A Few Bad Votes Too Many? Towards Robust Ranking in Social Media

Jiang Bian
College of Computing
Georgia Institute of Technology
jbian@cc.gatech.edu

Yandong Liu
Math & Computer Science
Emory University
yliu49@emory.edu

Eugene Agichtein
Math & Computer Science
Emory University
eugene@mathcs.emory.edu

Hongyuan Zha
College of Computing
Georgia Institute of Technology
zha@cc.gatech.edu

ABSTRACT

Online social media draws heavily on active reader participation, such as voting or rating of news stories, articles, or responses to a question. This user feedback is invaluable for ranking, filtering, and retrieving high quality content - tasks that are crucial with the explosive amount of social content on the web. Unfortunately, as social media moves into the mainstream and gains in popularity, the quality of the user feedback degrades. Some of this is due to noise, but, increasingly, a small fraction of malicious users are trying to "game the system" by selectively promoting or demoting content for profit, or fun. Hence, an effective ranking of social media content must be robust to noise in the user interactions, and in particular to vote spam. We describe a machine learning-based ranking framework for social media that integrates user interactions and content relevance, and demonstrate its effectiveness for answer retrieval in a popular community question answering portal. We consider several vote spam attacks, and introduce a method of training our ranker to increase its robustness to some common forms of vote spam attacks. The results of our large-scale experimental evaluation show that our ranker is significantly more robust to vote spam compared to a state-of-the-art baseline as well as the ranker not explicitly trained to handle malicious interactions.

Categories and Subject Descriptors

H.3.3 [**Information Search Retrieval**]: Relevance feedback, Search process; H.3.5 [**On-line Information Services**]: Web-based services

General Terms

Algorithms Measurement Experimentation

Keywords

Social Media, Ranking, Robustness, Vote Spam, Community Question Answering

1. INTRODUCTION

Social media sources provide an effective alternative to traditional web search by directly connecting users with the information needs to users willing to share the information. For example, users can post questions or news items, and rely on other users to comment or rank the content (e.g., sites such as Slashdot or Digg). While the responses could be excellent, the quality could vary greatly. Hence, user feedback, such as voting, or rating the content, has become a crucial aspect of the effectiveness of the community. For example, in community question answering (e.g. Yahoo! Answers [1]), users can give thumbs up or down votes to existing answers; while in social news and videos sharing services such as Digg [2] and Youtube, votes are used to judge the quality of the posted news or videos, as well for the quality of the comments.

A very successful case of a community organized around information needs (questions) and answers is Yahoo! Answers. In this *community question answering* portal (henceforth CQA) users can express specific information needs by posting questions in order to obtain answers authored by other web users. In addition, existing answers can be voted on by any user who wants to share her evaluation of the answers. All the questions and the answers are stored for future use and are eventually incorporated into web search results. Unfortunately, the quality of the content in this kind of QA portals varies drastically, and a large portion of the content is not useful for answering user queries [2]. Not surprisingly, user votes can provide crucial indicators into the quality and reliability of the content. For example, in Yahoo! Answers, more than half of the so called "best answers" to a question are chosen as the most popular answers according to the user votes. Our recent work [3] showed that incorporating user vote information can significantly improve the quality of a ranker over the CQA archives.

Unfortunately, not all user votes are reliable. Many "thumbs up" or "thumbs down" votes are generated without much thought, and, in some cases, by users intending to game the system – i.e., to promote specific answers or questions for fun or profit. We refer those bad or fraudulent votes as *vote spam*. We posit that vote spam is an increasingly common phenomenon in social media sites, and deserves explicit handling for robust ranking of social media content. In the specific case of Yahoo! Answers, the support team already semi-automatically removes some of the more obvious vote spam after the fact. We believe that this solution may not prove adequate in the long run: as the amount and the patterns of vote spam evolve, post-factum filtering could significantly degrade user experience until the product team had a chance to react to a new spam attack. To complicate this problem, vote spam methods can change significantly due to varying popularity of content, specifics of media and topic and as spammers adjust their methods. Therefore, we need a robust method to train a ranking function that remains resilient to evolving vote spam attacks.

In this paper we consider how to modify a recently presented algorithm for ranking social media [3] to make it more resilient to some common forms of vote spam. In order to

[1] http://answers.yahoo.com/
[2] http://digg.org/

AIRWeb '08, April 22, 2008 Beijing, China.

investigate the robustness of our method, we consider several common vote spam methods in social media. In particular, we focus on the specific case of CQA to explore the influence of vote spam on the quality of answer retrieval. Our specific contributions include:

- A parameterized vote spam model to describe and analyze some common forms of vote spam (Section 4)
- A method for increasing the robustness of ranking by injecting noise at training. (Sections 5).
- A comprehensive evaluation on ranking performance for community question answering, under a variety of simulated vote spam attacks, demonstrating robustness of our ranking (Section 6).

2. RELATED WORK

Social media services provide popular web applications such as photo sharing(Flickr), social bookmarking(Delicious), video sharing(Youtube), and, more recently, popular community Question Answering sites such as Yahoo! Answers. Question answering over community QA archives is different from traditional TREC QA [19], and applying QA techniques over the web [4]. The most significant difference is that traditional QA operates over a large collection of documents (and/or web pages) whereas we are attempting to retrieve answers from a social media archive with a large amount of associated user-generated metadata [2]. This metadata (such as explicit user feedback on answer quality) is crucial due to the large disparity of the answer quality, as any user is free to contribute his or her answer for any question.

Due to the explosive rise in popularity of Yahoo! Answers and other sites, community QA has recently become an active area of research. Jeon et al. [8] presented retrieval methods based on machine translation models to find similar questions from a community QA service, but did not take the quality of answers into consideration. Su et al. [18] analyzed the quality of answers in QA portals and found that the quality of each answers varies significantly. Jeon et al. [9] built a model for the quality of answers based on features derived from the specific answer being analyzed. Recently, Agichtein et al. [2] explored user interaction and content-based lexical features to identify high-quality content, and Bian et al. [3] presented a ranking framework to utilize user interaction information to retrieve high quality relevant content in social media. Our work diverges from reference [2] and [3] in that we explicitly consider the effect of malicious user interactions. and focus on modifying the ranking algorithm to be more resilient to vote manipulation or "shilling".

Our work is also related to integrating user interactions and feedback into web search [10, 11, 12, 1]. For example, implicit feedback in the form of result clickthrough was shown to be helpful for web search ranking. One of the serious problems when integrating user interactions to web search is click spam [7]. Many studies have analyzed robustness of web search ranking to click spam. Radlinski et al. [17] presented how click noise/spam bias the ranking results. Jansen [7] revealed the influence of malicious clicks on online advertising search and Metwally et al. [14] explored how to identify fraudulent clicks on advertisements. After a deep analysis on click fraud in online advertising, Immorlica et al. [6] demonstrated that a particular class of learning algorithms are resistant to click fraud in some sense. Radlinski et al. [16] analyzed click spam from a utility standpoint and investigated whether personalizing web search results can reduce spam. Several previous studies explored interactions spam in social systems. Heymann et al. [15] surveyed the approaches and challenges for fighting spam on social web sites. Mehta et al. [13] provided an algorithm for detecting spam in collaborative filtering. In this paper, we focus on a different setting of ranking in social media, and consider general methods of vote spam which exhibit distinct attack methods and characteristics from click fraud. We also explicitly validate the robustness of our method for ranking in a community question answering setting.

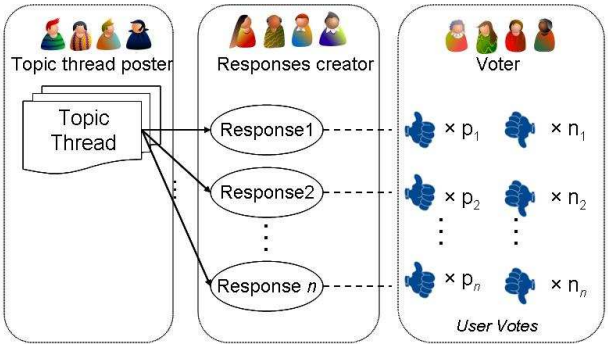

Figure 1: Illustration of social content and user votes in social media service: Users can post topic threads on social media sites Topic thread poster; Users can also submit responses to topic threads Response creator; Many social media services allow users to vote for existing responses using "thumb up" or "thumb down".

3. LEARNING RANKING FUNCTIONS IN SOCIAL MEDIA

Ranking functions are at the core of an effective search over social media. We present a learning-based approach ranking. In social media services, users can post some interesting topic threads, such as questions(Yahoo! Answers), news(Digg) or videos(Youtube), onto social media sites. These topic threads will trigger responses from the users who supply related information. Furthermore, users can give thumbs up or down votes to those responses, which present users' positive or negative judgment on the quality of those responses. For example, in Community Question Answering (CQA), a user can vote on existing answers. We summarize the structure of social content and user votes in social media in Figure 1.

In this paper, we focus on the specific characteristics of social media and discuss how to employ user interactions, especially *user votes*, to rank the users' responses. We employ the similar framework in our previous work [3]. In the rest of this section, we discuss how to represent textual and community elements in social media as features. Based on the extracted features and preference data, we apply the regression-based gradient boosting framework [20, 5] to learn the ranking. The material in this section summarizes our recently presented GBrank system described in [3].

3.1 Features and Preference Data Extraction

We represent each query-topic-response triple (qr, tp, rp) as a combination of textual features and user interaction features.

Textual Elements: We consider the topic threads, user responses and queries (showed in Figure 1). We represent each of these elements independently, using features such as "number of tokens for a query", "how long has the topic thread been posted", "number of received votes for a response", etc. Then, we also extract textual features from relationship between topic threads, user responses and queries, describing similarity between these three elements. For example, the number of overlapping terms and token number ratio between two of these three elements, etc.

User Interaction: As discussed before, there are three kinds of roles each user may play in a social system, namely topic thread poster, response creator and voter. Figure 1 shows the interactions between these three roles. For each user in the community of a social media, there are several features to describe his or her activities, such as "the number of topics he or she posted", "the number of responses he or she created", etc. These features to certain extent can approximate the user's expertise in the social media community. And user's expertise can in turn indicate the quality of his or her responses to the topic threads. Similarly reputation of topic posters and voters can also indicate quality of responses.

3.1.1 User Ratings as Preferences

as shown above, in many social media services, user evaluation in the form of *user votes* is an additional important type of user interaction/feedback. It is represented as the "thumbs up" and "thumbs down" metaphors and implies users judgment of the quality of content in social media.

We examine user vote data and extract a set of preference data which can be used for ranking the responses as follows. For each query qr, under the same topic thread tp, we consider two existing responses rp_1 and rp_2. Assume that in the user vote data, rp_1 has p_1 thumbs up votes and m_1 thumbs down votes out of n_1 impressions while rp_2 has p_2 thumbs up votes and m_2 thumbs down votes out of n_2 impressions. We want to consider response pairs rp_1 and rp_2 to see whether rp_1 is preferred over rp_2 in terms of their relevance to the topic tp. In general, since different users vote for response rp_i independently, we assume that the number of thumbs up votes p_i in a sequence of n_i impressions obeys binomial distribution, showed as following:

$$B(p_i; n_i, p) = \left(\begin{array}{c} n_i \\ p_i \end{array} \right) p^k (1-p)^{n_i - p_i}$$

where p is the probability users post the thumbs up vote. We utilize binomial distribution to model the number of votes because it is more effective and convenient to extract significant pairs than other kinds of distribution, e.g. uniform distribution.

We use the approach in [20] and apply likelihood ratio test to examine whether a pair of answers is significant or not, i.e., whether there are enough votes to compare the pair. In particular we compute the following statistic,

$$\lambda = \frac{B(p_1+p_2; n_1+n_2, (p_1+p_2)/(n_1+n_2))}{B(p_1; n_1, p_1/n_1) B(p_2; n_2, p_2/n_2)} \rightarrow -\chi^2$$

For a pair of response rp_1 and rp_2, when the above value is greater than a threshold, we say the pair is significant. If rp_1 and rp_2 form a significant pair, we extract preference data by comparing $\frac{p_1}{p_1+m_1+s}$ with $\frac{p_2}{p_2+m_2+s}$, where s is positive constant, i.e., if the former value is bigger than the later one, then we say rp_1 is preferred over rp_2 which is denoted by $rp_1 \succ rp_2$, and vice versa.

3.2 Learning Ranking Function from Preference Data

Once the features and preference data are extracted, the next question is how to use them for the purpose of learning a ranking function for social media. We apply a framework for solving ranking problems from preference data [20]. This framework proposes a new objective function for learning ranking function using preference data and develop an algorithm that adapts functional gradient descent for optimizing the proposed objective function. The idea behind the algorithm in [20] is as follows: for each extracted preference, if the ordering of the current ranking function contradicts the extracted preference, we need to modify the value of ranking function to force the ranking function to agree with the preference as much as possible.

Suppose the set of available preferences is

$$\mathcal{S} = \{\langle x_i, y_i \rangle \mid x_i \succ y_i, i = 1, ..., N\}$$

For each $\langle x, y \rangle \in S$, x, y denote the feature vectors for two query-topic-response triples with the same query. $x \succ y$ means that x is preferred over y, i.e. x should be ranked higher than y. In other words, the answer in x is considered more relevant than that in y with respect to the same query in both triples.

In [20], the problem of learning ranking functions as computing a ranking function h, such that h match the set of preferences, i.e., $h(x_i) \geq h(y_i)$, if $x_i \succ y_i$, $i = 1, ..., N$ as much as possible. The following objective function is then used to measure the risk of a given ranking function h,

$$\mathcal{R}(h) = \frac{1}{2} \sum_{i=1}^{N} (\max\{0, h(y_i) - h(x_i)\})^2$$

We summarize the algorithm for learning ranking function h using gradient boosting as in Figure 2. Two parameters need to be determined: the shrinkage factor and the number of iterations, this is usually done by cross-validation [20].

Algorithm GBrank:
Start with an initial guess h_0, for $k = 1, 2, ...$

1. Using h_{k-1} as the current approximation of h, we separate $\mathcal{S}$ into two disjoint sets,

$$\mathcal{S}^+ = \{\langle x_i, y_i \rangle \in \mathcal{S} | h_{k-1}(x_i) \geq h_{k-1}(y_i) + \tau\}$$

$$\mathcal{S}^- = \{\langle x_i, y_i \rangle \in \mathcal{S} | h_{k-1}(x_i) < h_{k-1}(y_i) + \tau\}$$

2. Fit a regression function $g_k(x)$ using Gradient Boosting Tree[9] and the following training data

$$\{(x_i, h_{k-1}(y_i) + \tau), (y_i, h_{k-1}(x_i) - \tau) | \langle x_i, y_i \rangle \in \mathcal{S}^-\}$$

3. Form the new ranking function as

$$h_k(x) = \frac{k h_{k-1}(x) + \eta g_k(x)}{k+1}$$

where η is a shrinkage factor.

Figure 2: GBrank Algorithm.

4. VOTE SPAM IN SOCIAL MEDIA

As discussed before, user votes are valuable to evaluate the quality of user responses related to topic threads. In our consideration, there are two main types of vote spam in social media: incorrect votes and malicious votes. The user who gives the votes may not be an expert to the topic thread and related responses, therefore it is likely that its votes are incorrect. In another case, some malicious users intend to promote some specific responses within the community of social media, and they attack the social media service by creating a number of thumbs up vote to the specific responses. For example, in order to do online advertising in social media services, malicious users post their advertisement into responses to some topic threads and promote those responses by introducing amount of thumbs up votes. In another way, they can submit thumbs down votes to decrease the rank of high quality responses.

4.1 Vote Spam Attack Models

In this paper, we will focus on the influence of malicious vote attack and analyzing the robustness of our ranking framework. In the rest of this section, we will introduce a general vote spam model in the social media service.

We simulate the vote spam attack as following: Given the whole set of topic threads is $TP = \{tp_1, tp_2, \cdots, tp_m\}$, we assume that $\beta\%$ of them are attacked by malicious votes. As the goal of vote spam is to promote advertisement or other information from malicious users, those attackers tend to post and promote some the specific responses under popular topic threads. Thus, if a topic thread is more popular, i.e. followed by much more responses, it is more likely to be attacked. In this paper, the probability that a topic thread is included in $\beta\%$ attacked thread set is proportional to the number of its responses.

As the set of topic threads to be attacked has been selected, the number of attackers to each topic thread may be different. In our approach, we assume Gaussian distribution to simulate the number of malicious users for each topic thread. We use N_i to denote the number of attackers to the topic thread tp_i, then

$$N_i \sim \mathcal{N}(\mu, \sigma^2)$$

Note that we can describe various number of attackers by simply changing the value of μ in Gaussian distribution. Other methods can also be used to simulate the number of malicious users.

For each topic thread to be attacked, we consider two general attack strategies, *thumbs up votes spam* and *thumbs up&down votes spam*. For *thumbs up votes spam*, malicious users aim to promote single response for one topic thread, so that they will propose thumbs up votes to the specific responses as many as possible. Using *thumbs up&down votes spam*, malicious users will give thumbs down votes to other responses in addition to thumbs up vote for one specific responses. In this way, attackers can promote specific responses by decreasing the ranking of others.

The next question is how many thumbs up or down votes malicious users will "contribute" for a particular topic. Most community portals and social media services enforce strict budget rules for user votes which constrain the number of votes one user can give. In our simulated attack approach, we assume an unlimited overall voting budget for a user, but include a common restriction that any user can vote (thumbs up or down) at most once for each item.

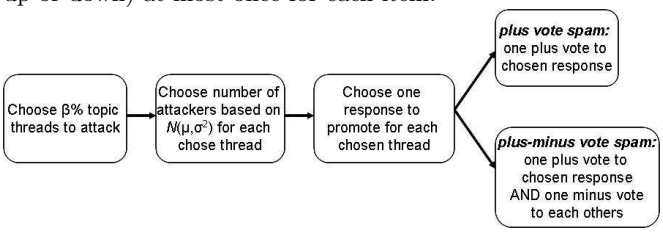

Figure 3: Summary of the stochastic vote spam generation process.

Figure 3 summarizes our attack models: First, we choose β% topic threads to attack; Then, the number of attackers for each attacked thread is decided based on Gaussian distribution; After selecting one response to promote for each attacked thread, we can choose one attack strategy which is either *plus vote spam* or *plus-minus vote spam*. In the next section, we will study Community Question Answering(CQA) service. We use the proposed framework for learning ranking function for QA retrieval as well as evaluate the robustness of learned ranking against vote spam.

5. COMMUNITY QUESTION ANSWERING

In this paper, we will focus on the specific characteristics of social Question Answering services. In the rest of this section, we will discuss how to employ CQA specified features especially user votes for ranking as well as how to evaluate the robustness of QA retrieval against vote spam.

5.1 Learning to Rank for Community QA

Community QA is a particularly popular form of social media, drawing millions of users to ask and answer each others' questions. Similar to other social media services, community QA services such as Yahoo! Answers provide several types of interactions among users that are specific to the QA domain. Users in Yahoo! Answers do not only ask and answer questions, but also actively participate in regulating the system. A user can vote for answers of other users, mark interesting questions and even report abusive behavior.

In order to define the problem of QA retrieval, we first abstract the social content in the QA system as a set of question-answer pairs:

$$\langle qst_i, ans_i^j \rangle$$

where qst_i is the ith question in the whole archive of the QA system and ans_i^j is the jth answer to this question. Given a user query, our goal is to order the set of QA pairs according to their relevance to the query, and the ordering is done by learning a ranking function for triples of the form,

$$\langle qr_k, qst_i, ans_i^j \rangle,$$

where qr_k is the k-query in a set of queries.

In our approach for learning QA ranking function, we will first extract textual features related to question, answer and query as well as social features related to user interactions described above. In addition, we also extract preference data from user votes. Then, based on the learning framework discussed in Section 3.2, we will obtain the ranking function for QA retrieval.

5.2 Robust Ranking Method

As shown in Section 6 and our previous work [3], our ranking function (*GBrank*) exhibits promising performance on QA retrieval. Our experiments demonstrate that user vote information provides much contribution to the high accuracy of our *GBrank*, when there is no vote spam. However, if user votes in CQA have been polluted by spam from malicious users and we continue using *GBrank* trained by clear data without vote spam, *GBrank* will still put much reliance on user vote information which however is supplying inaccurate information due to the spam.

In order to create a robust ranking method, we enhance our *GBrank* by using polluted training data during learning process. We apply the general vote spam model, described in Section 4, to generate vote spam into unpolluted QA data. Then, we train the ranking function based on new polluted data. To distinguish with *GBrank* trained by clear data, we denote our new ranking function as *GBrank-robust*. *GBrank-robust* is able to automatically account for the observed noise in the preference features, transferring more weight to other content and community features. Therefore, *GBrank-robust* can outperform the original *GBrank* method where simulated vote spam was not introduced at training time.

In the rest of this section, we will present our evaluation setup. First we describe our dataset including the queries and the corresponding corpus of questions and answers. Then we describe our evaluation metrics and the ranking methods to compare for the experimental results reported in Section 6.

5.3 Datasets

Factoid questions from the TREC QA benchmarks We use factoid questions from seven years of the TREC QA track evaluations (years 1999–2006)[3] for the experiments reported in Section 6. It is worth noting that TREC questions from the years 1999 to 2003 are independent of each other: each question is self-contained and we submit directly as the query. Starting from 2004, however, the questions are organized in groups with a 'target'. For those questions, we submit their 'target' as well as the questions themselves. In total, approximately 3,000 factoid TREC questions were compiled as the initial set of queries.

Since we need *some* candidate answers from Yahoo! Answers to estimate how well different ranking functions perform, we select the 1250 TREC factoid questions that have at least one similar question in the Yahoo! Answers archive.

Question-answer collection dataset Our dataset was collected in order to simulate a user's experience with a community QA site. We submit each TREC query to the Yahoo! Answers web service[4] and retrieve up to 10 top-ranked related questions according to the Yahoo! Answers ranking. For each of these Yahoo! questions, we retrieve as many answers as there are available for each question thread. There are, in total, 89642 $\langle query, question, answer \rangle$ tuples. 17711 tuples (19.8%) are labeled as "relevant" while 71931 (81.2%) are labeled as non-relevant.

Relevance Judgments In our experiment, the data are labeled in two ways: by using the TREC factoid answer patterns, and, independently, manually in order to validate the pattern-based automatic labels. For automatic relevance labels, we check every answer's text body, and if the text matches one of the answer patterns, we consider the answer text to be relevant, and non-relevant otherwise. In order to validate the accuracy of our automatically-assigned relevance labels, we in-

[3]http://trec.nist.gov/data/qa.html
[4]http://developer.yahoo.com/answers/

dependently labeled a number of answers by hand. The manually labeled answers were compared with the automatically generated labels, resulting in over 90% agreement between the automatic and manual methods. In summary, automatically generated labels, even though with some small degree of noise, nevertheless exhibit high agreement with manual relevance judgments, and serve as a good proxy for comparing rankings.

Vote Spam Recall that there may already be some spam in the original data, but it is limited as we are experimenting with past data that has been cleaned to large extent by the Yahoo! team. Therefore, we need to inject vote spam by ourselves. The model of vote spam has been discussed in Section 4. We will describe the parameters of the settings of vote spam in Section 5.6.

5.4 Evaluation Metrics

We adapt the following information retrieval metrics to evaluate the performance of the ranking function. Note that none of the metrics require a single best answer for a question. Rather, any correct answer is included in the metric computation.

- **Mean Reciprocal Rank(MRR)**: The MRR of each individual query is the reciprocal of the rank at which the first relevant answer was returned, or 0 if none of the top N results contained a relevant answer.The score for a sequence of queries is the mean of the individual query's reciprocal ranks. Thus, MRR is calculated as

$$MRR = \frac{1}{|Qr|} \sum_{q \in Qr} \frac{1}{r_q}$$

  where Qr is a set of test queries, r_q is the rank of the first relevant document for q.

- **Precision at K**: for a given query, $P(K)$ reports the fraction of answers ranked in the top K results that are labeled as relevant. In our setting, we require a relevant answer to be labeled "matched" for TREC pattern. For this metric, the position of relevant answers within the top K is irrelevant, while it measures overall user potential satisfaction with the top K results.

- **Mean Average of Precision(MAP)**: Average precision for each query is defined as the mean of the precision at K values calculated after each relevant answers was retrieved. The final MAP value is defined as the mean of average precisions of all queries in the test set. This metrics is the most commonly used single-value summary of a run over a set of queries. Thus, MAP is calculated as

$$MAP = \frac{1}{|Qr|} \sum_{q \in Qr} \frac{\sum_{r=1}^{N} (P(r) \times rel(r))}{|R_q|}$$

  where Qr is a set of test queries, R_q is the set of relevant document for q, r is the rank, N is the number retrieved, $rel()$ is a binary function on the relevance of a given rank, and $P()$ is precision at a given cut-off rank.

5.5 Ranking Methods Compared

To evaluate the QA retrieval quality, we compare the quality of following methods:

- **Baseline**: In this method, the answers are ranked by the score computed as the difference of positive votes and negative votes received for each answer. This ranking closely approximates the ranking obtained when a user clicks "Order by votes" option on the Yahoo! Answers site.

- **GBrank**: Ranking function with textual and community/social features: this is our method presented in Section 3.2.

- **GBrank-robust**: Similar to GBrank, we utilize textual and community features to train ranking function. However, the training data is polluted according to the chosen spam model. We will discuss how to evaluate vote spam's influence on QA retrieval in Section 5.6.

Since our data set is constructed automatically by querying Yahoo! Answers for TREC questions, we need to adjust the computation of the metrics for the Baseline method to match the user experience, if she had submitted such a question to the Answers' search service. The adjustment is necessary as there may be multiple Yahoo! Answers questions retrieved, and (presumably) the user would select the most relevant one. To match this experience, if our test question retrieves multiple Y!A question threads, we take the *maximum* of the MRR (or MAP or Precision) values for each thread. Hence, in a sense our Baseline is particularly strong in that it assumes a perfect ranking of the question threads (more generally, topics) by the Yahoo! Answers search engine.

5.6 Evaluation on Robustness of Ranking to Vote Spam Attack

To evaluate the robustness of proposed QA ranking algorithm, we compare the performance of the ranking algorithm in the situation with vote spam and without spam. In detail, we consider the following settings of vote spam and evaluate their influence on performance of ranking respectively:

- *Scope of Vote Spam*: The scope of vote spam is measured by the percentage of attacked question threads (β). We will compare the performance of ranking under vote spam attack when the number of attacked topic threads ($\beta\%$) is of different value.

- *Number of Attackers*: We will compare the performance of ranking under vote spam attack when the number of attacker varies. In our paper, the number of attacker for each question thread obey Gaussian distribution(Section 4.1). In this paper, we fix the variance σ^2 in Gaussian distribution and model the number of attackers by using different mean μ.

- *Attack Strategy*: We will compare the performance of ranking under two different strategies of vote spam attack: *thumbs up vote spam* and *thumbs up&down vote spam*. In this first strategy, malicious users promote one specific answer only by adding thumbs up votes to it. In the second one, attackers submit not only thumbs up votes to the specific answer but also thumbs down votes to the other answers in the same question thread.

6. EXPERIMENTAL RESULTS

In this section, we will describe our large scale experiments. These experiments are used to demonstrate that 1) *GBrank* method is effective to learn ranking function for social media; 2) how vote spam influence ranking performance; 3) *GBrank-robust* is robust to vote spam in certain scope and 4) how various features affect *GBrank-robust*'s robustness to vote spam.

6.1 Learning Ranking Function

To learn the ranking function, we generate the training and testing data as follows: we randomly select 800 TREC queries from total 1250 TREC queries and collect all the related QA for these 800 queries. As mentioned in Section 5.3, the relevance judgments was obtained by matching an answer with TREC answer patterns. We have also found that 90% of items were given the same labels under both manual labeling and TREC pattern labeling and the remaining 10% of automatic labels were erroneous. And we use ten-fold cross validation to perform the training of the proposed ranking function using the algorithm introduced above.

To discover the convergence behavior of GBrank, we measure the performance for the hold-out validation data against each iteration of our learning algorithm. We find that the algorithm converges after 60 iterations. Thus, for the following experiments, we stop at 60 iterations for training GBrank.

Table 1: MRR and MAP for GBrank and Baseline with clean training and testing data

	MRR	MAP
Baseline	0.662	0.441
GBrank	**0.782**	**0.465**

In Table 1, we illustrate the MAP and MRR scores for the baseline method as well as *GBrank*. There is no vote spam in training or testing data.

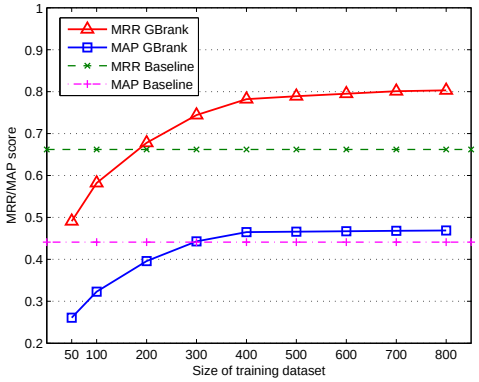

Figure 4: Mean Reciprocal Rank(MRR) and Mean Average Precision(MAP) for GBrank for varying the training set size.

We vary the training size for learning the ranking function. Figure 4 reports the MRR and MAP scores for the hold out validation data against different size of training dataset. We can see that MRR and MAP scores increases when the training dataset become larger. The scores remain stable after training size is larger than about 400. It is also clear to see that GBrank outperform than baseline method both for MAP and MRR when there are more than 300 questions for training.

6.2 QA Retrieval

In this experiment, we investigate the performance of QA retrieval under the attack of vote spam.

In order to simulate vote spam, we use the method described is Section 4.1 to add vote spam in dataset: we first sample 10% question threads to be attacked, denoted as $\{q_1, q_2, \cdots, q_s\}$. Then, we assume the number of attackers obeys the Gaussian distribution whose mean is 3 and variance is 1 and sample attacker numbers for each attacked thread, denoted as $\{N_1, N_2, \cdots, N_s\}$ respectively. In each sampled question thread q_i, we randomly select one answer which will be promoted by malicious users. And we use *thumbs up vote spam* strategy for attacking, i.e. adding N_i thumbs up votes to the specific answer in q_i respectively. In the following experiment, we use this attack model by default.

In our experiment, we train two ranking functions, *GBrank-robust* and *GBrank*, on training data (i.e., the 800 TREC queries) with vote spam and training data without vote spam respectively. The remainder hold-out testing data (i.e. 450 TREC queries and associated community QA pairs) is added with vote spam. Then, using the polluted testing data, we evaluate the performance for two ranking functions and baseline method.

Figure 5 illustrate the Precision at K of *GBrank* and *GBrank-robust* compared with the baseline method. Testing data for all the three metrics are extracted from polluted dataset. (Vote spam model: $\beta\% = 10\%$, $\mu = 3$ and $\sigma^2 = 1$). This figure shows that, under the situation that test data is polluted with vote spam, *GBrank-robust* still performs better than baseline method, but *GBrank* does not obtain better performance than baseline.

In Figure 5, we also demonstrate the Precision at K of *GBrank* and *GBrank-robust* when they are evaluated using unpolluted testing data. It is clear to see that *GBrank-robust* is "*graceful*" in that it is not much worse than original *GBrank* when there is actually no vote spam.

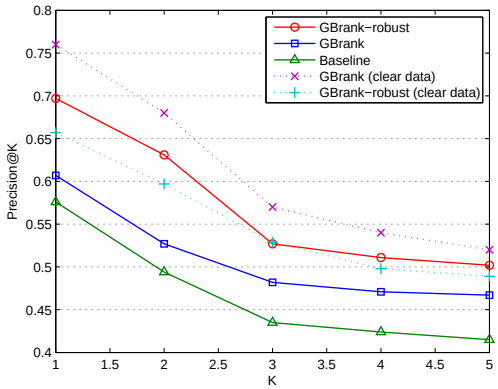

Figure 5: Precision at K for Baseline, GBrank and GBrank-robust for various K.

Table 2: Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP) for GBrank, GBrank-robust and Baseline

	MRR	MAP
GBrank	0.648	0.416
Baseline	0.624	0.405
GBrank-robust	**0.736**	**0.457**

In Table 2, we illustrate the MAP and MRR scores for the baseline method as well as *GBrank* and *GBrank-robust*. From the table, it is clear that *GBrank-robust* reaches much better performance than *GBrank* and baseline method. In particular, *GBrank-robust* still achieves a gain of about 15% relative to the baseline.

To understand how two GBrank functions can outperform an "oracle" baseline, consider that the ordering of answers within a question thread remains fixed (either by date – as the default, or by decreasing votes). In contrast, GBrank obtains a better ranking of answers within each question thread, as well as a global ranking of all answers. Then, improved ranking within each Yahoo questions thread contributes to the higher score than baseline. Overall, applied on Yahoo! Answers, our proposed framework achieves a significant improvement on the performance of QA retrieval over the Yahoo! Answers' default ranking and the supported optional votes-based ranking. In addition, from the experiment, we can find that our method is able to retrieve relevant answers at the top of results. In summary, we have shown that *GBrank-robust* significantly outperforms extremely strong baselines, achieving precision at 1 of nearly 70% and MRR of over 0.73, which are high values even for traditional QA retrieval.

6.3 Robustness to Vote Spam

In this section, we perform experiments to evaluate the robustness of our ranking function to user vote spam. As discussed in Section 4.1, there are three parameters to decide the model vote spam attack: scope of vote spam, number of attackers and attack strategy. We will also assess the vote spam influence on our ranking function under various settings of attack model.

In the following, we illustrate the influence of vote spam on *GBrank-robust* under various settings of attack model. Note that *GBrank-robust* is trained using default vote spam model ($\beta\% = 10\%$, $\mu = 3$ and $\sigma^2 = 1$). In particular, we evaluate our model's sensitivity to various parameter settings by using the testing data polluted by different vote spam model.

1. Number of attackers

First, we explore whether the number of attackers for each question thread affects *GBrank-robust* performance. As discussed above, we model number of attackers for each thread as Gaussian distribution($\mathcal{N}(\mu, \sigma^2)$). In our experiment, we fix σ^2 to 1 and only use *thumbs up vote spam* for the testing data. The other parameters remain default values shown

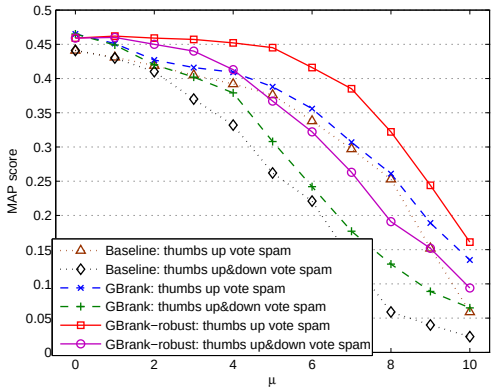

Figure 6: MAP scores for GBrank-robust, GBrank and Baseline for various mean number of attackers. We calculate the scores for thumbs up vote spam and thumbs up&down vote spam respectively.

above. Figure 6 illustrate the MAP scores of *GBrank-robust*, GBrank and baseline method under vote spam with different number of attackers for the test data, i.e. various value of μ. These figures shows that the performance of *GBrank-robust* outperform than both GBrank and baseline when there are vote spam in testing data. Although it declines as the average number of attackers increases, *GBrank-robust*, as we can see, is more robust to the vote spam.

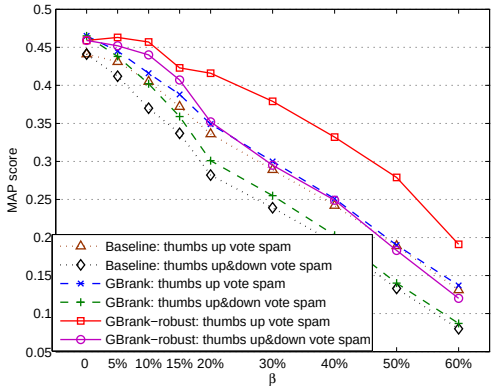

Figure 7: MAP scores for GBrank-robust and Baseline for various scope of vote spam. We calculate the scores for thumbs up vote spam and thumbs up&down vote spam respectively.

2. Scope of vote spam

Second, we investigate whether the scope of vote spam influences *GBrank-robust* performance. In our experiment, we use both *thumbs up vote spam* and *thumbs up&down vote spam* strategies and the other parameters remain default values except percentage of attacked question threads. Figure 7 illustrate MAP scores of *GBrank-robust* on the testing data which are polluted by different scope of vote spam, i.e. various value of $\beta\%$. It is showed that the performance of *GBrank-robust* outperform than both GBrank and baseline when there are vote spam in testing data. Although it declines while the scope of vote spam arises, *GBrank-robust*, as we can see, is more robust to the vote spam.

3. Attack Strategy

In addition, to gain understanding of how different spam strategies achieve different decreasing on performance, we perform the same experiment and illustrate the results in Figure 6 and 7. From these figures, it is obvious to see that *thumbs up&down vote spam* can cause more serious loss on ranking performance than *thumbs up vote spam*. We consider the reason behind is that *thumbs up&down vote spam* probably disorder much more the correct preference as this strategy gives more vote spam than *thumbs up vote spam*.

6.4 Analyzing Feature Contributions

To gain a better understanding of the important features for this domain we perform an ablation study on our feature set to explore which features are significant to answers ranking.

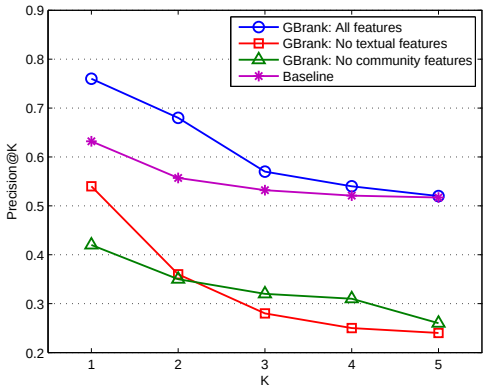

Figure 8: Precision at K for feature ablation study using clean data (GBrank)

As discussed in Section 5.1 there are two major categories of our feature set: textual features and community features. Figure 8 reports the Precision at K when learning ranking function with removing each category respectively. It is easy to see that the performance of *GBrank* is worse than baseline method when neglecting either category of features. Interestingly, textual features are less important for Precision at 1. We hypothesize that for the top result it is more important for an answer to be chosen as "best" by the asker (one of the community features), than to have appropriate textual characteristics.

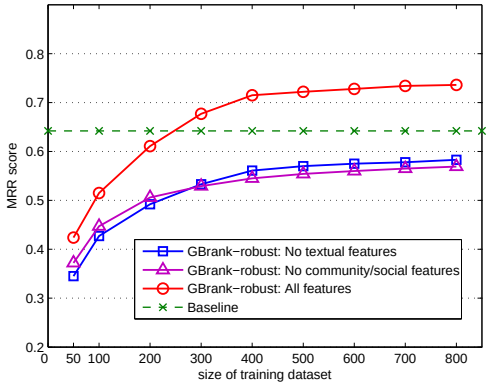

Figure 9: Mean Reciprocal Rank(MRR) for GBrank-robust with different size of training data on feature ablation study. The MRR for baseline using polluted data is also shown in the figure.

In addition, we also carry out an ablation study when training and testing data have been polluted by vote spam. Figure 9 illustrates the MRR scores when learning ranking function with all features or removing one of categories, and the learning is based on various size of polluted training data. The MRR score of baseline method is presented in the figure as well. From the figure, we can find that removing either textual features or community features cause a significant degradation of performance, especially for small amounts of training data. Even in the presence of vote spam, our ranker with all of the features is able to outperform the baseline after only 300 questions in the training set. In summary, we have shown that our ranker is both effective and robust to a variety of vote spam methods.

We evaluate the feature contributions for all the features both when training and testing data are polluted by spam and when neither is polluted. Table 4 shows the Information

Gain for all features when training and testing data have been polluted by vote spam; While Table 3 shows the Information Gain for all features when training and testing data have been polluted by vote spam.

Table 3: Information Gain for all features both when training and testing data are not polluted

Info Gain	Feature Name
0.048	similarity between query and question
0.045	number of resolved question for answerer
0.043	length ratio between query and answer
0.032	number of thumbs down vote
0.030	number of stars for answerer
0.021	number of thumbs up vote
0.014	similarity between query and qst + ans
0.013	number of answer terms
0.013	number of question asked by answerer
0.011	answer's lifetime

Table 4: Information Gain for all features both when training and testing data are polluted

Info Gain	Feature Name
0.048	similarity between query and question
0.045	number of resolved question for answerer
0.043	length ratio between query and answer
0.029	number of stars for answerer
0.026	similarity between query and qst + ans
0.018	number of answer terms
0.013	number of question asked by answerer
0.011	answer's lifetime
0.010	number of question terms
0.009	length ratio between query and question
...	...
0.003	number of thumbs down vote
0.002	number of thumbs up vote
...	...

From these two tables, we can find that (1) some textual features and community features have much influence on the ranking function. (2) When there is no spam in training and testing data, user vote information is important to ranking results. (3) However when incorporating vote spam in training and testing data, both of them contribute much less than before. Therefore, vote spam can give rise to a little decreasing on the performance of ranking function; however, due to the contribution of textual features and community features, our ranking function works in a robust way.

Based on our experiments reported in this section, we conclude that *GBrank-robust* is resilient to vote spam under various settings of the vote spam attack model. And even when the test data does not have overt vote spam, *GBrank-robust* degrades only slightly over our original *GBrank* method (and still performs significantly better than a state-of-the-art baseline). We also observe that *GBrank-robust* is not sensitive to the specific parameters of the vote spam model. Our feature analysis shows that *GBrank-robust* manages to automatically assign more weight to the textual and other more difficult-to-spam interaction features when properly trained.

7. CONCLUSIONS

Social media is transforming the way people find and evaluate information online. Users do not only share information on social media sites, but also contribute their ratings of the content. As such, user feedback has become a crucial mechanism for content quality control, ranking, and filtering. Unfortunately, as the popularity of community and social media sites grows, so do the incentives and incidents of malicious user behavior, such as vote spam.

We have presented a robust, effective method which incorporates social and content information for retrieving information from social media. In particular, we focused on the *robustness* of ranking in the presence of malicious feedback (vote spam), analyzing general models for common vote spam strategies and developing a training method that improves the robustness of ranking by injecting simulated spam into the training data. Our extensive experiments on a particularly important case of social media –Community Question Answering– demonstrated the effectiveness of our approach, which is robust in that the accuracy of ranking degrades gracefully with increased spam, better than unpolluted model. Furthermore, we have shown that our method of training is resilient to vote spam that is significantly different from the training data. In the future we plan to explore further the different spam strategies and corresponding robust ranking methods.

ACKNOWLEDGMENTS We are grateful for the support of the Yahoo! Answers team to allow extensive usage of the Answers search API, and to Ke Zhou from Shanghai Jiaotong Univ. for the implementation of the GBrank algorithm.

8. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of SIGIR*, 2006.

[2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media with an application to community-based question answering. In *Proceedings of WSDM*, 2008.

[3] J. Bian, Y. Liu, E. Agichtein, and H. Zha. Finding the right facts in the crowd: Factoid question answering over social media. In *Proc. of 17th International World Wide Web Conference (WWW2008)*, 2008.

[4] E. Brill, S. Dumais, and M. Banko. An analysis of the askmsr question-answering system. In *Proceedings of EMNLP*, 2002.

[5] J. Friedman. Greedy function approximation: a gradient boosting machine. In *Ann. Statist.*, 2001.

[6] N. Immorlica, K. Jain, M. Mahdian, and K. Talwar. Click fraud resistant methods for learning click-through rates. In *Workshop on Internet and Network Economics (WINE)*, 2005.

[7] B. J. Jansen. Adversarial information retrieval aspects of sponsored search. In *Proc. of the 2nd international workshop on adversarial information retrieval on the web (AIRWeb)*, 2006.

[8] J. Jeon, W. Croft, and J. Lee. Finding similar questions in large question and answer archives. In *Proceedings of CIKM*, 2005.

[9] J. Jeon, W. Croft, J. Lee, and S. Park. A framework to predict the quality of answers with non-textual features. In *Proceedings of SIGIR*, 2006.

[10] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of KDD*, 2002.

[11] T. Joachims, L. Granka, B. Pang, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of SIGIR)*, 2005.

[12] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: A bibliography. In *SIGIR Forum*, 2003.

[13] B. Mehta, T. Hoffmann, and P. Fankhauser. Lies and propaganda: detecting spam users in collaborative filtering. In *Proc. of the 12th International Conference on Intelligent User Interfaces (IUI)*, 2007.

[14] A. Metwally, D. Agrawal, and A. E. Abbadi. Detectives: Detecting coalition hit inflation attacks in advertising networks streams. In *Proc. of the International World Wide Web Conference (WWW)*, 2007.

[15] H. P, G. Koutrika, and H. Garcia-Molina. Fighting spam on social web sites: a survey of approaches and future challenges. In *Internet Computing, IEEE*, 2007.

[16] F. Radlinski. Addressing malicious noise in clickthrough data. In *Proc. of the 3rd international workshop on adversarial information retrieval on the web (AIRWeb)*, 2007.

[17] F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2006.

[18] Q. Su, D. Pavlov, J. Chow, and W. Baker. Internet-scale collection of human-reviewed data. In *Proc. of the 16th international conference on World Wide Web (WWW2007)*, 2007.

[19] E. M. Voorhees. Overview of the TREC 2003 question answering track. In *Text REtrieval Conference*, 2003.

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