for well-resolved questions on SO/Math.

coefficient performs as well as Cosine similarity on SO/Math, but almost consistently better on SO/CPP. Throughout the experiments, we adopted Boolean term weighting schema and Jaccard coefficient to calculate the graph Laplacian.

5.3 Estimation for Cold-Start Questions

The second experiment tested the methods in estimating difficulty scores of cold-start questions. We employed Boolean term weighting schema to represent a cold-start question, and utilized Jaccard Coefficient to select its nearest neighbors.

Figure 4 and Figure 5 list the cold-start estimation accuracies of different methods on SO/CPP and SO/Math respectively, with different K values (the number of nearest neighbors). As the accuracy oscillates drastically with a K value smaller than 11 on SO/CPP and smaller than 6 on SO/Math, we report the results with $K \in [11, 20]$ on SO/CPP and $K \in [6, 15]$ on SO/Math. The averaged (over different K values) cold-start estimation accuracies are further given in Table 4. All the results are reported on the cold-start sets, with the optimal parameter settings adopted in Section 5.2. From the results, we can see that 1) Cold-start estimation is possible, and can achieve a considerably high accuracy by choosing a proper method (e.g. RCM), making applications such as better question routing and better incentive mechanism

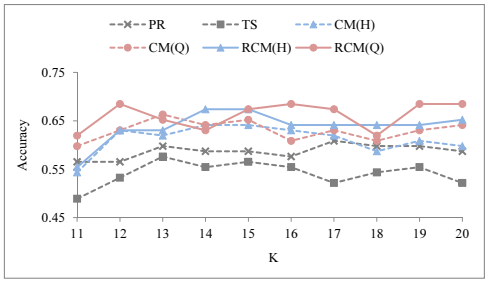


Figure 4: ACC of different methods for cold-start questions on SO/CPP.

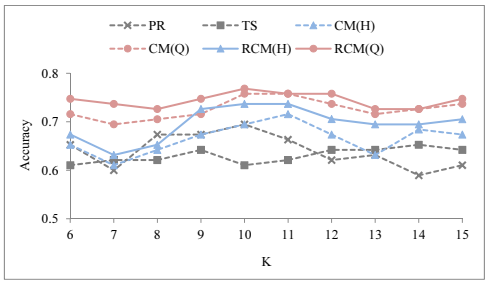


Figure 5: ACC of different methods for cold-start questions on SO/Math.

design feasible in practice. 2) As the value of K varies, RCM (the red/blue solid line) performs almost consistently better than CM with the same loss type (the red/blue dotted line), as well as PR and TS (the gray dotted lines), showing the advantages of RCM in the cold-start estimation. 3) The cold-start estimation accuracies on SO/Math are higher than those on SO/CPP, again demonstrating that the textual descriptions of the SO/Math questions are more powerful in reflecting their difficulty levels. This is consistent with the phenomenon observed in Section 5.2.

5.4 Difficulty Levels of Words

The third experiment explored how a word's difficulty level can be measured by RCM automatically and quantitatively.

On both SO/CPP and SO/Math, we evenly split the range of question difficulty scores (estimated by RCM) into 10 buckets, and assigned questions to the buckets according to their difficulty scores. A larger bucket ID indicates a higher difficulty level. Then, given a word w, we calculated its frequency in each bucket as follows:

$$f_i(w) = \frac{\text{\# questions in bucket } i \text{ where } w \text{ occurs}}{\text{\# all questions in bucket } i}.$$

To make the frequency meaningful, buckets with less than 50 questions were discarded. We picked

	PR	TS	CM		RCM	
			H	Q	H	Q
SO/CPP	0.5870	0.5413	0.6120	0.6304	0.6380	**0.6609**
SO/Math	0.6411	0.6305	0.6653	0.7263	0.6958	**0.7442**

Table 4: Averaged ACC of different methods for cold-start questions.

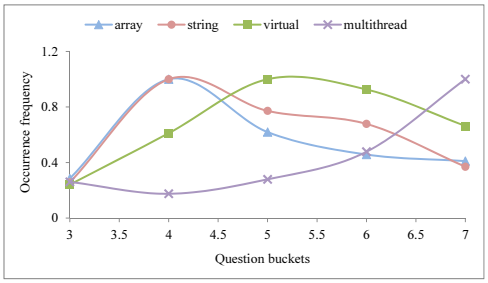


Figure 6: Frequencies of different words in the buckets on SO/CPP.

four words from each data set as examples. Their normalized frequencies in different buckets are shown in Figure 6 and Figure 7. On SO/CPP, we can observe that "array" and "string" occur most frequently in questions with lower difficulty levels, "virtual" higher, and "multithread" the highest. It coincides with the intuition: "array" and "string" are usually related to some basic concepts in programming language, while "virtual" and "multithread" usually discuss more advanced topics. Similar phenomena can be observed on SO/Math. The results indicate that RCM might provide an automatic way to measure the difficulty levels of words.

6 Related Work

QDE is relevant to the problem of estimating task difficulty levels and user expertise levels in crowdsourcing services (Yang et al., 2008; Whitehill et al., 2009). Studies on this problem fall into two categories: 1) binary response based and 2) partially ordered response based. In the first category, binary responses (i.e. whether the solution provided by a user is correct or not) are observed, and techniques based on item response theory are further employed (Whitehill et al., 2009; Welinder et al., 2010; Zhou et al., 2012). In the second category, partially ordered responses (i.e. which of the two given solutions is better) are observed, and pairwise comparison based methods are further adopted (Yang et al., 2008; Liu et al., 2013). QDE belongs to the latter.

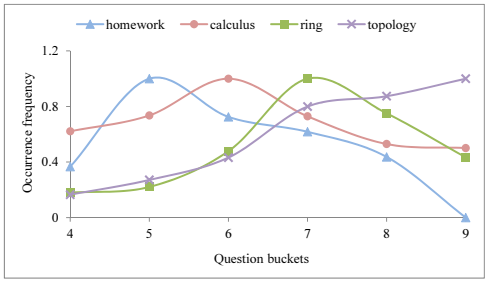


Figure 7: Frequencies of different words in the buckets on SO/Math.

The most relevant work to ours is a pairwise comparison based approach proposed by Liu et al. (2013) to estimate question difficulty levels in C-QA services. They have also demonstrated that a similar approach can be utilized to estimate user expertise levels (Liu et al., 2011). Yang et al. (2008) and Chen et al. (2013) have also proposed pairwise comparison based methods, for task difficulty estimation and rank aggregation in crowdsourcing settings. Our work differs from previous pairwise comparison based methods in that it further utilizes textual information, formalized as a manifold regularizer.

Manifold regularization is a geometrically motivated framework for machine learning, enforcing the learning model to be smooth w.r.t. the geometrical structure of data (Belkin et al., 2006). Within the framework, dimensionality reduction (Belkin and Niyogi, 2001; Cai et al., 2008) and semi-supervised learning (Zhou et al., 2004; Zhu and Lafferty, 2005) algorithms have been constructed. In dimensionality reduction, manifold regularization is utilized to guarantee that nearby points will have similar low-dimensional representations (Cai et al., 2008), while in semi-supervised learning it is utilized to ensure that nearby points will have similar labels (Zhou et al., 2004). In our work, we assume that nearby questions (in terms of textual similarities) will have similar difficulty levels.

Predicting reading difficulty levels of text is also a relevant problem (Collins-Thompson and Callan, 2004; Schwarm and Ostendorf, 2005). It is a key to automatically finding materials at appreciate reading levels for students, and also helps in personalized web search (Collins-Thompson et al., 2011). In the task of predicting reading difficulty levels, documents targeting different grade levels are taken as ground truth, which can be easily obtained from the web. However, there is no naturally annotated data for our QDE task on the web. Other related problems include query difficulty estimation for search engines (Carmel et al., 2006; Yom-Tov et al., 2005) and question difficulty estimation for automatic question answering systems (Lange et al., 2004). In these tasks, query/question difficulty is system-oriented and irrelevant with human knowledge, which is a different setting from ours.

7 Conclusion and Future Work

In this paper, we have proposed a novel method for estimating question difficulty levels in CQA services, called Regularized Competition Model (RCM). It takes fully advantage of questions' textual descriptions besides question-user comparisons, and thus can effectively deal with data sparsity and perform more accurate estimation. A K-Nearest Neighbor approach is further adopted to estimate difficulty levels of cold-start questions. Experiments on two publicly available data sets show that RCM performs significantly better than existing methods in the estimation task, for both well-resolved and cold-start questions, demonstrating the advantage of incorporating textual information. It is also observed that RCM might automatically measure the knowledge levels of words.

As future work, we plan to 1) Enhance the efficiency and scalability of RCM. The complexity analysis in Section 3.2 indicates that storing and processing the graph Laplacian is a bottleneck of RCM. We would like to investigate how to deal with the bottleneck, e.g., via parallel or distributed computing. 2) Apply RCM to non-technical domains. For non-technical domains such as the "news and events" category of Yahoo! Answers, there might be no strongly distinct notions of "experts" and "non-experts", and it might be more difficult to distinguish between "hard questions" and "easy questions". It is worthy investigating whether RCM still works on such domains.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (grant No. XDA06030200), the National Key Technology R&D Program (grant No. 2012BAH46B03), and the National Natural Science Foundation of China (grant No. 61272427).

References

Mark S. Ackerman and David W. McDonald. 1996. Answer garden 2: merging organizational memory with collaborative help. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work*, pages 97–105.

Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, pages 585–591.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434.

Stephen Boyd, Lin Xiao, and Almir Mutapcic. 2003. Subgradient methods. *Lecture Notes of EE392o, Stanford University*.

Deng Cai, Xiaofei He, Xiaoyun Wu, and Jiawei Han. 2008. Non-negative matrix factorization on manifold. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 63–72.

David Carmel, Elad Yom-Tov, Adam Darlow, and Dan Pelleg. 2006. What makes a query difficult? In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 390–397.

Olivier Chapelle. 2007. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178.

Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pages 193–202.

Fan RK. Chung. 1997. *Spectral Graph Theory*, volume 92.

Kenneth Church. 2011. How many multiword expressions do people know. In *Proceedings of the ACL-HLT Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 137–144.

Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. 1999. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR workshop on Recommender Systems*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Kevyn Collins-Thompson and James P. Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 193–200.

Kevyn Collins-Thompson, Paul N Bennett, Ryen W White, Sebastian de la Chica, and David Sontag. 2011. Personalizing web search results by reading level. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 403–412.

Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Zhicheng Dou, Ruihua Song, and Ji Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th International Conference on World Wide Web*, pages 581–590.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. Trueskill: a bayesian skill rating system. In *Advances in Neural Information Processing Systems*, pages 569–576.

Anna Huang. 2008. Similarity measures for text document clustering. In *Proceedings of the 6th New Zealand Computer Science Research Student Conference*, pages 49–56.

Rense Lange, Juan Moran, Warren R. Greiff, and Lisa Ferro. 2004. A probabilistic rasch analysis of question answering evaluations. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 65–72.

Baichuan Li and Irwin King. 2010. Routing questions to appropriate answerers in community question answering services. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1585–1588.

Jing Liu, Young-In Song, and Chin-Yew Lin. 2011. Competition-based user expertise score estimation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 425–434.

Jing Liu, Quan Wang, Chin-Yew Lin, and Hsiao-Wuen Hon. 2013. Question difficulty estimation in community question answering services. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 85–90.

Kevin Kyung Nam, Mark S. Ackerman, and Lada A. Adamic. 2009. Questions in, knowledge in?: a study of naver's question answering community. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 779–788.

Larry Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: bringing order to the web. *Technical Report, Stanford University*.

Joseph Lee Rodgers and W. Alan Nicewander. 1988. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260.

Sarah E. Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. 2004. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th International Conference on World Wide Web*, pages 675–684.

Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. 2010. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems*, pages 2424–2432.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier R Movellan. 2009. Whose vote should count more: optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems*, pages 2035–2043.

Jiang Yang, Lada Adamic, and Mark Ackerman. 2008. Competing to share expertise: the taskcn knowledge sharing community. In *Proceedings of the 2nd International AAAI Conference on Weblogs and Social Media*.

Elad Yom-Tov, Shai Fine, David Carmel, and Adam Darlow. 2005. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 512–519.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328.

Yanhong Zhou, Gao Cong, Bin Cui, Christian S. Jensen, and Junjie Yao. 2009. Routing questions to the right users in online communities. In *Proceedings of the 25th IEEE International Conference on Data Engineering*, pages 700–711.

Dengyong Zhou, John C Platt, Sumit Basu, and Yi Mao. 2012. Learning from the wisdom of crowds by minimax entropy. In *Advances in Neural Information Processing Systems*, pages 2204–2212.

Xiaojin Zhu and John Lafferty. 2005. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 1052–1059.