Vote Calibration in Community Question-Answering Systems*

Bee-Chung Chen
LinkedIn
Mountain View, CA
bchen@linkedin.com

Anirban Dasgupta
Yahoo! Labs
Sunnyvale, CA
anirban@yahoo-inc.com

Xuanhui Wang
Facebook
Menlo Park, CA
xuanhui@fb.com

Jie Yang
Google
Mountain View, CA
cnjieyang@gmail.com

ABSTRACT

User votes are important signals in community question-answering (CQA) systems. Many features of typical CQA systems, e.g. the best answer to a question, status of a user, are dependent on ratings or votes cast by the community. In a popular CQA site, Yahoo! Answers, users vote for the best answers to their questions and can also thumb up or down each individual answer. Prior work has shown that these votes provide useful predictors for content quality and user expertise, where each vote is usually assumed to carry the same weight as others. In this paper, we analyze a set of possible factors that indicate bias in user voting behavior – these factors encompass different gaming behavior, as well as other eccentricities, e.g., votes to show appreciation of answerers. These observations suggest that votes need to be calibrated before being used to identify good answers or experts. To address this problem, we propose a general machine learning framework to calibrate such votes. Through extensive experiments based on an editorially judged CQA dataset, we show that our supervised learning method of content-agnostic vote calibration can significantly improve the performance of answer ranking and expert ranking.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications – Data Mining.

Keywords: Reputation, user modeling, crowdsourcing, community question-answering.

1. INTRODUCTION

Community question answering (CQA) systems form the crowdsourced alternative to search engines for providing information. Popular and effective CQA systems such as Yahoo! Answers [1] provide an environment for people to share knowledge and experience, which complement search engines by allowing questions to be posed in natural language and be answered by human beings through active content (i.e., answers) creation, instead of mining existing Web pages. Since the basis of CQA systems is the wide participation, the hurdle for users to proffer answers is typically minimal. Hence, CQA systems are also prone to spam and other kinds of abuse. Commercial spam itself is a widespread problem, and one that can mostly be tackled by conventional machine learning. Other forms of abuse, e.g., low quality content, willfully wrong answers are difficult for machines to detect, and again crowdsourcing quality content identification is an engaging, scalable way of ensuring that high quality questions and answers are identified and the creators appropriately rewarded. Different CQA systems have thus implemented different kinds of voting mechanisms coupled with user reputation systems in order to surface high quality content and discourage undesired behavior.

In this paper, we focus on *votes* in CQA sites. Take Yahoo! Answers for example, which is heavily dependent on the best answer voting scheme. An asker (user who asks a question) can vote for the best answer to his/her question. If the asker does vote, the best answer is declared and the question is labeled as resolved. If the asker does not vote for the best answer within certain period of time, other users in the community can vote for the best answer to that question. The answer that receives the largest number of best-answer votes within a certain period of time is then declared as the best answer to this question, and the question is declared resolved. Other than best-answer votes, users can also demonstrate their quality or opinion preferences by casting thumb-up or thumb-down votes on each individual answer.

Although such voting mechanisms are designed with the intuition that they should be able to identify high quality answers and the corresponding answerers, there has been little study on the actual effectiveness of these systems in achieving this goal. On the other hand, many studies [1, 6, 15, 3, 8, 17, 10] have treated the user-voted best answers as the ground truth source of high quality answers and have developed models to predict whether an answer would be voted as the best answer based on features extracted from the answer, past activities of the answerer, etc. As a counter-point, other studies, e.g., [14, 9], report best answers not to be entirely high quality ones. In fact, as part of this work, we also observe that user-voted best answers do not always have high quality, possibly because of bias in users' voting behavior. A few examples of potential bias are:

- Users may vote more positively for their friends' answers.

*This work was conducted when all authors were affiliated with Yahoo!

[1] answers.yahoo.com

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

Even users who have interacted with each other in the CQA before might have a positive or negative bias depending on their past experience.

- Users may use votes to show their appreciation to answerers instead of trying to identify high quality content.
- Users who are trying to game the system in order to obtain high status can create multiple accounts and vote for one another.
- For questions about opinions, users tend to vote for the answers that share same opinions as theirs, but not necessarily of high quality.

Thus, instead of treating users' best-answer votes as the ground truth for answer quality, we ask trained human editors to judge answers based on a set of well-defined guidelines. Our first observation is that raw user votes have low correlation with editorial judgment. Based on this finding, we ask "*is it possible to calibrate the votes to mitigate the effects of potential bias, in order to increase the correlation between votes and editorially judged answer quality?*"

Contributions: We make the following contributions.

- We propose the novel problem of *vote calibration* in CQA systems. The basic idea is to associate every vote with a weight. Intuitively, votes from reputable voters who consistently vote for high quality answers should carry more weight, while votes susceptible to bias should carry less weight.
- In Section 5, based on exploratory data analysis we identify a variety of potential factors that possibly bias the votes.
- We develop a novel model for vote calibration based on supervised learning in Section 4. Intuitively, with each vote k we associate a feature vector $\boldsymbol{x}_k$ which includes features capturing potential bias. Then, the importance weight of vote k is a function of the weighted sum of feature values, i.e., $f(\boldsymbol{x}_k'\boldsymbol{\beta})$, where $\boldsymbol{\beta}$ is a vector of regression coefficients, $\boldsymbol{x}_k'\boldsymbol{\beta}$ is the inner product of $\boldsymbol{x}_k$ and $\boldsymbol{\beta}$, and $f(\cdot)$ is a monotone function. Based on a set of editorially labeled answers, $\boldsymbol{\beta}$ is determined by minimizing the error of using calibrated votes $f(\boldsymbol{x}_v'\boldsymbol{\beta})$ to predict the editorial labels.
- In Section 7, we experimentally show that the proposed vote calibration model outperforms a number of alternative methods for the answer ranking problem and the expert ranking problem. In particular, our model significantly outperforms the state-of-the-art SVM model [12] for expert ranking.

We note that since our focus is on vote calibration, we take a content-agnostic approach (i.e., no use of features extracted from answer content) similar to [12]. This makes our model applicable to votes on other types of content. Also, although we use Yahoo! Answers as the example CQA system throughout the paper, our model can be easily applied to other CQA systems. In fact we believe that our approach has the following useful properties – (i) it is robust to vote gaming, having incorporated a variety of such gaming behavioral features, and (ii) it provides a general framework for calibrating user votes in general crowdsourced rating platforms.

2. RELATED WORK

Although there have been many studies on CQA, to our knowledge, the problem of how to calibrate user votes to improve the correlation between votes and answer quality has not been systematically studied before. The related work can broadly be categorized into three threads.

Predicting user-voted best answers: There is a large body of work on building models to predict user-voted best answers (e.g., [1, 6, 15, 3, 8, 17, 10]). The assumption behind this line of work is that quality judgments of answers are expensive to obtain and it seems to be reasonable to use the readily available user-voted best answers as the target to predict. Different from this line of work, we do not try to predict user-voted best answers, but show that user-voted best answers may not be high quality ones because of potential bias in users' voting behavior. Our focus is on how to calibrate votes to minimize the discrepancy between votes and quality.

Predicting editorial judgments: Along with [2, 7, 16, 9, 14, 12], we define answer quality by a set of guidelines and obtain the ground-truth quality scores through an editorial process. For example, Agichtein et al. [2] used features extracted from text, user interaction graphs and votes to predict editorial judgments. Suryanto et al. [16] developed a number of graph-based methods to estimate user expertise, and evaluated these methods using editorially judged quality and relevance. Sakai et al. [9, 14] noticed bias in users' voting behavior that can cause user-voted best answers to be unsuitable for model evaluation, and proposed a number of evaluation methods based on editorial judgments. Similar to these studies, we also want to predict editorially labeled quality scores. Although user votes have commonly been used as features, calibration of each individual vote has not be studied.

Content-agnostic user expertise estimation: User expertise score estimation is an important use case of vote calibration. In the Section 7, we will show that using a simple average of the calibrated votes of a user as his/her expertise score can outperform the best methods reported in Liu et al. [12], where a variety of methods, such as PageRank-based method, HITS-based methods, competition-based methods, are empirically compared based on editorial judgments. Earlier work on user expertise is described in the following. Zhang et al. [18] proposed a PageRank-style algorithm for online forums. Jurczyk et al. [11] leveraged the HITS algorithm. Pal et al. [13] exploited the observation that experts usually answer questions that have not yet received good answers. Bouguessa et al. [4] developed a method to identify the number of users to be considered as experts based on a mixture of two Gamma distributions.

3. DATASET

Our dataset consists of editorial data and voting data.

3.1 Editorial Data

To determine the quality of an answer, we designed a set of guidelines together with a set of carefully selected examples for each quality grade and ask a human editor to judge it. We randomly sampled around 10,000 resolved questions in five popular categories (4.2K from each of "Sports" and "Entertainment and Music", 1.2k from each of "Business and Finance" and "Consumer Electronics", and 500 from "News and Events") from US Yahoo! Answers and, for each question, sampled three answers. These questions were selected by a diffusion process first starting with a random seed of 'active users – those who have answered at least 10 questions and provided 5 best answers; we then sampled resolved questions that these users answered, and then again from all answers to that question. For construction of features when modeling, we however looked at our entire dataset.

When judging an answer, an editor first read the question and answer, and then gave a quality *grade* to the answer according to a

Type	Editorial Quality Grade				#Ans
	Excellent	Good	Fair	Bad	
ABA	4.1%	22.9%	65.3%	7.8%	3,086
CBA	4.9%	24.5%	60.4%	10.3%	4,263
Non-Best	2.6%	18.2%	65.7%	13.5%	14,176
#Ans	698	4,337	13,899	2,591	21,525

Table 1: Best Answer vs. Editorial Quality

pre-determined set of editorial guidelines. In the interest of space, we only give a high-level description of the guidelines.

- *Excellent*: The answer provides a significant amount of useful information, fully answering the question.
- *Good*: The answer provides a partial but not entirely persuasive answer to the question. Minor typos and grammatical problems are acceptable.
- *Fair*: The answer fails to add much value to the knowledge base and is neither good nor bad — it's simply not sufficiently useful, interesting or entertaining to promote.
- *Bad*: The answer is either abusive, not germane to the question or so poor in substance or style.

Each answer was judged by one editor only. We ignored all the non-English questions and answers and also ignored all the answers that the editors could not make a confident judgement on. In total, we obtain 21,525 editorially judged answers on 7,372 questions (note that some questions do not have three answers).

User-voted best answers vs. quality: Table 1 shows the distribution of editorial quality grades for three types of answers: **ABA** denotes the asker-voted best answer, while **CBA** denotes the community-voted best answer through the majority rule. On Yahoo! Answers, CBA votes happen only when the asker of a question does not pick the best answer within certain period of time. **Non-Best** denote the answers that are not best answers. Notice that these three types of answers are disjoint. Each row of Table 1 sums up to one. The numbers of answers of the three types and the four editorial grades are reported in the last column and row. We make the following observations.

- The distribution of editorial grades for best answers are not very different from that of non-best answers, although the percentage of excellent or good best answers is slightly higher than that of excellent or good non-best answers. This indicates low correlation between users' best-answer votes and answer quality.
- A significant percentage (> 70%) of best answers are not even good (i.e., fair or bad). Anecdotally, users sometimes give best-answer votes to show appreciation, instead of identifying good answers. Also, a fair answer is in fact the best answer among other bad answers. These observations show the need to calibrate votes in order to use them to identify good content.
- Many non-best answers are actually good or excellent. This is due to the fact that only one answer can be selected as the best answer to each question even when there are many excellent ones.

Numeric quality scores: To build our vote calibration model, we give a numeric score to each editorial grade: Excellent = 1, Good = 0.5, Fair = 0, Bad = -0.5. We note that slight changes of assignments of numeric scores to editorial grades do not make a significant difference for the results as long as the order is preserved. They also do not affect the evaluation metrics in any way, since we use ranking metrics.

3.2 Voting Data

We collect the voting data in the following manner. We start with the set of answerers of the editorially judged answers and collect all of the resolved questions that were answered by this set of answerers in a one-year period. We then collect all of the answers to this set of questions, together with all of the votes on these answers. As a result, our voting data includes 1.3M questions, 7.0M answers, 0.5M asker best answer votes, 2.1M community best answer votes and 9.1M thumb up/down votes.

Uniqueness of our data: Voters' identities are important to calibrate their votes. To our knowledge, no public CQA dataset releases the identities of the voters. Thus, we were not able to evaluate our methods based on public datasets. Instead, we collect our own data, where each vote is associated with the anonymized IDs of the voter and recipient. Due to user confidentiality agreements, our data is not publicly available.

4. VOTE CALIBRATION MODEL

We present our vote calibration model in this section. Let $z_k \in \{+1, -1\}$ denote the value of vote k. The basic idea is simple. We associate each vote k with an importance weight w_k so that the calibrated value $z_k w_k$ of a vote would have higher correlation with answer quality. In order to determine these importance weights, we define a set of features to capture potential bias in users' voting behavior. Let $\boldsymbol{x}_k$ denote the feature vector of vote k. Then, we predict importance weight w_k using its corresponding features; i.e., $w_k = f(\boldsymbol{x}_k'\boldsymbol{\beta})$, where $\boldsymbol{\beta}$ is a vector of regression coefficients, $\boldsymbol{x}_k'\boldsymbol{\beta}$ is the inner product of the two column vectors ($\boldsymbol{x}_k'$ is the transpose of $\boldsymbol{x}_k$), and $f(\cdot)$ is a monotone function to calibrate the value $\boldsymbol{x}_k'\boldsymbol{\beta}$ (for example, to constrain w_k to be between 0 and 1). The parameter that needs to be determined is $\boldsymbol{\beta}$. Based on a set of answers labeled with ground-truth quality scores and the votes on these answers, we estimate $\boldsymbol{\beta}$ by minimizing the error of predicting quality scores using the calibrated votes.

In the rest of this section, we first introduce the notations and describe how to generate $+1$ and -1 vote values from the three kinds of votes on Yahoo! Answers, then explain how to predict quality scores using calibrated votes, and finally present a training algorithm for estimating the model parameters.

4.1 Notations

We use q to denote a question ID and u to denote an answerer ID. Since each user can provide at most one answer to each question in Yahoo! Answers, (q, u) form the ID of the answer to q provided by answerer u. We use v to denote a voter ID and t to denote a vote type. In Yahoo! Answers, there are three types of votes:

- **Asker votes:** These are the best answer votes by the asker of questions.
- **CBA votes:** These are the best answer votes by users in the community, called community best answer (CBA) votes.
- **Thumb votes:** These are the thumb-up and thumb-down votes on each individual answers.

Since a user can provide at most one vote for each answer and vote type, we use $z_{quv}^{(t)} \in \{+1, -1\}$ to denote the value of the type-t vote that voter v gives the answer provided by answerer u to question q. Let $\boldsymbol{x}_{quv}$ denote the feature vector associated with the vote that voter v gives answer (q, u). Notice that we did not add a superscript t on $\boldsymbol{x}_{quv}$ to reduce notational cumbersome; our method

can easily handle features defined specifically for a particular vote type. Let y_{qu} denote the ground-truth numeric quality score of answer (q, u). We discuss how we convert editorial grades into numeric scores in Section 3.1. Finally, we use $\mathcal{V}_{qu}^{(t)}$ to denote the set of all the voters who cast type-t vote on answer (q, u), and $\mathcal{Q}_u^{(t)}$ to denote the set of questions to which answerer u provides answers that receive type-t votes.

4.2 Vote Value before Calibration

In this paper, we focus on binary votes. Extension to numeric ratings is future work. We generate binary votes for the three vote types in the following way.

- **Asker votes:** If the asker of a question votes for an answer as the best answer, then that answer gets a $+1$ vote from the asker and all the other answers get -1 votes from the asker.
- **CBA votes:** Similar to asker votes, if a voter votes for an answer as the best answer, then that answer gets a $+1$ vote from the voter and all the other answers get -1 vote from the voter.
- **Thumb votes:** If a voter thumbs up an answer, then the answer gets a $+1$ vote from the voter. If a voter thumbs down an answer, then the answer gets a -1 vote from the voter.

4.3 Calibrated Vote Aggregation

We now describe how to predict the quality scores using calibrated votes. Intuitively, we predict the quality score y_{qu} of an answer by a weighted sum of (1) the average vote value of the answer (q, u) and (2) the average vote value of the answerer u. Because there are multiple types of votes, we compute average vote value for each type separately and then do a weighted average, where the weight on each type will be learned from a training dataset.

Calibrated vote value: Using the notations defined in Section 4.1, the value of calibrated type-t vote that voter v gives answer (q, u) is $z_{quv}^{(t)} f(\boldsymbol{x}'_{quv}\boldsymbol{\beta}_t)$, where $z_{quv}^{(t)}$ is the vote value before calibration, $\boldsymbol{x}'_{quv}$ is the feature vector associated with this vote and $\boldsymbol{\beta}_t$ is a vector of regression coefficients for type t. Notice that we have a type-specific $\boldsymbol{\beta}_t$, which gives us the flexibility to calibrate votes of different types differently. Although $f(\cdot)$ can be any differentiable monotone function, for concreteness, in this paper we use the sigmoid function: $f(x) = \frac{1}{1+e^{-x}}$. For comparison purposes, we also consider $f(\cdot)$ as the identity function; i.e., $f(x) = x$. We call the votes calibrated through the sigmoid function **sigmoid-calibrated votes** and call the votes calibrated through the identity function **linearly-calibrated votes**.

It is instructive to compare sigmoid calibration with linear calibration based on their definitions.

- Sigmoid calibration is based on a nonlinear function of the features, while linear calibration is based on a linear function of the features. Thus, sigmoid calibration has the potential to capture nonlinear characteristics in vote features.
- The range of the sigmoid function is between 0 and 1. Thus, the calibrated vote value is bounded by -1 and $+1$. The effect of a single vote is always bounded. This is not the case for linear calibration, where a vote may have unreasonably large value when some features take large values.

In Section 7, we will compare these two kinds of calibration experimentally and show the benefit of using the sigmoid function.

Average vote value of an answer: When there are multiple votes on an answer, we use the average calibrated vote value on the answer to predict its quality. The choice of average instead of the sum is prompted by the fact that the total number of votes received is often biased severely by the age of the answer, how prominently it is displayed etc. To make this average stable, we add some number γ_t of "pseudo votes" to the average. Specifically, using the notations defined in Section 4.1, the average type-t vote value of answer (q, u) is computed as

$$\textit{AnsValue}_{qu}^{(t)}(\boldsymbol{\beta}_t) = \frac{\mu_t\gamma_t + \sum_{v\in\mathcal{V}_{qu}^{(t)}} z_{quv}^{(t)} f(\boldsymbol{x}'_{quv}\boldsymbol{\beta}_t)}{\gamma_t + |\mathcal{V}_{qu}^{(t)}|}, \quad (1)$$

where μ_t and γ_t are two tuning parameters. In this paper, we set $\gamma_t = 1$ and $\mu_t =$ the average uncalibrated vote value over all type-t votes, which is computed by the sum of all type-t uncalibrated vote values divided by the total number of type-t votes in the system. Notice that μ_t is computed based on a large number of votes and should be a quite stable constant. If an answer receives no vote, its *AnsValue* would be exactly μ_t. Thus, μ_t can be thought of as the "prior mean" of *AnsValue*. If an answer receives a small number of votes, its *AnsValue* would still be close to μ_t, where how close they are depends on γ_t — the larger γ_t is, the closer they are. However, if an answer receives many votes, then its *AnsValue* would not be influenced much by μ_t.

Average vote value of an answerer/user: When an answer receives no vote or a small number of votes, we can also use the average value of the votes received by the answerer to predict the quality of the answer. Similar to Equation 1, the average type-t vote value of user u is computed as

$$\textit{UsrValue}_{u}^{(t)}(\boldsymbol{\beta}_t) = \frac{\mu_t\gamma_t + \sum_{q\in\mathcal{Q}_u^{(t)}}\sum_{v\in\mathcal{V}_{qu}^{(t)}} z_{quv}^{(t)} f(\boldsymbol{x}'_{quv}\boldsymbol{\beta}_t)}{\gamma_t + \sum_{q\in\mathcal{Q}_u^{(t)}} |\mathcal{V}_{qu}^{(t)}|}. \quad (2)$$

Notice that this formula is almost the same as Equation 1, except that here we sum over all of the type-t votes received by answerer u in the numerator and compute the total number of type-t votes received by answerer u in the denominator.

Quality prediction function: Given the answer-level and user-level average vote values of all types on an answer, its quality is predicted by a weighted sum of these average vote values. Specifically, we predict the quality score y_{qu} of answer (q, u) by

$$\textit{Score}_{qu}(b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = b + \sum_t \alpha_{t,0}\, \textit{AnsValue}_{qu}^{(t)}(\boldsymbol{\beta}_t) + \alpha_{t,1}\, \textit{UsrValue}_{u}^{(t)}(\boldsymbol{\beta}_t),$$

where b is a bias term, $\boldsymbol{\alpha} = \{\alpha_{t,0}, \alpha_{t,1}\}_{\forall t}$ consists of the weights of different components, and $\boldsymbol{\beta} = \{\boldsymbol{\beta}_t\}_{\forall t}$ consists of all of the regression-coefficient vectors. Since answer quality is predicted by aggregating calibrated votes, we call this model **calibrated vote aggregation** model. When sigmoid calibration is used, we call it **sigmoid-calibrated vote aggregation (SVA)** model. When linear calibration is used, we call it **linearly-calibrated vote aggregation (LVA)** model. Notice that b, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the model parameters that need to be learned from a training dataset.

User expertise: To estimate the expertise score of a user, we simply use the average calibrated vote values of that user. Specifically, the expertise score of user u is $\textit{Expertise}_u(b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = b + \sum_t \alpha_{t,1}\, \textit{UsrValue}_{u}^{(t)}(\boldsymbol{\beta}_t)$. Since this is a special case of the full model, we do not specifically discuss this special case.

4.4 Training Algorithm

Given a training dataset consisting a set of answers together with ground-truth quality scores and votes on them, we determine the

model parameters $\Theta = (b, \boldsymbol{\alpha}, \boldsymbol{\beta})$ by minimizing the following loss function

$$\ell(\Theta) = \tfrac{1}{2}\sum_{qu}(y_{qu} - \mathit{Score}_{qu}(\Theta))^2 + \tfrac{\lambda_1}{2}\|b\|^2 + \tfrac{\lambda_2}{2}\|\boldsymbol{\alpha}\|^2 + \tfrac{\lambda_3}{2}\|\boldsymbol{\beta}\|^2,$$

where $\|\boldsymbol{\alpha}\|^2$ denotes the sum of squares of individual elements in $\boldsymbol{\alpha}$, which serves as a regularizer to prevent potential overfitting. λ_1, λ_2 and λ_3 are the regularization weights of each regularizer. We minimize this loss function using gradient descent.

Gradients: The formulas of the gradients are as follows. Let $e_{qu}(\Theta) = y_{qu} - \mathit{Score}_{qu}(\Theta)$ denote the difference between the ground-truth quality score and the predicted quality score.

$$\tfrac{d}{db}\ell(\Theta) = -\sum_{qu} e_{qu}(\Theta) + \lambda_1 b$$

$$\tfrac{d}{d\alpha_{t,0}}\ell(\Theta) = -\sum_{qu} e_{qu}(\Theta)\, \mathit{AnsValue}_{qu}^{(t)}(\boldsymbol{\beta}_t) + \lambda_2\, \alpha_{t,0}$$

$$\tfrac{d}{d\alpha_{t,1}}\ell(\Theta) = -\sum_{qu} e_{qu}(\Theta)\, \mathit{UsrValue}_{u}^{(t)}(\boldsymbol{\beta}_t) + \lambda_2\, \alpha_{t,1}$$

$$\tfrac{d}{d\boldsymbol{\beta}_t}\ell(\Theta) = -\sum_{qu} e_{qu}(\Theta)\big[\alpha_{t,0}\, \tfrac{d}{d\boldsymbol{\beta}_t}\mathit{AnsValue}_{qu}^{(t)}(\boldsymbol{\beta}_t) + \alpha_{t,1}\, \tfrac{d}{d\boldsymbol{\beta}_t}\mathit{UsrValue}_{u}^{(t)}(\boldsymbol{\beta}_t)\big] + \lambda_2\, \boldsymbol{\beta}_t,$$

where

$$\tfrac{d}{d\boldsymbol{\beta}_t}\mathit{AnsValue}_{qu}^{(t)}(\boldsymbol{\beta}_t) = \frac{\sum_{v\in\mathcal{V}_{qu}^{(t)}} z_{quv}^{(t)} f'(\boldsymbol{x}'_{quv}\boldsymbol{\beta}_t)\, \boldsymbol{x}_{quv}}{\gamma_t + |\mathcal{V}_{qu}^{(t)}|}$$

$$\tfrac{d}{d\boldsymbol{\beta}_t}\mathit{UsrValue}_{u}^{(t)}(\boldsymbol{\beta}_t) = \frac{\sum_{q\in\mathcal{Q}_u^{(t)}}\sum_{v\in\mathcal{V}_{qu}^{(t)}} z_{quv}^{(t)} f'(\boldsymbol{x}'_{quv}\boldsymbol{\beta}_t)\, \boldsymbol{x}_{quv}}{\gamma_t + \sum_{q\in\mathcal{Q}_u^{(t)}}|\mathcal{V}_{qu}^{(t)}|}.$$

If $f(\cdot)$ is the sigmoid function in sigmoid calibration, its derivative $f'(x)$ is $f(x)(1-f(x))$. For linear calibration, $f'(x) = 1$ always. However, it can be shown that linear calibration can be transformed into a regular linear regression problem. Thus, we solve linear calibration by using a standard linear regression package.

Loss minimization: There are many gradient descent algorithms, which can be easily applied to minimize our loss function. In our implementation, we use L-BFGS [5].

5. EXPLORATORY ANALYSIS

Before we define the features to be used in our calibration model, we discuss a range of potential indicators of voting bias in this section. In the interest of space, we mainly use community best-answer (CBA) votes to show our analysis. Many of the insights also apply to other types of votes.

Characteristics of CBA votes: First, the importance of CBA votes is seen in Figure 1, which plots the fraction of questions resolved by CBA votes as a function of the number of answers to the question. On average around 61% of the total number of questions in our corpus get resolved by CBA votes, and this fraction reaches its peak for questions with smaller number of answers (being near 70% for questions with less than three answers). One possible explanation for the U-shaped curve of Figure 1 could be that the number of answers being too small indicates that the question is probably not interesting, likely posed by a casual user who is less inclined to give a best answer vote herself – and thus the community has to step in.

Furthermore, CBA votes are rarely unanimous. Figure 2 shows how many answers to a question on average receive at least one CBA vote as a function of the number of answers to the question.

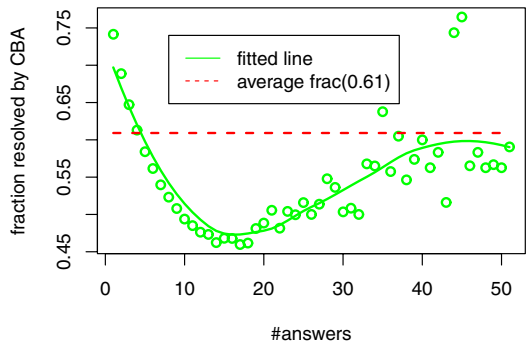


Figure 1: Fraction of questions resolved by CBA votes

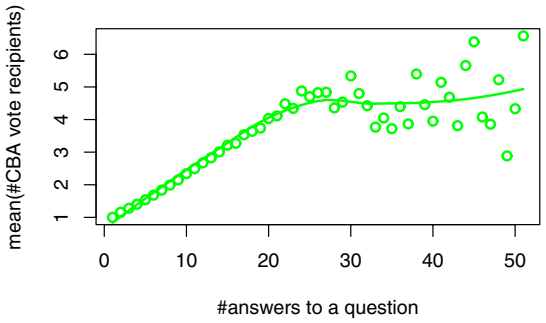


Figure 2: Number of answers receiving CBA votes

The y-axis shows the number of answers that receive at least one CBA vote, averaged over all questions that have the same number of answers. There is a clear linear trend up for questions with ≤ 30 answers. Beyond that, the trend is almost flat and noisier. Sparsity of data is likely the main reason behind the noise. The flatness of the trend can be possibly explained by the fact that a community member looking to vote an answer as CBA is likely only to look at the top few answers in the page; some answers can also be submitted after the CBA deadline passes.

In the following, we discuss our observations about some potential sources of bias that we see in our dataset. For each of these sources, it is important to note that we are not claiming that this bias immediately implies that the corresponding CBA votes provide no signal to predict answer quality. Our goal here is to identify a variety of indicators of potential bias, so that our model can use them to calibrate votes.

Self Voting: Figure 3 shows the extent of self voting (where a user votes for his/her own answers) in our dataset. Self votes contribute to 33% of the total CBA votes, and when we consider users who cast at least 20 votes, the percentage of self votes goes even above 40%. Self votes are certainly unlikely to represent any quality judgment, they can at most be taken as a signal of the user's familiarity with the voting mechanism of the CQA site, which in itself could provide a useful signal.

Vote Spread and Reciprocity: We define *in/out spread* of a user u as the ratio between the number of distinct users who cast/receive a vote to/from u and the number of votes that go into/out of u, which measures the spread of u's incoming/outgoing votes over different users. This measure could capture potential biased behavior in which users are either receiving most of their votes from a small number of users, or are themselves casting the votes in a concentrated manner. This phenomenon might be due to gaming, a user's

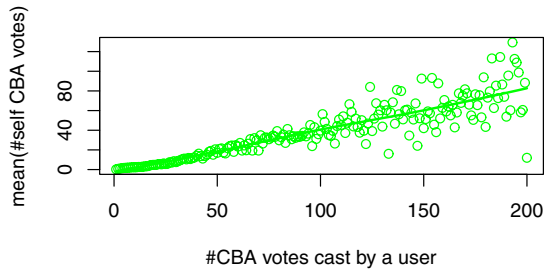


Figure 3: Number of self CBA votes by a user, along with fitted curve showing the trend.

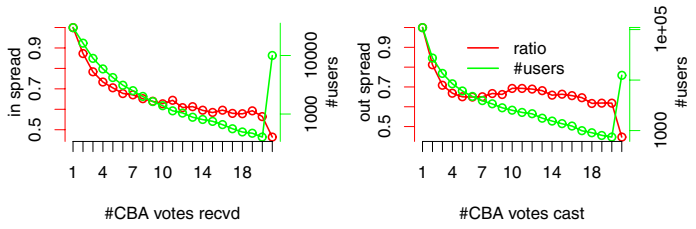


Figure 4: (a) In Spread and (b) Out Spread of CBA votes. The right end point of the x-axis includes all users with > 20 votes

narrow interests, or a scenario where only a small number users provide high quality answers. We report *out spread* and *in spread* in the Figure 4. The red curve together with the left hand y-axis represents in/out spread as a function of the number of votes that go into/out of a user. The green curve with the right hand y-axis shows, in log-scale, the number of users who received/cast a certain number of CBA votes. The spread decreases with increasing number of CBA votes cast/received. It is lowest on average for users who have cast/received more than 20 votes – which could indicate gaming by a small number of active users.

One interesting user-user interaction, almost a "quid pro quo" action, is when two users cast CBA votes on each other thereby creating a reciprocal relation between the two. We call a vote a *reciprocal* vote if the recipient also vote back for an answer authored by the voter. We define *in/out reciprocity* of user u as the fraction of incoming/outgoing votes that are reciprocal. Figure 5 shows in/out reciprocity as a function of the total number of incoming and outgoing CBA votes of a user. Both the curves are concave, and stabilize around 0.46 for users with large number of votes – again indicating potential gaming by heavy users. The number of users involved in such a reciprocal voting pattern is also seen in Figure 6 – over 20K users in our dataset are involved in at least 10 reciprocal votes. Considering this pattern as a feature when calibrating CBA votes certainly seems advisable.

Other Interaction Bias: In the remaining part of the section, we investigate whether CBA votes are independent of previous interaction between the voter and recipient. Our strategy is to use a hypothesis testing framework to analyze whether the null hypothesis — that CBA voting is independent of previous interaction — holds true individually for different interaction types. The different types of interaction we consider are the following: Whether one user answered another's question, whether they thumbed up/down each other, and whether they gave CBA/ABA votes to each other (recall that ABA denote asker best-answer). We represent each of these by a formal relation as follows. Let t be a time point. ANS_t denotes the set of all pairs (u, v) of users such that u answered a question posed by v before time t. TU_t denotes the set

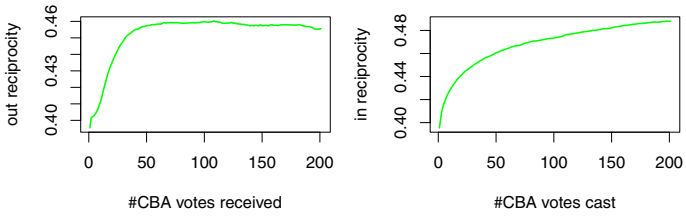


Figure 5: (a) Out Reciprocity and (b) In Reciprocity of CBA votes.

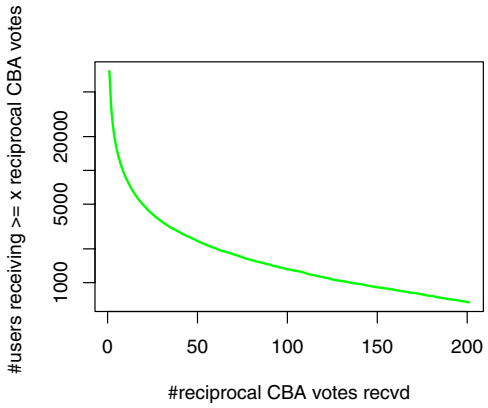


Figure 6: Number of users having at least x reciprocal votes

of the pairs (u, v) such that u thumbed up an answer by v before time t. TD_t, CBA_t, ABA_t are similarly defined based on thumb-down votes, CBA votes and ABA votes that user u gives user v. Our goal is to investigate whether a CBA vote that v gives to u at time $t + 1$ is independent of whether $(u, v) \in R_t$ for $R \in \{\mathsf{ANS}, \mathsf{TU}, \mathsf{TD}, \mathsf{CBA}, \mathsf{ABA}\}$. Before describing our observation, we briefly describe our methodology.

Chi-squared Statistic: Let $CBA(u)$ denote the set of recipients of the CBA votes cast by user u. Let R be any of the relation described above. Then, the chi-squared statistic for measuring independence of R and $CBA(u)$ can be computed based on the following counts, show in Table 2. Let $N = a + b + c + d$. Then, the chi-squared statistic $X^2(CBA, R)$ is defined as

$$X^2(CBA, R) = \frac{N(ad - bc)^2}{(a + b)(c + d)(a + c)(b + d)}$$

For measuring the effect of previous interaction on a CBA vote, we will compute this statistic based on the dynamic interaction stream. We go through the list of user actions performed in increasing order of time, and maintain the current set of user pairs who are in relation R. At any point, when we are processing a CBA vote, we use the relation R constructed up until this point to find which of the above cells we should increment the counts of. The X^2 statistic is computed after the entire stream is processed – this is then used for the test.

Randomization Tests: In order to get the confidence interval of the

	$(u, v) \in R$	$(u, v) \notin R$
$v \in CBA(u)$	a	b
$v \notin CBA(u)$	c	d

Table 2: Table for chi-squared statistic.

	empirical ($\hat{X}$)	max random	min random	$\frac{\hat{X}-\mu}{\sigma}$
TU	66941.65	35718.70	35263.40	261.76
TD	5830.44	7369.88	7046.85	-18.81
CBA	55515.30	29976.85	29360.74	168.97
ANS	13391.79	8677.06	8496.51	103.60
ABA	9616.96	5921.43	5755.67	82.56

Table 3: Effect of previous interaction (row indicates type) on CBA.

above X^2 value under the null hypothesis, we resort to randomization of the data. The way the data is randomized is as follows – for each question, we first decide on a list of candidate answers who are eligible to receive CBA votes. We select these to be the answerers who have received at least one CBA vote for this question. The randomized version of the X^2 statistic is then computed by the following process – we again traverse the list of interactions in increasing order of time. Each CBA vote is now replaced by a random choice from the list of candidates to this question – the updates to the counts in Table 2 are now made according to this random choice. Finally, the X^2 statistic is computed at the end.

Effect of Interaction: The above randomization was performed 20 times. In Table 3, we present the statistics obtained. The four columns present the X^2 statistic obtained on our data, the maximum and minimum over the 20 randomizations, and the normalized value of the statistic based on the empirical mean and variance. Based on the 20 randomizations, the null hypothesis of the CBA being independent of the relation was rejected with probability 0.1 for each of the relation TU, TD, CBA, ANS and ABA. Note that this only means that there is some correlation between CBA votes and past interaction of any of the above types – it does not establish a direct bias. Such correlation could, for instance, occur if active users are also experts. Nevertheless, this result suggest past interaction could be useful features for vote calibration.

6. FEATURES

Based on the exploratory analysis, in this section we define the features that we use to calibrate votes. We consider two types of features: voter features, which try to capture a user's voting behavior, and relation features, which try to capture potential bias associated with the relation between the voter and the answerer (vote recipient).

Definitions: We say that user v gives user u a vote if v gives a vote to an answer authored by u. In this relation, v is the voter and u is the answerer. A vote from v to u is a *reciprocal* vote if u has given v a vote before. A vote from user v on an answer is a *majority* vote if the answer receives at least one additional vote and has the largest number of votes among other answers to the same question.

Notations: We will use the following notations.

- *nVotes*(v,u) = # of votes from user v to user u.
- *nVotesBy*(v) = # of votes cast by user v.
- *nVotesTo*(u) = # of votes given to user u.
- *nUsersBy*(v) = # of unique users receiving votes cast by v.
- *nUsersTo*(u) = # of unique users who give votes to user v.
- *nRecVotes*(v) = # of *reciprocal* votes cast by user v.
- *nMajVotes*(v) = # of *majority* votes cast by user v.
- *nVotersQ*(q) = # voters voting for any answer to question q.
- *ratio*$(x,y,n) = \frac{x+\mu n}{y+n}$ is a smoothed ratio between two counts x and y, where n is a pseudo count to stabilize the ratio and μ denote the average of x/y over all of the (x,y) pairs for which we want to compute the ratio with $y \geq n$.

Voter features: We define the following features for voter v, for each vote type separately (when appropriate).

- **Vote volume**: *nVotesBy*(v), *nVotesTo*(v), *nUsersBy*(v) and *nUsersTo*(v).
- **Vote spread**: *ratio*(*nUsersBy*(v), *nVotesBy*(v), 3), which measures the spread of the votes of v over different users.
- **Vote reciprocity**: *ratio*(*nRecVotes*(v), *nVotesBy*(v), 3), which measures how often v casts reciprocal votes. We also include the raw count *nRecVotes*(v) as a feature.
- **Self vote**: % of times v votes for his/her own answers.
- **Majority vote**: *ratio*(*nMajVotes*(v), *nVotesBy*(v), 3) and the raw count *nMajVotes*(v).

Relation features: We define the following features for each pair (v,u) of users, where v is the voter and u is the answerer, for each vote type separately.

- **Voting probability**: *ratio*(*nVotes*(v,u), *nVotesBy*(v), 3), which measures the probability that v would vote for u.
- **Receiving probability**: *ratio*(*nVotes*(v,u), *nVotesTo*(u), 3), measuring the probability that u would receive votes from v.
- **Vote-back probability**: *ratio*(*nVotes*(u,v), *nVotesBy*(u), 3), which measures the probability that u would vote back to v.
- **Voter contribution**: Average of $\frac{1}{nVotersQ(q)}$ over all of the questions on which v vote for u, where $\frac{1}{nVotersQ(q)}$ represents the contribution of v among all the voters who vote for any answer to question q.

Feature transformation: For each of the features C that are counts, we consider $\log(1+C)$ as an additional feature. For ratio features R, we also include a quadratic term R^2.

Intercept: Same as in linear regression, we always include a feature, called intercept, which value is always 1.

7. EXPERIMENTAL RESULTS

With the above features, we evaluate our methods based on our editorially labeled dataset described in Section 3.

7.1 Evaluation Setup

We evaluate our models at two different levels:

- **User-level expert ranking**: User expertise score estimation is a much studied topic [12, 18, 11]. As described in Section 4.3, our models can be used to estimate such expertise scores for each user. Here, we evaluate how well we rank users based on the predicted user-level scores.
- **Answer ranking**: Our models can be used to predict the quality of each individual answer based on calibrated votes on the answer (if any) and answerer. Here, we evaluate how well we rank answers based on the predicted answer-level scores.

Note that our focus is on vote calibration. Thus, we take a content-agnostic approach to the above two ranking problems. Since the most related work [12] on content-agnostic methods is in the expert ranking setting, this forms our main comparison setting. Next, we compare variants of our methods based on both user-level and answer-level score estimation.

An ideal way to evaluate user-level scores is to collect a ground-truth data which has the expertise grade for each user. As noted by a few research studies, such as [12], obtaining such a dataset by human judgment is very costly because all the activities of a user need to be examined in order to assess her expertise. An alternative approach is to use certain heuristics to define experts (e.g., users with the "top contributor" badge on Yahoo! Answers), which are

also prone to have biases or be abused. Thus in our paper, we take an indirect approach similar to [12] in using the editorially judged (question, answer) pairs to evaluate user-level expertise. The evaluation hypothesis is that the answers provided by a user with higher expertise should have higher quality. Thus, we rank all the answers to a question using the user-level expertise scores of the answers and use the standard ranking metrics for evaluation.

Data: We try to predict the editorial quality scores on answers using either user-level or answer-level scores. We obtain 21,525 editorially judged answers as described in Section 3.1. All of the methods to be compared have access to the voting data described in Section 3.2.

Cross-validation: We use 10-fold cross validation to evaluate our models. We randomly split users into 10 groups. We use the editorially judged answers authored by users in 9 groups to train a model and then use this model to predict the user-level scores of the users in the remaining group and the answer-level scores for the answers they authored. This process is repeated 10 times to obtain the predicted scores for all the users and answers. It is important to note that the score of each user is predicted by a model that does not use any label information about that user.

Evaluation Metrics: In Yahoo! Answers, a resolved question has 3.6 answers on average. When creating our editorial dataset, we sampled at most 3 answers for each question. Thus, the ranked list per question is short in our evaluation and the ability to distinguish the performance of different methods is limited. In order to overcome this limitation, we proposed three types of evaluation schemes as follows.

- **Question Level**: For each question, we rank the $\leq$ 3 answers to this question and evaluate using NDCG metrics. The NDCG metrics are sensitive to the highly ranked answers and thus are more suitable for question level evaluation. The NDCG@k is defined to be

$$\text{NDCG@}k = \frac{1}{Z_k}\sum_{i=1}^{k}\frac{G_i}{\log_2(i+1)}$$

  where G_i is computed based on the label of the i-th ranked answer and Z_k represents a normalization factor to guarantee that the NDCG@k for the perfect ranking (among all the permutations) is 1. In our paper, we set $G_i = 3, 1, 0, 0$ for Excellent, Good, Fair, Bad.

- **Global Level**: In the global level evaluation, we pool all the (question, answer) pairs together and rank all of them according to a model. After obtaining this ranked list f, we compare it to the ground-truth ranked list g, ranked according to editorial grades, using the Kendall τ rank correlation:

$$\frac{|\{(u,v): u \succ_f v \& u \succ_g v\}| - |\{(u,v): u \succ_f v \& u \prec_g v\}|}{\frac{1}{2}n(n-1)}$$

  which combines both concordant and discordant pairs in the evaluation. It is important to note that the editorial grade of an answer is on an *absolute scale* across all answers, instead of being relative to other answers to the same question. Thus, it makes sense to pool all (question, answer) pairs together in to a single ranked list. A positive number means that two rank have a positive correlation. The higher the number, the better.

- **Category Level**: Between the above two levels, we have the category level evaluation. In Yahoo! Answers, each question belongs to a category. The category level evaluation is to pool all the questions belonging to the corresponding category together. Each method provides a ranked list of (question, answer) pairs for each category. Here, the ranked lists are much longer than per-question ranked lists. We thus adopt the commonly used Mean Average Precision (MAP) and Precision at 10 (P@10) for evaluation by treating Excellent and Good as relevant but Fair and Bad as irrelevant labels.

Methods for Comparison: The variants of our methods to be compared include (1) **SVA.Full**: SVA model with both voter and relation features; (2) **SVA.Voter**: SVA model with only voter features; (3) **LVA.Full**: LVA model with both voter and relation features; (4) **LVA.Voter**: LVA model with only voter features; (5) **NoCalib**: SVA model that only uses the intercept feature (which value is always one, thus no calibration for individual votes; however, different vote types are weighted differently). All these methods can be applied for either answer ranking or user-level expert ranking. We compare with the following baseline methods:

- Smoothed Best Answer Rate (**BAR**) estimates the probability that an answer provided by a user u would be a best answer. Let $\text{Ans}(u)$ be the number of answers that u provided and $\text{BA}(u)$ be the number of the best answers that u received in our voting data. We estimate best answer rate by

$$\text{BAR}(u) = \frac{\text{BA}(u) + \mu\text{BAR}_{\text{avg}}}{\text{Ans}(u) + \mu}$$

  where BAR_{avg} is the probability of a randomly chose answer to be a best answer and μ is the smoothing parameter.

- Smoothed Competition Win Rate (**CWR**) is based on the competition defined in [12]. In this model, an answerer whose answer is the best answer for a question win a competition over each of other answerers to the same question, as well as the asker. Thus, we have (winner, loser) pairs from our data. The CWR is the rate that a user is the winner in all the pairs that the user appeared. Similarly, we also have a smoothing parameter μ for this method.

- **SVM** is the model proposed in [12]. It computes a single score for each user based on the competition pairs. Let the score vector be $\mathbf{w}$. The problem is formulated in the SVM format

$$\begin{aligned}&\text{minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum \xi_{ij}\\ &\text{s.t. } w_i - w_j \geq 1 - \xi_{ij}, \xi_{ij} \geq 0 \text{ for } i \succ j\end{aligned}$$

  where $i \succ j$ means user i wins over user j in a competition.

- **BAR+BA**: All the above methods are for user-level scores, which can be applied on expert ranking. For the answer ranking evaluation, we combine BAR with a binary score indicating whether the answer is the best answer based on a weighted sum of the two.

7.2 Experiment Results

7.2.1 Overall comparison

We compare our methods with all the baselines in Table 4. In this table, we show the three types of evaluation setting for both user-level expert ranking and answer-level ranking. All the parameters, μ for BAR and CWR, C for SVM, and a coefficient for BAR+BA, are tuned and we report their optimal results. In our methods, all the regularization parameters are set to 1.

Significance test: We use the paired t-test to assess the significance of the performance difference in NDCG@k, P@10 and MAP between two methods. Each query (i.e., question or category) gives a point in the significance test. For Kendall's rank correlation, each

Table 4: Overall comparison. Symbol $^+$ indicates significant improvement over BAR, BAR+BA, CWR and SVM; * indicates significant improvement over NoCalib (p-value < 0.05).

		Expert Ranking					Answer Ranking		
	Metric	BAR	CWR	SVM	NoCalib	SVA.Full	BAR+BA	NoCalib	SVA.Full
Question Level	NDCG@1	0.675	0.675	0.676	0.694$^+$	**0.701**$^+$	0.690	0.701$^+$	**0.712**$^{+*}$
	NDCG@2	0.748	0.752	0.752	0.760$^+$	**0.765**$^+$	0.760	0.766$^+$	**0.769**$^+$
	NDCG@3	0.800	0.801	0.802	0.808$^+$	**0.811**$^+$	0.807	0.811$^+$	**0.814**$^{+*}$
Category Level	P@10	0.336	0.315	0.307	0.383$^+$	**0.407**$^+$	0.365	0.475$^+$	**0.501**$^+$
	MAP	0.366	0.353	0.354	0.386$^+$	**0.396**$^{+*}$	0.381	0.425$^+$	**0.444**$^{+*}$
Global	Kendall	0.147	0.144	0.143	0.182+	**0.203**+*	0.152	0.185+	**0.209**+*

method generates a single number which makes t-test inapplicable. In this case, we use bootstrap sampling to construct the distribution of Kendall's correlation and assess significance based on that.

We analyze our results from the following perspectives:

Supervised vs Unsupervised: For either expert ranking or answer ranking methods, we have supervised methods (NoCalib and SVA.Full) and unsupervised methods (all others). In Table 4, the best results are highlighted. SVA.Full outperforms all other methods by a large margin. For example, for expert ranking, SVA.Full improves over BAR by 3.8% on NDCG@1, 8.0% on MAP and 38% on Kendall; for answer ranking, SVA.Full improve over BAR+BA by 3.2% on NDCG@1, 16% on MAP and 37% on Kendall. All these improvement are statistically significant. Among the unsupervised methods, BAR is slightly worse than SVM and CWR on NDCG metrics but consistently better on MAP and Kendall metrics. This shows that BAR is a robust baseline to for expert ranking. Also, NoCalib (which uses three types of votes) consistently outperforms BAR. This shows that in CQA systems, considering all types of voting information is helpful, which has been unfortunately ignored in previous work. Our methods are effective to combine all the votes to improve the utility of CQA.

Effect of Calibration: We analyze the effect of calibration in Table 4 by comparing NoCalib and SVA.Full. From this table, we can see that SVA.Full is significantly better than NoCalib on most of the metrics, especially for answer ranking. For example, the improvement is 1.5% for NDCG@1 and 4.5% on MAP for the answer ranking. All these improvement are statistically significant at p-value < 0.05. Together with Section 5, this result confirms the existence of bias in the current voting data and shows that vote calibration using our methods is effective.

7.2.2 Comparison of Calibration Models

In this section, we compare different models of vote calibration and different sets of features. In Figure 7(a), we compare variants of SVA and LVA together with the NoCalib using the category-level metrics on two sets of users: a set with all users and a set with heavy users who have more than 50 answers in our voting data (there are 11 such users). We again see SVA and LVA outperform NoCalib. Furthermore, we have the following observations:

SVA vs LVA: From both figures, we can that SVA (SVA.Full and SVA.Voter) is consistently better than LVA (LVA.Full and LVA.Voter) on both expert ranking and answer ranking. The difference is more significant in the answer ranking setting. This shows that proper normalization is important for the vote calibration in our models.

Full vs Voter: By comparing different set of features, full and voter, we see that the models with full features are also better than those with only voter features. This shows that the relation features are useful in vote calibration.

7.2.3 Impact on Heavy Users

We conducted a stratified study on heavy users by selecting them according to their level of activities in our dataset. Specifically, we set a threshold t and select only those users who have at least t answers in our voting data set. We vary t from 2 to 50 and plot the results in Figure 8. In this figure, the larger t is, the more active the set of users are. Clearly, both figures show that accuracy increases as user activity level increases. This makes intuitive sense because we have more information about heavy users and thus their expertise scores and the quality of the answers they provide can be better estimated. On both expert ranking and answer ranking, our models are consistently better than NoCalib and the relative order of different methods stay the same. Furthermore, for expert ranking, we see a larger margin between models with calibration and NoCalib. This suggests that calibration is more important for heavy users.

7.2.4 Feature Importance and Tuning Parameters

Feature importance: We now investigate the importance of different features defined in Section 6. To assess the importance of a set of features, we use that set of features alone to build our vote calibration model and compute the Kendall's correlation as the importance score of that set of features. The top 5 sets of features in the order of their importance are: Majority vote, vote spread, self vote, vote reciprocity and voting probability.

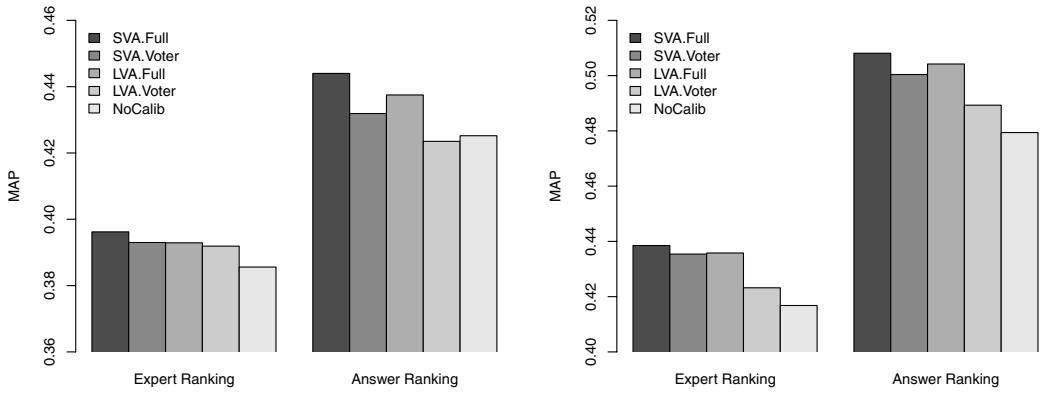
Tuning parameters: Recall that our model has two sets of tuning parameters γ_t and λ_k. To prevent overfitting, all the above results are based on simply setting $\gamma_t = 1$ and $\lambda_k = 1$ without tuning. Here, we investigate the sensitivity of SVA.Full to these tuning parameters. We use Kendall's correlation as the evaluation metric.

Setting	0.01	0.1	1	10
Fix $\gamma_t = 1$, vary λ_k	0.209	0.209	0.209	0.205
Fix $\lambda_k = 1$, vary γ_k	0.208	0.209	0.209	0.200

As can be seen, our model is not sensitive to tuning parameter settings.

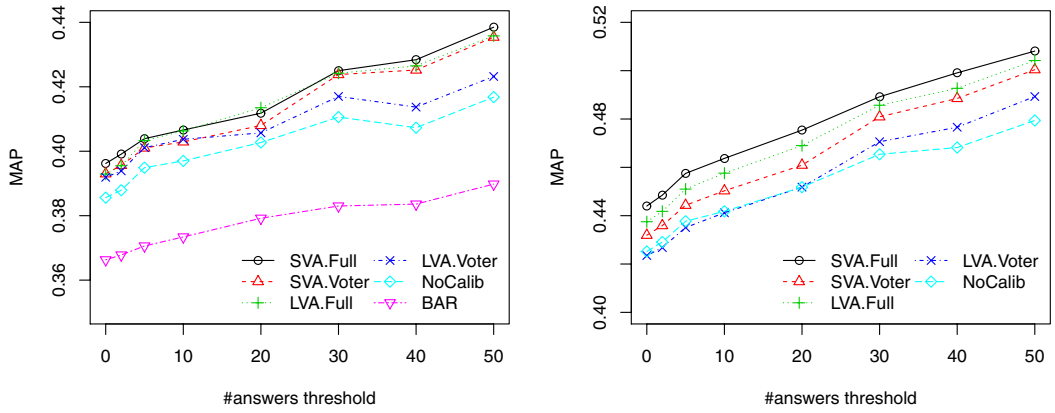
8. CONCLUSION

In this paper, we introduce vote calibration to CQA systems. By analyzing potential bias in users' voting behavior we propose a set of features to capture such bias. Using supervised models we show that our calibrated models are better than the non-calibrated versions on both user expertise and answer quality estimation.



(a) All users (b) Heavy users with ≥ 50 answers

Figure 7: Comparison of Calibration Models



(a) Expert ranking (b) Answer ranking

Figure 8: Results on heavy users

9. REFERENCES

[1] L. A. Adamic, J. Zhang, E. Bakshy, and M. S. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In *WWW*, 2008.

[2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *WSDM*, 2008.

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

[4] M. Bouguessa, B. Dumoulin, and S. Wang. Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In *KDD*, 2008.

[5] R. H. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 1995.

[6] B. Dom and D. Paranjpe. A bayesian technique for estimating the credibility of question answerers. In *SDM*, 2008.

[7] F. M. Harper, D. Raban, S. Rafaeli, and J. A. Konstan. Predictors of answer quality in online q&a sites. In *CHI*, 2008.

[8] L. Hong, Z. Yang, and B. D. Davison. Incorporating participant reputation in community-driven question answering systems. In *SIN*, 2009.

[9] D. Ishikawa, N. Kando, and T. Sakai. Ntcir-8 community qa pilot task. *The 8th NTCIR Workshop*, 2010.

[10] B. John, C. A. Yeow-Kuan, and G. D. Hoe-Lian. What makes a high quality user-generated answer? *IEEE Internet Computing*, 2010.

[11] P. Jurczyk and E. Agichtein. Discovering authorities in question answer communities by using link analysis. In *CIKM*, 2007.

[12] J. Liu, Y.-I. Song, and C.-Y. Lin. Competition-based user expertise score estimation. In *SIGIR*, 2011.

[13] A. Pal and J. A. Konstan. Expert identification in community question answering: exploring question selection bias. In *CIKM*, 2010.

[14] T. Sakai, D. Ishikawa, N. Kando, Y. Seki, K. Kuriyama, and C.-Y. Lin. Using graded-relevance metrics for evaluating community qa answer selection. In *WSDM*, 2011.

[15] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers on large online qa collections. In *ACL-HLT*, 2008.

[16] M. A. Suryanto, E. P. Lim, A. Sun, and R. H. L. Chiang. Quality-aware collaborative question answering: methods and evaluation. In *WSDM*, 2009.

[17] X.-J. Wang, X. Tu, D. Feng, and L. Zhang. Ranking community answers by modeling question-answer relationships via analogical reasoning. In *SIGIR*, 2009.

[18] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. In *WWW*, 2007.