User Reputation in a Comment Rating Environment

Bee-Chung Chen, Jian Guo*, Belle Tseng and Jie Yang
Yahoo! Labs, Santa Clara, CA
*Harvard University
{beechun,belle,jielabs}@yahoo-inc.com, *jguo@hsph.harvard.edu

ABSTRACT

Reputable users are valuable assets of a web site. We focus on user reputation in a comment rating environment, where users make comments about content items and rate the comments of one another. Intuitively, a reputable user posts high quality comments and is highly rated by the user community. To our surprise, we find that the quality of a comment judged editorially is almost uncorrelated with the ratings that it receives, but can be predicted using standard text features, achieving accuracy as high as the agreement between two editors! However, extracting a pure reputation signal from ratings is difficult because of data sparseness and several confounding factors in users' voting behavior. To address these issues, we propose a novel bias-smoothed tensor model and empirically show that our model significantly outperforms a number of alternatives based on Yahoo! News, Yahoo! Buzz and Epinions datasets.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: User profiles and alert services; G.3 [**Mathematics of Computing**]: Probability and Statistics

General Terms

Algorithms, Experimentation, Human Factors, Measurement

1. INTRODUCTION

The success of social media applications, such as news groups, blogs, online forums, question-answering systems, and social news services, depends highly on the quality and attractiveness of the contributions from regular users. Usually, good contributions are produced by a relatively small set of "reputable" users and consumed by a large user population. Intuitively, one can use a reputation score to quantify how good a user is at producing good contributions. At a high level, we distinguish between two types of reputation scores in a web system, *external scores* and *internal scores*. External (or public) reputation scores are user facing, and usually used to incentivize users for desired behavior; e.g., Yahoo! Answers displays the points that a user earned to recognize those who provided good answers. The key problem is how to design a good reward and incentive mechanism. This is an important problem, but this paper is devoted to another equally important problem, namely estimating internal reputation scores from data. Internal reputation scores are not revealed to users, but used to support internal applications. They are useful for at least the following use cases: (1) *Ranking and recommendation:* Content items (i.e., user contributions) can be ranked or recommended based on the author's reputation. Users can also be recommended to follow their content based on their reputation scores. These scores can also be used in any ranking algorithms as features. (2) *Content enrichment:* Media portals can use the contribution from reputable users to enrich existing content pages. For example, a news site can add related tweets from reputable users to a news article page. (3) *Crowdsourcing:* To reduce costs, it may be desirable for media portals to give some regular users privilege to curate (add value to) their content pages and moderate (manage) user contributions and the contributors. Reputable users are good candidates for such tasks. (4) *Abuse detection:* Reputable users are not likely to abuse the system. Reputation scores can be important features to improve abuse detectors. Also, reputable users may help report spam for removal.

Although we are interested in user reputation in any UGC (user-generated content) systems and the methods developed in this paper may also be applied to them, we ground our study in a particular kind of UGC environment, namely the comment rating environment where users make comments about content items and rate (e.g., thumb up or down) the comments of one another. The reasons are two-fold: (1) Focusing on a specific environment allows us to better understand the notion of reputation since discussion of reputation is usually not meaningful without a specific purpose and context [10, 42]. (2) Almost all UGC systems have an environment to allow users to express their opinions through comments. Thus, the scope of this study is not limited to the specific comment rating environment that we investigate — the results are useful for almost all UGC systems.

To determine the reputation score of a user, we identify two important and quite distinct dimensions:

- *Quality* of the comments that the user makes, which is defined by a set of guidelines that incorporate relevance, usefulness, abusiveness, writing quality, etc., and can be judged by human editors.

KDD'11, August 21–24, 2011, San Diego, California, USA.

- *Support* that the user receives from the user community, which is estimated from the rating data.

Originally, we focused on comment quality and would like to identify users who make high quality comments by appropriately aggregating the ratings that they receive. However, to our surprise, we find that the quality of a comment and the ratings on the comment are almost uncorrelated. Also to our surprise, we find that standard features extracted from comment text can be used to predict quality-based reputation scores with accuracy as high as the agreement between two trained editors (with rank correlation around 0.4 in terms of Kendall's tau). This result suggests that, even for trained editors, it is not easy to have high agreement to what means by high quality, and a few standard text features are sufficient to capture what they agree to. Therefore, we turn our focus to understanding what user ratings stand for. Further investigation reveals that users usually give high ratings to the opinions or viewpoints that they support irrespective of the quality; e.g., supporters of President Obama almost always give pro-Obama comments high ratings. Thus, the level of support that a user receives from the user community is an important notion of user reputation in the comment rating environment, almost orthogonal to quality.

However, it is challenging to quantify the level of support that a user receives based on biased rating data. If we were able to require all users to rate all comments, the average rating that a user receives (i.e., the average rating on all of his/her comments from all users) would be the level of support that he/she receives from the user community. We call this the unobserved *true support score* of a user. However, each user j usually only receives ratings from a small and biased subset of users. For example, the subset of users may happen to include only users who share the same opinion as user j. One of our main technical contributions is to address how to approximate the true support score using sparse and biased rating data.

Any reputation system is subject to gaming and attack. A rigorous solution to this issue is future work.

Contributions and Outline: Specifically, we make the following contributions:

- *Difference between comment quality and ratings:* In Section 2, we define comment quality and show the surprising lack of correlation between quality and ratings based on more than ten thousand editorially labeled comments on Yahoo! News and Yahoo! Buzz. To our knowledge, this lack of correlation has not been reported before. In Section 3, we also show that standard machine-learning models with standard text features can achieve high accuracy for predicting quality-based user reputation, whereas rating-based methods (including PageRank on the rating graph) fail miserably.
- *Average rating over the entire user community as a support-based reputation score:* In Section 4, we define support-based user reputation and propose a simple method to approximate the true support score based on a rating prediction model built on rating data. To our knowledge, this approach to defining user reputation has not been studied before. Since users interact with one another in multiple contexts (e.g., politics vs. entertainment), we further break down support scores by contexts.
- *Bias-smoothed tensor model:* In Section 5, we develop a novel latent factor model for multi-context rating prediction. This model is based on a rating-generation assumption especially suited for estimating user reputation in a comment rating environment. Interesting, the true support score corresponds to the author-reputation factor in the model, which represents reputation after removing a number of confounding factors. In Section 6, we show strong empirical performance of the proposed model.

We conclude and discuss related work in Section 7.

2. EDITORIAL AND RATING DATA

Users who make high quality comments are valuable. It is sometimes believed that, by appropriately aggregating the ratings, we can find those users. In this section, we define the quality-based reputation score, and show that user ratings are almost uncorrelated with comment quality.

Data: We collected comment rating data from Yahoo! Buzz (Y!Buzz) and Yahoo! News (Y!News) during a 3-month period. Users on these sites can post comments on articles and rate the comments of one another by thumb-up votes (positive ratings) and thumb-down votes (negative ratings). We say that user i gives user j a rating if i gives a rating to j's comment. The Y!Buzz dataset includes 1.8M comments authored by 217K users and 11M ratings given by 184K users. The Y!News dataset includes 9M comments authored by 1.6M users and 67M ratings given by 1.5M users.

2.1 Defining Quality Scores

We define the quality score of a comment based on a set of editorial guidelines, and the quality score of a user is obtained by averaging the quality scores of his/her comments. Specifically, we asked Yahoo! editors (who are hired and trained to label web pages for different tasks) to label each comment using the following five labels:

- *Excellent:* Expert opinion, superb quality, thoughtful, voice of reasoning, very useful.
- *Good:* Somewhat useful/thoughtful, informative, interesting, good quality.
- *Fair:* No meaningful/valuable point, but not harmful.
- *Bad:* Irrelevant to the context (article and other comments), bad writing.
- *Abuse:* In violation of terms of service, e.g., spam, harassment, adult content.

A set of example comments for each of the five labels is also given to the editors to help distinguish different labels. We then assign numerical scores to these labels: Excellent=2, Good=1, Fair=0, Bad=−1, and Abuse=−2. We tried other ways of mapping levels to numerical scores and did not observe any significant difference in any of our results. The quality-based reputation score of a user is the average score of the comments posted by the user. The average is, in fact, computed base on 10 labeled comments per user. In total, we collected 13,700 editorial labels on comments in Yahoo! Buzz and 3,000 on comments in Yahoo! News.

2.2 Difference between Quality and Ratings

At first glance, one might think user ratings should have some correlation with quality scores. However, we find that these two do not correlate well. To understand the correlation, we randomly select 707 Yahoo! Buzz users who re-

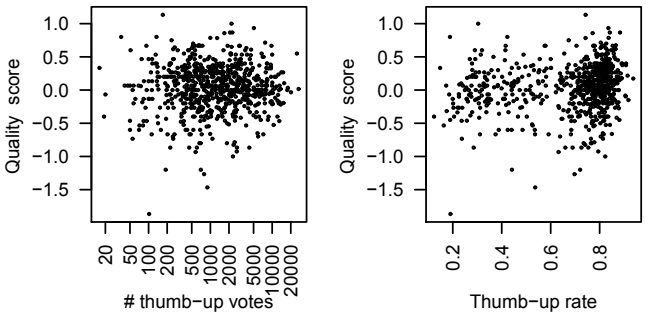

Figure 1: Editorial score vs. rating-based score

ceived at least 100 ratings. Note that on Yahoo! Buzz, users rate comments by thumb-up (positive rating) or thumb-down (negative rating) votes. For each of these user, we obtain his/her quality score and two versions of simple rating-based scores: *#thumb-up votes* of a user is the number of thumb-up votes that the user receives, and the *thumb-up rate* is the user's #thumb-up votes divided by the total number of votes that he/she receives. We also tried different kinds of denominators for thumb-up rate, e.g., number of comments he/she posts, number of unique user rated his/her comments, etc, and observe no significant difference. Figure 1 shows the scatter plots that compare the quality scores to these two rating-based scores. Each point represents one user. If quality scores correlate well with rating-based scores, we should see points along the diagonal line. However, the points in the two plots are almost random, showing lack of correlation between the quality and rating-based scores. The same behavior is observed in Yahoo! News. In Section 3, we will show that more complicated ways of using user ratings to predict quality scores (including a PageRank-style method) also fail to be useful. After inspecting the comments of those users, we find that:

- Editors rate a comment based on the guidelines.
- Users vote for whatever they support. For example, a low quality comment that supports President Obama may draw many thumb-up votes from his supporters. Likewise, a high quality comment that supports President Obama may draw many thumb-down votes from people who are against him.

The above two criteria are very different.

3. PREDICTING QUALITY SCORES

We now apply machine-learning methods to predict quality scores and show that standard text features can capture comment quality to the level of the agreement between two different editors who independently judge those comments.

Features: We extract the following sets of features from a comment: (1) *Length:* Numbers of total words, unique words and characters in a comment. (2) *Lexical diversity:* Number of unique words divided by number of words. (3) *Textual similarity:* Similarity between the comment and the article that the comment is about. We include cosine similarity and jaccard similarity. (4) *Bad words:* We collected hundreds of words considered bad in comments and count the number of bad words in the comment. (5) *Spellcheck:* Percentage of words in the comment that are not in a standard English dictionary. (6) *Readability:* Three readability scores [28, 11, 16]. (7) *Capital words:* Percentage of capital words, percentage of capital characters. (8) *Rating:* Numbers of thumb-up and thumb-down votes, thumb-up rate.

Table 1: Accuracy of Quality Score Prediction

Model	Rank correlation			
	Yahoo! Buzz		Yahoo! News	
	use vote	no vote	use vote	no vote
Random Forest	0.4164	0.4049	0.4601	0.4684
GBRank	0.4054	0.4016	0.4554	0.4515
Linear model	0.3926	0.3901	0.4706	0.4619
Two editors	0.4157		0.4280	
PageRank	0.0576		-0.0368	

Models: We build and compare a number of machine-learning models: Random forest [8], GBRank [44] and linear regression model. Each model takes the features of a comment as the input and predict the editorial score of that comment. The predicted reputation score of a user is the average of the predicted scores of his/her comments.

PageRank: As PageRank-style methods are frequently used to identify authority or reputation, we include an edge-weighted version of PageRank in our comparison. Rating data can be represented as a graph. We draw an edge from user i to user j if i gives a rating to the comment of j. Let n_{ij}^+ and n_{ij}^- denote the number of positive ratings (thumb-up) and the number of negative ratings (thumb-down) that i gives j. We try several different ways of defining edge weights w_{ij}. The first is an unweighted version; i.e., $w_{ij} = 1$ if $n_{ij}^+ > 0$; otherwise, $w_{ij} = 0$. Then, we consider $w_{ij}(b_0, b_1, b_2) = \lfloor b_0 + b_1\, n_{ji}^+ - b_2\, n_{ji}^- \rfloor$; with different settings of b_0, b_1, b_2, we can obtain different weighting schemes. We tune b_0, b_1, b_2 and the damping factor settings heavily and only report the result of the best setting. This heavy tuning may lead to overfitting. However, what we will see is that even overfitted PageRank cannot predict quality scores.

Evaluation: To evaluate the accuracy of a model, we first rank the users by their predicted quality scores and then determine how well it correlates with the ranking produced by the editor-labeled ground-truth quality scores. A rank correlation metric (Kendall's tau) is used to measure the accuracy. If the two ranked lists are the same, the rank correlation is 1. If the model just randomly predicts scores, we get 0. We show the 5-fold cross validation results in Table 1. To understand the importance of ratings in a model, we compare a version of the model that uses rating features (the "use vote" column) to a version that does not (the "no vote" column). To understand how good a rank correlation value (e.g., 0.41) is, we asked two editors to label the same set of comments independently and compute two quality scores for each user in the selected set. The rank correlation between two editors is shown in Table 1. As can be seen, the performance of the models are as good as the agreement between two editors. In the Yahoo! News case, surprisingly our models produce better rank correlations than the agreement between two editors. This is unexpected, but possible because our models are trained and tested based on quality scores given by a single editor per comment (though different comments may be judged by different editors), but the two editors make independent judgment without looking at the labels generated by each other.

Discussion: We first note that PageRank fails miserably even after heavy tuning. Next, by comparing the "use vote" column with the "no vote" column, we find that rating-based features almost add no value to quality prediction models. Finally, we note that the rank correlation 0.41∼0.42 between two editors is not very high. We did refine the guidelines after each round of judgment by going through some examples; however, no significant improvement is observed. Table 1 suggests that our text features almost completely capture what different editors think in common in terms of comment quality.

4. SUPPORT-BASED REPUTATION

As discussed in Section 2.2, users' ratings show their support for different opinions or viewpoints, instead of comment quality. Thus, the level of support that a user receives from the user community is an important notion of reputation in a comment rating environment. Because users interact with one another in multiple contexts (e.g., topics or content categories), it is useful to quantify support level on a per-context basis. In this section, we define support-based reputation scores or simply called *support scores*.

Contexts: A user has different levels of expertise and different degrees of interest in different areas, thus receiving different levels of support. A single score for each user is usually insufficient. For example, a user who receives a high level of support in the context of politics may not receive any support in the context of entertainment. Thus, it is usually desirable to break comments down by *contexts*, which may be topics, categories, etc., and estimate a support score for each user in each context. Now, the question is what should be the contexts in which we estimate support scores. In this paper, we define the set of contexts to be the set of the topic categories that Yahoo! Buzz and Yahoo! News use to classify news articles. Specifically, the context of a comment is the category of the news article that the comment is about, and the context of a rating is the context of the rated comment. Example categories include Business, Entertainment, Health, Lifestyle, Politics, Technology, Sports, etc. These categories are also what the users use to find content items on the sites, and thus, are reasonable choices to define support scores on. If the categories are not available, one can apply topic discovery techniques (e.g., Latent Dirichlet Allocation [5]) to assign topics to comments.

Notation: The rating data consists of a set of observed ratings. We say that user i gives user j a rating (or user i rates user j) if i gives a rating to a comment authored by user j. We call i the rater and j the author. Let y_{ijk} denote the rating that user i gives user j in context k, and $\mathcal{U}$ denote the set of all users in the comment rating environment.

True support score: We define the *true support score* s_{jk} of user j in context k as the average rating that user j would receive in context k from all users, i.e., $s_{jk} = \sum_{i \in \mathcal{U}} y_{ijk}/|\mathcal{U}|$, where $|\mathcal{U}|$ denote the size of set $\mathcal{U}$. For example, consider a binary rating environment (e.g., thumb-up vs. thumb-down). The true support score of user j is proportional[1] to the total number of users who would rate user j positively if we were able to require all users to rate user j. It is a very natural and ideal definition of support. It is also interesting to note that true support scores are not sensitive to gaming and attacks since it involves a summation over the entire user base (as long as attackers do not control a large fraction of users). However, these scores are not computable from rating data because each user j usually only receives ratings from a small subset of users.

[1]The denominator of a true support score is a constant.

Challenges: It is challenging to estimate the true support scores. One simple approach would be to compute the support score of user j in context k as the average rating over the small subset of users who have rated user j in context k. The major issues with this approach are:

- *Data sparseness:* Because the subset is small, this average is usually unreliable.
- *Biased sample:* Even if the user subset is large enough to compute a reliable average number, it may not be a representative sample of the entire user base. For example, this subset of users may happen to include only users who like one another (e.g., friends), or users who share the same viewpoint as user j in the context.

Prediction-based support score: One conceptually simple approach to approximate the true support score is to use a rating-prediction model — whenever we see an unobserved (voter i, author j, context k) triple, we use the predicted rating $\hat{y}_{ijk}$, which is output from the model. The accuracy of these predicted support scores are directly linked to the accuracy of the rating-prediction model used; the better the model, the better the scores. However, averaging over all users can still be quite computationally intensive; nevertheless, one may simply use a large enough random sample to compute the average. In the next section, we propose a novel rating-prediction model that is accurate and does not require this averaging process to obtain support scores.

5. BIAS-SMOOTHED TENSOR MODEL

In this section, we introduce a novel latent factor model for rating prediction, which is useful for extracting context-specific reputation scores from rating data. This model is motivated by the observation that users' ratings are usually generated according to several factors:

- *Author reputation:* The comments made by reputable users tend to receive high ratings.
- *Rater bias:* Some users rate more positively than others.
- *Agreement in opinions:* Some users vote positively (or negatively) on a comment because they agree (or disagree) with the opinion in the comment, even when the comment has no useful information (e.g., "Obama is great").

To extract pure reputation scores (author reputation) from rating data, it is important to remove the effect of confounding factors (rater bias and agreement in opinions). For example, a positive vote from a user who always rates user j positively should carry little weight when computing user j's reputation score. Interestingly, as we will see later, the author reputation factor with the effect of confounding factors removed corresponds to the true support score.

5.1 Model

Notation: In addition to ratings, we may also have features for users. Since we always use i to denote a rater and j to denote an author, we abuse the notation a bit by using $\boldsymbol{x}_{ik}$ to denote user i's feature vector when he/she acts as a rater

in context k and $\boldsymbol{x}_{jk}$ to denote user j's feature vector when he/she acts as an author in context k.

Model specification: We assume that each rating y_{ijk} is generated according to (1) the rater bias α_{ik} of user i in context k, (2) the author reputation factor β_{jk} of user j in context k and (3) the similarity between the opinions of the two users in context k, which is modeled as a tensor product $\langle \boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{w}_k \rangle = \sum_\ell v_{i\ell}\, v_{j\ell}\, w_{j\ell}$ of the opinion factor vectors $\boldsymbol{v}_i$, $\boldsymbol{v}_j$ of the two users and a context-specific weight vector $\boldsymbol{w}_k$. We use $\dim(v)$ to denote the number of dimensions of the opinion factor vector of a user. Mathematically, we assume:

$$y_{ijk} \sim N(\mu_{ijk},\ \sigma_y^2) \text{ or Bernoulli}(1/(1+\exp\{-\mu_{ijk}\}))$$
$$\mu_{ijk} = \alpha_{ik} + \beta_{jk} + \langle \boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{w}_k \rangle$$

For numeric ratings, we use the Gaussian distribution denoted by $N(\text{mean,var})$; for binary ratings, we use the Bernoulli distribution and the standard logistic transformation. We then specify the prior distributions of the factors:

$$\alpha_{ik} \sim N(\boldsymbol{g}_k' \boldsymbol{x}_{ik} + q_k \alpha_i,\ \sigma_{\alpha,k}^2), \quad \alpha_i \sim N(0,1) \tag{1}$$
$$\beta_{jk} \sim N(\boldsymbol{d}_k' \boldsymbol{x}_{jk} + r_k \beta_j,\ \sigma_{\beta,k}^2), \quad \beta_j \sim N(0,1) \tag{2}$$
$$\boldsymbol{v}_i \sim N(\boldsymbol{0},\ \sigma_v^2 I), \quad \boldsymbol{w}_k \sim N(\boldsymbol{0}, I) \tag{3}$$

These priors describe the generation process of the factors. We first generate a global rater-bias factor α_i for user i from the standard Gaussian. Then, the expected value of the context-specific factor $E[\alpha_{ik}]$ for context k is determined by a linear function $\boldsymbol{g}_k' \boldsymbol{x}_{ik} + q_k \alpha_i$ of the global factor α_i and features $\boldsymbol{x}_{ik}$, where $\boldsymbol{g}_k$ and q_k are the context-specific regression weights. The actual context-specific factor α_{ik} is then generated by adding a Gaussian random effect to $E[\alpha_{ik}]$ to allow deviation from the linear function. β_{jk} is generated in a similar way, where β_j is the global author-reputation factor for user j, and $\boldsymbol{d}_k$ and r_k are the regression weights for context k. For the opinion-related factors $\boldsymbol{v}_i$ and $\boldsymbol{w}_k$, we just put a Gaussian zero-mean prior, assuming no opinion preference a priori. We call this model the *bias-smoothed tensor model*, because the per-user context-specific "bias" terms in the model (i.e., α_{ik} and β_{jk}) are "smoothed" based on the global factors (i.e., α_i and β_j). We note that this model can be thought of as a significant extension to [18] by adding regression-based hierarchical priors to the per-user bias terms.

To summarize, given a rating dataset $\boldsymbol{y} = \{y_{ijk}\}$, we will learn the factors $\boldsymbol{\eta} = \{\alpha_{ik}, \beta_{jk}, \alpha_i, \beta_j, \boldsymbol{v}_i, \boldsymbol{w}_k\}$ and prior parameters (regression weights and variances) $\boldsymbol{\Theta} = \{\boldsymbol{g}_k, \boldsymbol{d}_k, q_k, r_k, \sigma_y^2, \sigma_{\alpha,k}^2, \sigma_{\beta,k}^2, \sigma_{v,k}^2\}$ from the data. Note that it is appropriate to set the prior variances of α_i, β_j and $\boldsymbol{w}_k$ to 1, because q_k, r_k and $\boldsymbol{v}_i$ are learned from data; if we scale up the variance of α_i, q_k will automatically be scaled down, since they are not separable in $q_k \alpha_i$.

Special case: Smoothed reputation-only model: To better understand the above model, it is instructive to study a simplified special case. Let us keep only the author-reputation factors and set others to zero (thus called reputation-only model); i.e.,

$$\mu_{ijk} = \beta_{jk}, \quad \beta_{jk} \sim N(\boldsymbol{d}_k' \boldsymbol{x}_{jk} + r_k \beta_j,\ \sigma_{\beta,k}^2), \quad \beta_j \sim N(0,1)$$

This model handles the data sparseness issue, but does not address the sample bias issue (discussed in Section 4). To understand how it handles data sparseness, we note that if we have a lot of rating data for user j in context k, the author-reputation factor β_{jk} would mostly depend on the observed data. However, if the data is sparse, β_{jk} would be influenced by the rating data of user j in other contexts and the user features $\boldsymbol{x}_{jk}$. In fact, β_{jk} would be "shrunk" toward $\boldsymbol{d}_k' \boldsymbol{x}_{jk} + r_k \beta_j$. The second term $r_k \beta_j$ combines information from other contexts; β_j is intuitively a weighted average over β_{jk} and the weight of context k depends on $\sigma_{\beta,k}^2$, which is learned from data. The first term $\boldsymbol{d}_k' \boldsymbol{x}_{jk}$ is the prediction based on features $\boldsymbol{x}_{jk}$ and the weight vector $\boldsymbol{d}_k$ is learned from data; when the features are predictive, the number of ratings required to achieve good accuracy would be small. Intuitively, the potentially unstable β_{jk} is "smoothed" based on a regression on β_j and $\boldsymbol{x}_{jk}$. The full model also enjoys this nice property.

Opinion factors: To understand why $\langle \boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{w}_k \rangle$ represents agreement between opinions, consider another simplified special case where $\dim(v) = 1$ and $\boldsymbol{w}_k = 1$. Intuitively, positive v_i means a positive opinion on an issue; negative v_i means a negative opinion on the issue; and the absolute value of v_i represents the strength of the opinion. If v_i and v_j are both positive or both negative, $\langle \boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{w}_k \rangle = v_i' v_j$ is high (at least a positive number); otherwise, it is low (a negative number). The prior $v_i \sim N(0, \sigma_v^2 I)$ will "shrink" v_i toward zero, meaning if we do not have much data for a user, we do not associate him/her with a strong opinion. The extension from $\dim(v) = 1$ to n allows the model to capture opinions on multiple latent issues. Ideally, each context should have a different set of issues. This would lead to a high dimensionality problem; e.g., if we have m contexts and n issues per context, then $\dim(v) = mn$. The extension from fixing $\boldsymbol{w}_k = 1$ to learning $\boldsymbol{w}_k$ from data allows us to reduce the dimensionality by assuming that the agreement between two users in context k is determined by applying a context-specific weight vector $\boldsymbol{w}_k$ to the global latent opinion profiles $\boldsymbol{v}_i$, $\boldsymbol{v}_j$ of the two users.

5.2 Predicted Support Scores

Let $\hat{y}_{ijk}$ be the predicted rating that user i would give user j in context k, $\bar{\alpha}_k = \sum_i \alpha_{ik}/|\mathcal{U}|$ and $\bar{\boldsymbol{v}} = \sum_i \boldsymbol{v}_i/|\mathcal{U}|$. For simplicity, we first consider the Gaussian model. The predicted support score for user j in context k is

$$\hat{s}_{jk} = \sum_{i \in \mathcal{U}} \hat{y}_{ijk}/|\mathcal{U}| = \beta_{jk} + \bar{\alpha}_k + \langle \bar{\boldsymbol{v}}, \boldsymbol{v}_j, \boldsymbol{w}_k \rangle$$

Efficient computation: Notice that $\bar{\alpha}_k$ and $\bar{\boldsymbol{v}}$ are sufficient statistics for computing $\hat{s}_{jk}$. We only need to compute $\bar{\alpha}_k$ and $\bar{\boldsymbol{v}}$ once; after that, the computation of $\hat{s}_{jk}$ does not involve any averaging.

Removing global opinion bias: Sometimes it is useful to force $\bar{\boldsymbol{v}} = 0$, meaning that we believe, after averaging over all users, there is no global opinion bias. After training, this can be done by defining the following factors: $\boldsymbol{v}_i^{\text{new}} = \boldsymbol{v}_i - \bar{\boldsymbol{v}}$, $\alpha_{ik}^{\text{new}} = \alpha_{ik} + \langle \boldsymbol{v}_i^{\text{new}}, \bar{\boldsymbol{v}}, \boldsymbol{w}_k \rangle$ and $\beta_{jk}^{\text{new}} = \beta_{jk} + \langle \boldsymbol{v}_j^{\text{new}}, \bar{\boldsymbol{v}}, \boldsymbol{w}_k \rangle$. It can be shown that

$$\mu_{ijk} = \alpha_{ik}^{\text{new}} + \beta_{jk}^{\text{new}} + \langle \boldsymbol{v}_i^{\text{new}}, \boldsymbol{v}_j^{\text{new}}, \boldsymbol{w}_k \rangle + c_k,$$

where c_k is a constant, and $\bar{\boldsymbol{v}}^{\text{new}} = 0$.

Confounding factor removal: Consider the case where $\bar{\boldsymbol{v}} = 0$ or $\bar{\boldsymbol{v}}^{\text{new}} = 0$. We ignore the superscript for succinctness. Then, $\hat{s}_{jk} = \beta_{jk} + \bar{\alpha}_k$. Since $\bar{\alpha}_k$ is a constant, the

predicted support score is basically the author-reputation factor β_{jk}.

Logistic model: For the logistic model, the correspondence is not that direct. However, since reputation scores are used to rank users, the absolute values of the scores are not important. Thus, we may redefine the support score s_{jk} to be the average log-odds (instead of probability or predicted rating) that other users would rate user j positively in context k. Then, the above discussion applies.

5.3 Fitting Procedure

We use the Monte Carlo EM algorithm [6] to fit the model. The EM algorithm iterates between an E-step and an M-step until convergence. Let $\hat{\boldsymbol{\Theta}}^{(t)}$ denote the current estimated value of the set of prior parameters $\boldsymbol{\Theta}$ (defined in Section 5.1) at the beginning of the tth iteration.

- **E-step:** We take expectation of the complete data log likelihood with respect to the posterior of latent factors $\boldsymbol{\eta}$ conditional on the observed data $\boldsymbol{y}$ and the current estimate of $\boldsymbol{\Theta}$; i.e., compute

$$f_t(\boldsymbol{\Theta}) = E_{\boldsymbol{\eta}\sim\Pr(\boldsymbol{\eta}\,|\,\boldsymbol{y},\hat{\boldsymbol{\Theta}}^{(t)})}[\log\Pr(\boldsymbol{y},\boldsymbol{\eta}\,|\,\boldsymbol{\Theta})]$$

  as a function of $\boldsymbol{\Theta}$, where the expectation is taken over the posterior distribution of $(\boldsymbol{\eta}\,|\,\boldsymbol{y},\hat{\boldsymbol{\Theta}}^{(t)})$, which is not in closed-form, thus, approximated by Monte Carlo mean.
- **M-step:** We maximize the expected complete data log likelihood from the E-step to obtain updated values of $\boldsymbol{\Theta}$; i.e., find $\hat{\boldsymbol{\Theta}}^{(t+1)} = \arg\max_{\boldsymbol{\Theta}}\ f_t(\boldsymbol{\Theta})$.

Note that the actual computation in the E-step is to generate sufficient statistics for computing $\arg\max_{\boldsymbol{\Theta}}\ f_t(\boldsymbol{\Theta})$, so that we do not need to scan the raw data every time when we need to evaluate $f_t(\boldsymbol{\Theta})$. At the end, we obtain a maximum likelihood estimate of $\boldsymbol{\Theta}$, i.e.,

$$\arg\max_{\boldsymbol{\Theta}}\Pr(\boldsymbol{y}\,|\,\boldsymbol{\Theta}) = \arg\max_{\boldsymbol{\Theta}}\int\Pr(\boldsymbol{y},\boldsymbol{\eta}\,|\,\boldsymbol{\Theta})\,d\boldsymbol{\eta},$$

modulo local maximums and Monte Carlo errors. We then can use the estimated $\hat{\boldsymbol{\Theta}}$ to obtain the posterior distribution of the factors $(\boldsymbol{\eta}\,|\,\boldsymbol{y},\hat{\boldsymbol{\Theta}})$. For simplicity, we only describe the algorithm to fit the Gaussian model. The logistic model for binary response is fitted using variational approximation [22]. For convenience, we provide the formulas of $\log\Pr(\boldsymbol{y},\boldsymbol{\eta}\,|\,\boldsymbol{\Theta})$ and $E_{\boldsymbol{\eta}}[\log\Pr(\boldsymbol{y},\boldsymbol{\eta}\,|\,\boldsymbol{\Theta})]$ in the appendix.

5.4 E-Step

We use a Gibbs sampler to draw L samples of the factors and compute the Monte Carlo mean.

Draw α_i and α_{ik}: Now, consider that all the other factors are given. Let $o_{ijk} = y_{ijk} - \beta_{jk} - \langle\boldsymbol{v}_i,\boldsymbol{v}_j,\boldsymbol{w}_k\rangle - \boldsymbol{g}_k'\boldsymbol{x}_{ik}$ and $\alpha_{ik}^* = \alpha_{ik} - \boldsymbol{g}_k'\boldsymbol{x}_{ik}$. Then, we have

$$o_{ijk}\sim N(\alpha_{ik}^*,\ \sigma_y^2),\ \alpha_{ik}^*\sim N(q_k\alpha_i,\ \sigma_{\alpha,k}^2),\ \alpha_i\sim N(0,1)$$

Let $\mathcal{J}_{ik}$ denote the set of authors that user i has rated in context k. Let $\boldsymbol{o}_{ik} = \{o_{ijk}\}_{\forall j\in\mathcal{J}_{ik}}$. Since all the distributions are normal, the distribution of $(\alpha_i|\boldsymbol{o}_{ik})$ is also normal, and can be obtained by $\int p(\alpha_i,\alpha_{ik}^*\,|\,\boldsymbol{o}_{ik})\,d\,\alpha_{ik}^*$. Let $\rho_{ik} = (1+|\mathcal{J}_{ik}|\,\sigma_{\alpha,k}^2/\sigma_y^2)^{-1}$. We have

$$E[\alpha_i\,|\,\boldsymbol{o}_{ik}] = Var[\alpha_i\,|\,\boldsymbol{o}_{ik}]\left(\rho_{ik}\,q_k\sum_{j\in\mathcal{J}_{ik}}\tfrac{o_{ijk}}{\sigma_y^2}\right)$$
$$Var[\alpha_i\,|\,\boldsymbol{o}_{ik}] = \left(1+\tfrac{q_k^2}{\sigma_{\alpha,k}^2}(1-\rho_{ik})\right)^{-1}$$

Then, letting $\boldsymbol{o}_i = \{\boldsymbol{o}_{ik}\}_{\forall k}$, we obtain the distribution of $(\alpha_i\,|\,\boldsymbol{o}_i)$, which is normal with

$$E[\alpha_i\,|\,\boldsymbol{o}_i] = Var[\alpha_i\,|\,\boldsymbol{o}_i]\left(\sum_k\tfrac{E[\alpha_i\,|\,\boldsymbol{o}_{ik}]}{Var[\alpha_i\,|\,\boldsymbol{o}_{ik}]}\right)$$
$$Var[\alpha_i\,|\,\boldsymbol{o}_i] = \left(1+\sum_k\left(\tfrac{1}{Var[\alpha_i\,|\,\boldsymbol{o}_{ik}]}-1\right)\right)^{-1}$$

Now, we draw α_i from this distribution. Then, for each k, we draw α_{ik} from the distribution of $(\alpha_{ik}\,|\,\alpha_i,\boldsymbol{o}_i)$, which is normal with

$$E[\alpha_{ik}\,|\,\alpha_i,\boldsymbol{o}_i] = V_{ik}^{(\alpha)}\left(\tfrac{q_k\alpha_i}{\sigma_{\alpha,k}^2}+\sum_{j\in\mathcal{J}_{ik}}\tfrac{o_{ijk}}{\sigma_y^2}\right)+\boldsymbol{g}_k'x_{ik}$$
$$Var[\alpha_{ik}\,|\,\alpha_i,\boldsymbol{o}_i] = V_{ik}^{(\alpha)} = \left(\tfrac{1}{\sigma_{\alpha,k}^2}+\tfrac{1}{\sigma_y^2}|\mathcal{J}_{ik}|\right)^{-1}$$

Draw β_j and β_{jk}: This is similar to the previous case.

Draw $\boldsymbol{v}_i$: For each user i, draw $\boldsymbol{v}_i$ from $(\boldsymbol{v}_i\,|\,\text{Rest})$, which is normal, for all i. Note that "Rest" here denote all the other factors. Let $\mathcal{I}_{ik}$ denote the set of users that user i receives ratings from. Let $o_{ijk} = y_{ijk} - \alpha_{ik} - \beta_{jk}$ and $\boldsymbol{v}_{jk} = \boldsymbol{v}_j\circ\boldsymbol{w}_k$, meaning element-wise product.

$$E[\boldsymbol{v}_i\,|\,\text{Rest}] = V_i^{(v)}\left(\sum_{j\in\mathcal{J}_{ik}}\tfrac{o_{ijk}\boldsymbol{v}_{jk}}{\sigma_y^2}+\sum_{j\in\mathcal{I}_{ik}}\tfrac{o_{jik}\boldsymbol{v}_{jk}}{\sigma_y^2}\right)$$
$$Var[\boldsymbol{v}_i\,|\,\text{Rest}] = V_i^{(v)} = \left(\tfrac{I}{\sigma_v^2}+\sum_{j\in(\mathcal{J}_{ik}\cup\mathcal{I}_{ik})}\tfrac{\boldsymbol{v}_{jk}\boldsymbol{v}_{jk}'}{\sigma_y^2}\right)^{-1}$$

Draw $\boldsymbol{w}_k$: For each context k, draw $\boldsymbol{w}_k$ from $(\boldsymbol{w}_k\,|\,\text{Rest})$, which is normal, for all k. Let $\mathcal{U}_k = \{(i,j) : \text{user } i \text{ gives user } j \text{ a rating in context } k\}$ and $o_{ijk} = y_{ijk} - \alpha_{ik} - \beta_{jk}$.

$$E[\boldsymbol{w}_k\,|\,\text{Rest}] = V_k^{(w)}\left(\sum_{(i,j)\in\mathcal{U}_k}\tfrac{o_{ijk}(\boldsymbol{v}_{ik}\circ\boldsymbol{v}_{jk})}{\sigma_y^2}\right)$$
$$Var[\boldsymbol{w}_k\,|\,\text{Rest}] = V_k^{(w)} = \left(I+\sum_{(i,j)\in\mathcal{U}_k}\tfrac{(\boldsymbol{v}_{ik}\circ\boldsymbol{v}_{jk})(\boldsymbol{v}_{ik}\circ\boldsymbol{v}_{jk})'}{\sigma_y^2}\right)^{-1}$$

5.5 M-Step

We use the following notations: $\hat{a}$ and $\hat{V}[a]$ to denote the posterior mean and variance of factor a, and $\hat{V}[a,b]$ to denote the covariance between factor a and b, which are computed as the sample mean, variance, covariance based on the Monte Carlo samples obtained in the E-step.

Estimating $(\boldsymbol{g}_k, q_k, \sigma_{\alpha,k}^2)$: Let $\boldsymbol{\theta}_k = (q_k,\boldsymbol{g}_k)$, $\boldsymbol{z}_{ik} = (\hat{\alpha}_i,\boldsymbol{x}_{ik})$, $\Delta_i = \text{diag}(\hat{V}[\alpha_i],\boldsymbol{0})$ and $\boldsymbol{c}_{ik} = (\hat{V}[\alpha_{ik},\alpha_i],\boldsymbol{0})$. We want to find $\boldsymbol{\theta}_k$ and $\sigma_{\alpha,k}$ that minimize

$$\tfrac{1}{\sigma_{\alpha,k}^2}\sum_i\left((\hat{\alpha}_{ik}-\boldsymbol{\theta}_k'\boldsymbol{z}_{ik})^2+\boldsymbol{\theta}_k'\Delta_i\boldsymbol{\theta}_k-2\boldsymbol{\theta}_k'\boldsymbol{c}_{ik}+\hat{V}[\alpha_{ik}]\right)$$
$$+\ N_k\ \log\sigma_{\alpha,k}^2,$$

where N_k is the number of raters in context k. By setting the derivative to zero, we obtain

$$\hat{\boldsymbol{\theta}}_k = \left(\sum_i(\Delta_i+\boldsymbol{z}_{ik}\boldsymbol{z}_{ik}')\right)^{-1}\left(\sum_i(\boldsymbol{z}_{ik}\hat{\alpha}_{ik}+\boldsymbol{c}_{ik})\right)$$
$$\hat{\sigma}_{\alpha,k}^2 = \left((\hat{\alpha}_{ik}-\hat{\boldsymbol{\theta}}_k'\boldsymbol{z}_{ik})^2+\hat{\boldsymbol{\theta}}_k'\Delta_i\hat{\boldsymbol{\theta}}_k-2\hat{\boldsymbol{\theta}}_k'\boldsymbol{c}_{ik}+\hat{V}[\alpha_{ik}]\right)\ /\ N_k$$

Estimating $(\boldsymbol{d}_k, r_k, \sigma_{\beta,k}^2)$: Similar to the previous case.

Estimating σ_v^2 and σ_y^2: Let R and N denote the total numbers of ratings and users. We obtain $\hat{\sigma}_v^2 = (\sum_i\text{trace}(\hat{V}[\boldsymbol{v}_i])+\hat{\boldsymbol{v}}_i'\hat{\boldsymbol{v}}_i)/(N\dim(v))$ and $\sigma_y^2 = ((y_{ijk}-\hat{\mu}_{ijk})^2+\hat{V}[\mu_{ijk}])/R$.

6. EXPERIMENTAL RESULTS

We show strong empirical performance of our model in this section. One novel aspect of our model is the use of a regression-based hierarchical prior (Equation 1 and 2) to link to the global factor. We first evaluate this component by considering a special case, the smoothed reputation-only model. Next, we show that the full bias-smoothed tensor model outperforms a number of competitive models. Finally, we show that, by removing bias, we see a higher correlation between the predicted support scores and the quality scores.

6.1 Smoothed Reputation-Only Model

We use the Y!Buzz and Y!News datasets described in Section 2 to evaluate this model. The ratings in these two datasets are thumb-up and thumb-down votes. Thus, we would like to see whether the regression-based hierarchical prior can help improve thumb-up rate prediction in the presence of data sparseness. Each dataset has 10 contexts, which are the content categories. Note that for users who receive many votes, their *observed thumb-up rates*, defined as number of thumb-up / total vote, would be very close to their true thumb-up rates. Thus, we select (user, category) pairs that receive at least 50 votes, for users who have data in at least 4 categories, and treat the observed thumb-up rates for such pairs as the ground truth. We only evaluate our model based on these ground-truth pairs.

We use 5-fold cross validation to compare the following methods: (1) *Constant:* The constant model predicts every (user, category) pairs to have the same value, which is the average thumb-up rate in the test set. (2) *Per-user average:* This model computes the average thumb-up rate for each user (ignoring contexts) based on the training data, and then predict the user's thumb-up rates in unobserved contexts as this user-specific average rate. (3) *Hierarchical-only:* This is the smoothed reputation-only model without the regression part; i.e., $\beta_{jk} \sim N(\beta_j,\ \sigma^2_{\beta,k})$. (4) *Regression & hierarchical:* This is the smoothed reputation-only model; i.e., $\beta_{jk} \sim N(\boldsymbol{d}'_k\boldsymbol{x}_{jk}+r_k\beta_j,\ \sigma^2_{\beta,k})$. We measure the model accuracy using the root mean squared error (RMSE) between the predicted thumb-up rate and the observed thumb-up rate. The result is in the following table.

	RMSE	
Method	Buzz	News
Regression & hierarchical	0.1049	0.1488
Hierarchical-only	0.1142	0.1504
Per-user average	0.1222	0.1537
Constant	0.1901	0.1789

We note that the order of the four methods according to their accuracies are the same in all five folds — the difference is significant. On the Yahoo! Buzz data, Regression & hierarchical reduces the error by 45% from Constant, 14% from Per-user average and 8% from Hierarchical-only. On the Yahoo! News data, the error reduction is smaller. This is probably due to the fact that our current News categories are automatically extracted from article URLs (not editorially labeled) and thus noisier than Buzz categories (which are editorially labeled). Thus, the data does not provide precise context information for the model to take advantage.

6.2 Rating Prediction

Since the bias-smoothed tensor model is a rating-prediction model, we compare its performance to other rating prediction models. Our goal is not to show that our model beats the best of all rating prediction methods, but to demonstrate that, by incorporating the rating behavior on comments, we can achieve better performance than general-purpose state-of-the-art methods. After all, any rating prediction model can be used to predict support scores. In addition to Y!News and Y!Buzz datasets, we also use the Epinions dataset [32]; although it is not a comment rating dataset, it is publicly available and can be used to compare different rating prediction methods. Since reputable users should have certain level of activity, we only select users who authored at least 30 comments to be included in the Y!News and Y!Buzz experiments. For the Epinions experiment, we only select authors who posted at least 10 reviews and received at least 100 ratings; we also create binary ratings by setting rating values ≥ 5 to 1 and otherwise 0. We use the 10 content categories as the contexts for Y!News and Y!Buzz, and the top 10 verticals as the contexts for Epinions.

We compare the following four methods: (1) *BST*: the proposed bias-smoothed tensor model. (2) *BSM*: a simplified version of BST, called bias-smoothed matrix factorization. Instead of tensor factorization, we do matrix factorization; i.e., $\mu_{ijk} = \alpha_{ik} + \beta_{jk} + \boldsymbol{v}'_i\boldsymbol{v}_j$. (3) *GMF*: the global matrix factorization model, which does not use the context information; i.e., $\mu_{ijk} = \alpha_i + \beta_j + \boldsymbol{v}'_i\boldsymbol{u}_j$. We also tried using $\boldsymbol{v}'_i\boldsymbol{v}_j$ instead of $\boldsymbol{v}'_i\boldsymbol{u}_j$ and observed no significant difference. (4) *SMF*: the separate matrix factorization model, where we perform independent matrix factorization for each context; i.e., $\mu_{ijk} = \alpha_{ik} + \beta_{jk} + \boldsymbol{v}'_{ik}\boldsymbol{u}_{jk}$ without hierarchical prior on α_{ik} and β_{jk}. Again, we also tried using $\boldsymbol{v}'_{ik}\boldsymbol{v}_{jk}$ instead of $\boldsymbol{v}'_{ik}\boldsymbol{u}_{jk}$ and observed no significant difference. Let Model.n (e.g., BST.5) denote a model with $\dim(v)=n$ (i.e., n opinion factors per user). We only report the best $\dim(v)$ for each method found in tuning. The mixed-membership stochastic blockmodel (MMSB) [3] is also tried on the rating graph (if user i rates user j positively (or negatively), we create a positive (or negative) edge from i to j). We slightly modified MMSB to take negative edges (instead of assuming no edge means negative). However, we observe that MMSB has much worse accuracy than the above factorization methods. To make the plots cleaner, we do not plot its performance.

Since the ratings are binary, we measure the performance of different methods by AUC (area under ROC curve) [7]; higher AUC means better accuracy. To understand how different methods handle data sparseness, when we compute the test-set AUC numbers, we only consider (author, context) pairs, each of which receives $\leq N$ ratings in the training set. We then vary N from 0 to 10, so that the performance of a method is represented by a curve. These AUC curves are shown in Figure 2 for the three datasets. As can be seen, BST has the best performance across the three datasets. On the Epinions dataset, BST and BSF are indistinguishable, suggesting that the interaction behavior between users is similar across all contexts. By comparing BST to SMF and GMF, we see a big gain by using regression-based hierarchical priors.

6.3 Support vs. Quality

Although we saw lack of correlation between ratings and quality, here we show that, by removing bias, we can increase the correlation. The rank correlation between the predicted support scores and editor-labeled quality scores are shown in the following table.

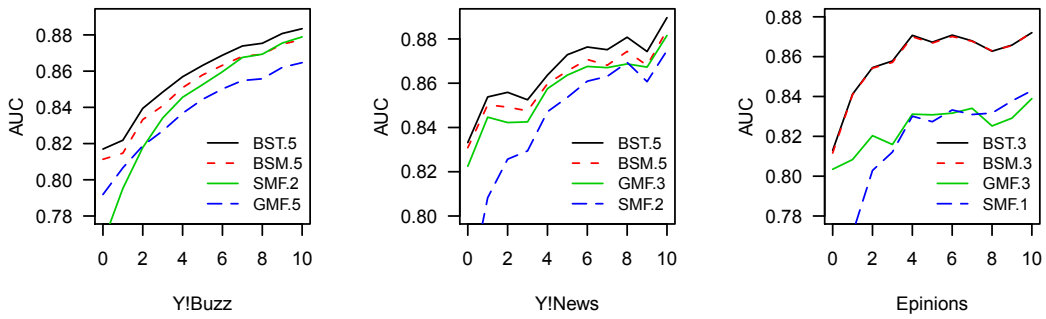

Figure 2: AUC of different rating prediction methods as a function of different data-sparseness levels

Model	Buzz	News
No opinion factor (dim(v)=0)	0.0817	0.0900
Use opinion factor (dim(v)=1)	0.1264	0.1328

As can be seen, having the opinion factors in the model (that removes the effect of opinion bias) significantly improves the rank correlation. Interestingly, different numbers of option factors do not make a difference. Comparing to Table 1, we see that our predicted support scores (based purely on ratings without using any editorial data) have much better performance than PageRank, but much worse than models trained using editorial labels. This worse performance is again due to the inherent difference between quality and support as discussed in Section 2.2.

7. CONCLUSION AND RELATED WORK

We study the problem of how to estimate user reputation in a comment rating environment and provide three novel contributions: (1) We find surprising lack of correlation between comment quality and user ratings, which has not been reported before. (2) We provide a new approach to estimating support-based reputation scores. (3) We propose a novel latent factor model that captures the generation behavior of comment ratings. In the following, we discuss prior work related to these contributions.

Quality vs. Ratings: Text quality is a heavily studied problem [40]. The most related studies are those on quality of UGC items, e.g., blogs [17], reviews [29, 12, 30], answers [2] and others [20]. On the rating behavior side, votes on reviews and answers are a popular topic [9, 29, 2]. Specifically, an exploratory analysis of votes on reviews is reported in [9]. Liu et al. [29] found three biases of user votes in reviews and developed a classifier based on text features. However, they did not try to model these biases or confounding factors as we do. Lu et al. [30] proposed an ℓ_2-regularization model to improve review quality prediction by incorporating user profiles and social networks, and Hsu et al. [20] proposed an SVM-based supervised ranking algorithm to rank comment qualities. Talwar et al. [39] found that a user's rating partly reflects the difference between true quality and prior expectation of quality in a hotel review dataset. Agichtein et al. [2] found that ratings are useful for predicting answer quality, unlike what we found in comments. To our knowledge, the sharp difference between quality and ratings in a comment rating environment has not been studied before.

Definition of reputation: External (user facing) reputation scores are used to incentivize users for desired behavior. A few examples are [35, 37]. These scores are usually count-based and have to be simple so that users can easily understand them. However, our focus is on internal scores. In this area, many graph-based approaches have been proposed based on PageRank [19, 23], trust/distrust propagation [15, 45], mutual reinforcement propagation between user reputation scores and content quality [4, 33] and frequent patterns [14]. In addition, Agarwal et al. [1] also proposed a graph-based algorithm to identify influential bloggers. Zhang et al. [43] conducted an extensive study comparing different scoring-based and graph-based reputation estimation methods using a Java online forum. While most existing graph-based approaches ignore the presence of confounding factors, Shin et al. [38] proposed a propagation algorithm trying to separate user sociability from reputation. Statistical models are also used to identify reputable users. For example, Maeno [31] developed a model for covert node discovery in social networks and applied it to identifying suspicious logs. Kuter [27] proposed a trust inference algorithm that uses a probabilistic sampling technique to estimate the confidence in the trust information from some designated sources. See Jøsang et al. [24] and Farmer et al. [10] for surveys of reputation methods. Different from the above work, we define a conceptually simple, but *unobserved* support score and address the challenges of computing the score using statistical models.

Latent factor models: We reduce the problem of estimating support scores to a rating prediction problem and develop a novel latent factor model. Although any rating prediction model can be used, we have shown strong performance of ours. Our model is a significant extension to [18] by adding regression-based hierarchical priors to the per-user bias terms. Rating prediction is a hot problem in recommender systems. Factorization methods [36, 25] are very competitive methods if not the best. Our goal is not to show that our model beats the best of rating prediction methods, but to demonstrate that, by incorporating the rating behavior in comments, we can achieve better performance than general-purpose state-of-the-art methods.

Other related work: We note that the problem of finding reliable users in crowdsourcing is an interesting line of research [21, 34, 41]. Exploratory analysis of comment rating environments includes [26, 13].

8. REFERENCES

[1] N. Agarwal, H. Liu, L. Tang, and P. S. Yu. Identifying the influential bloggers in a community. In *WSDM*, 2008.

[2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. *Proceedings of the international conference on Web search and web data mining - WSDM '08*, 2008.

[3] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *JMLR*, 2008.

[4] 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.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3, 2003.

[6] J. Booth and J. Hobert. Maximizing generalized linear mixed model likelihoods with an automated monte carlo EM algorithm. *J.R.Statist. Soc. B*, 1999.

[7] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 1997.

[8] L. Breiman. Random forests. *Machine learning*, 2001.

[9] C. Danescu-niculescu mizil, J. Kleinberg, and L. Lee. How Opinions are Received by Online Communities : A Case Study on Amazon.com Helpfulness Votes. *WWW*, 2009.

[10] R. Farmer and B. Glass. *Building Web Reputation Systems*. Yahoo Press, 2010.

[11] R. Flesch. A new readability yardstick. *Journal of Applied Psychology*, 1948.

[12] A. Ghose and P. G. Ipeirotis. Estimating the Helpfulness and Economic Impact of Product Reviews : Mining Text and Reviewer Characteristics. *TKDE*, 2009.

[13] V. Gómez, A. Kaltenbrunner, and V. López. Statistical analysis of the social network and discussion threads in slashdot. In *WWW*, 2008.

[14] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Discovering leaders from community actions. In *CIKM*, 2008.

[15] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW*, 2004.

[16] R. Gunning. *The Technique of Clear Writing*. 1952.

[17] R. Hellmann, J. Griesbaum, and T. Mandl. Quality in blogs: How to find the best user generated content. In *Business Information Systems*. 2010.

[18] P. D. Hoff. Bilinear mixed-effects models for dyadic data. *J. of the Amer. Stat. Ass.*, 2005.

[19] L. Hong and Z. Yang. Incorporating participant reputation in community-driven question answering systems. In *Symposium on Social Intelligence and Networking*, 2009.

[20] C.-F. Hsu, E. Khabiri, and J. Caverlee. Ranking comments on the social web. In *International Conference on Computational Science and Engineering*, 2009.

[21] P. G. Ipeirotis, F. Provost, and J. Wang. Quality Management on Amazon Mechanical Turk. *KDD The Human Computation Workshop*, 2010.

[22] T. S. Jaakkola and M. I. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 2000.

[23] Jianshu Weng, et al. Twitterrank: finding topic-sensitive influential twitterers. In *WSDM*, 2010.

[24] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 2007.

[25] Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 2010.

[26] J. Kunegis, A. Lommatzsch, and C. Bauckhage. The slashdot zoo: Mining a social network with negative edges. In *WWW*, 2009.

[27] U. Kuter. Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models. In *AAAI*, 2007.

[28] G. H. M. C. Laughlin. SMOG Grading ŮŰ a New Readability Formula. *Journal Of Reading*, 1969.

[29] J. Liu, et al. Low-Quality Product Review Detection in Opinion Summarization. *EMNLP-CoNLL*, 2007.

[30] Y. Lu and P. Tsaparas, et al. Exploiting social context for review quality prediction. In *WWW*, 2010.

[31] Y. Maeno. Node discovery in a networked organization. In *Intl Conf on Sys., Man and Cyber.*, 2009.

[32] P. Massa and P. Avesani. Trust metrics in recommender systems. In *Computing with Social Trust*, 2009.

[33] M. G. Noll, et al. Telling experts from spammers: Expertise ranking in folksonomies. In *SIGIR*, 2009.

[34] V. C. Raykar, et al. Learning From crowds. *JMLR*, 2010.

[35] P. Resnick and Z. Richard. *Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System*. Elsevier Science, 2002.

[36] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2008.

[37] J. Schneider, et al. Disseminating trust information in wearable communities. In *HUC*, 2000.

[38] H. Shin, et al. Separating the reputation and the sociability of online community users. In *SAC*, 2010.

[39] A. Talwar, R. Jurca, and B. Faltings. Understanding user behavior in online feedback reporting. In *EC*, 2007.

[40] S. Valenti, F. Neri, and A. Cucchiarelli. An Overview of Current Research on Automated Essay Grading. *Journal of Information Technology Education*, 2003.

[41] P. Welinder, S. Branson, S. Belongie, and P. Perona. The Multidimensional Wisdom of Crowds. *NIPS*, 2010.

[42] P. Windley, K. Tew, and D. Daley. A framework for building reputation systems. In *WWW*, 2007.

[43] J. Zhang, et al. Expertise networks in online communities: structure and algorithms. In *WWW*, 2007.

[44] Z. Zheng, H. Zha, K. Chen, and G. Sun. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR*, 2007.

[45] C.-N. Ziegler and G. Lausen. Propagation models for trust and distrust in social networks. *Info. Sys. Frontiers*, 2005.

Appendix: Let R = #observed rating, N_k = #raters and M_k = #authors in context k. The complete data log likelihood of bias-smoothed tensor model is:

$$
\begin{aligned}
2\log\Pr(\mathbf{y},\boldsymbol{\eta}\,|\,\boldsymbol{\Theta}) = {}& \text{ some constant} \\
& - R\,\log\sigma_y^2 - \sum_{ijk}(y_{ijk}-\alpha_{ik}-\beta_{jk}-\langle \boldsymbol{v}_i,\boldsymbol{v}_j,\boldsymbol{w}_k\rangle)^2/\sigma_y^2 \\
& - \sum_k N_k\,\log\sigma_{\alpha,k}^2 - \sum_k\sum_i(\alpha_{ik}-\boldsymbol{g}_k'\boldsymbol{x}_{ik}-q_k\alpha_i)^2/\sigma_{\alpha,k}^2 \\
& - \sum_k M_k\,\log\sigma_{\beta,k}^2 - \sum_k\sum_j(\beta_{jk}-\boldsymbol{d}_k'\boldsymbol{x}_{jk}-r_k\beta_j)^2/\sigma_{\beta,k}^2 \\
& - \sum_i\left(\dim(v)\,\log\sigma_v^2 + \|\boldsymbol{v}_i\|^2/\sigma_v^2\right) - \sum_i\alpha_i^2 - \sum_j\beta_j^2 - \sum_k\|\boldsymbol{w}_k\|^2
\end{aligned}
$$

Use the notation defined in Section 5.5. The expected log likelihood (over $\boldsymbol{\eta}$) is

$$
\begin{aligned}
2\,E_{\boldsymbol{\eta}}[\log\Pr(\mathbf{y},\boldsymbol{\eta}\,|\,\boldsymbol{\Theta})] = {}& \text{ some constant} \\
& - R\,\log\sigma_y^2 - \sum_{ijk}\left((y_{ijk}-\hat{\mu}_{ijk})^2+\hat{V}[\mu_{ijk}]\right)/\sigma_y^2 \\
& - \sum_k N_k\,\log\sigma_{\alpha,k}^2 - \sum_k M_k\,\log\sigma_{\beta,k}^2 \\
& - \sum_k\sum_i\frac{(\hat{\alpha}_{ik}-g_k'x_{ik}-q_k\hat{\alpha}_i)^2+\hat{V}[\alpha_{ik}]-2q_k\hat{V}[\alpha_{ik},\alpha_i]+q_k^2\hat{V}[\alpha_i]}{\sigma_{\alpha,k}^2} \\
& - \sum_k\sum_j\frac{(\hat{\beta}_{jk}-d_k'x_{jk}-r_k\hat{\beta}_j)^2+\hat{V}[\beta_{jk}]-2r_k\hat{V}[\beta_{jk},\beta_j]+r_k^2\hat{V}[\beta_j]}{\sigma_{\beta,k}^2} \\
& - \sum_i\left(\dim(v)\,\log\sigma_v^2+\left(\hat{\boldsymbol{v}}_i'\hat{\boldsymbol{v}}_i+\mathrm{trace}(\hat{V}[\boldsymbol{v}_i])\right)/\sigma_v^2\right)
\end{aligned}
$$