Measuring Contributions to Email-Based Discussion Groups

Ian Pye
UC Santa Cruz
Department of Computer Science
ipye@cs.ucsc.edu

Luca de Alfaro
UC Santa Cruz
Department of Computer Science
luca@cs.ucsc.edu

ABSTRACT

Email-based discussion groups are a vast source of non-canonical crowd sourced information. However, due to their open nature (anyone can post), evaluating the quality of answers is challenging. In this work, we develop a framework for analyzing author contributions to email-based discussion groups.

Sentiment analysis is the process of extracting the overall feeling from a body of text. We present a novel technique which applies sentiment analysis to evaluate the quality of answers. We present six novel algorithms and compare their results to a manually calculated baseline, two machine learning algorithms, and two algorithms based on link analysis. We find that by using sentiment analysis, our algorithms out perform both the machine learning and link analysis approaches in most experiments. We also find that a simple text-based approach without sentiment analysis is surprisingly powerful.

1. INTRODUCTION

Email-based discussion groups are frequently the first place a knowledgeable user will look for help when confronted with a technical problem. Precisely because users may interact directly with other users in search of a solution, discussion groups are a trove of information. Their open nature means that not all answers are of similar quality.

We present a framework to analyze the quality of user contributions to email-based discussion groups. We developed six novel algorithms which accurately measure the contribution each user makes in this semi-structured environment with minimal supervision. From this contribution measurement, we calculate a reputation for each author. In our results section, we show that author reputation is a good predictor of future answer quality.

Evaluating the quality of community question answering (CQA) forums and general email exchanges are the focus of much recent research [2, 7, 4]. CQA forums are highly structured: there is one question and multiple answers. The question is a request for information and all answers are attempts to provide this information. Mapping a CQA forum to a graph of user interactions results in a tree, with each posting having exactly one parent to whom they are replying.

In contrast, email exchanges are much more decentralized and free-form. Anyone can email anyone about any subject. Both statements and questions are allowed; a graph of interactions between authors can contain many cycles.

Discussion groups occupy a position between these two extremes. As in a CQA context, each thread begins with a root message, to which answering authors respond. The key distinction between CQA and discussion groups is that answering authors can respond to each other, in addition to the root question. While they enforce an absence of cycles, they allow more general interactions than a straight question and answer format. In our message sample, we observe feature announcements and a user poll, in addition to the prevalent paradigm of problem descriptions accompanied by a request for help.

Previous work in the areas of answer quality rating and expert identification rely on features such as message length and timing, message frequency, as well as explicit rating mechanisms such as the ratio of thumbs up to thumbs down for each message. Additionally, several authors [2, 4] use link analysis algorithms such as HITS [15] in their analysis.

We approach the problem from a textual standpoint, leveraging the content of messages to determine author scores. We do not rely on any external explicit ratings, as these do not usually exist for discussion groups. Our main algorithm does not require any supervision at all.

A discussion group is composed of multiple conversation threads. Each thread is treated as a separate case: an author's reputation over an entire group is simply the sum of that author's reputation in each thread in which the author participates. While doing so they may quote other answers. Therefore, a post in a discussion group is composed of *original* text, written by the post's author, and *quotes*, which were written by an earlier author.

Our framework parses a discussion group, determining the tree of threads, questions, and answers. Each quote is mapped back to the message where this quote originally appeared. Additionally, the sentiment for every original line is computed.

Sentiment analysis, or opinion mining, is the process of determining the opinion of a speaker on a given topic. We break the sentiment of each message into positive, negative, and objective scores. We then use the ratio of positive to negative sentiment to judge how the current author feels about quoted text.

For example, consider the following two messages which are included in our sample set.

Message 1:

```
...Using distributed cache is actually
not that tough from pig...
```

Message 2:

```
Great tip.
Many thanks, I'll try it.
....
-----Original Message-----
...Using distributed cache is actually
not that tough from pig...
```

WWW2010, April 26-30, 2010, Raleigh, North Carolina.
.

Since the original content of Message 2 is highly positive in tone and it quotes Message 1, our algorithm boosts the contribution score for the author of Message 1. The amount of boost is determined by the exact ratio of positive to negative sentiment, as well as the current contribution score of author two. Our system is *content driven*: users gain reputation through the ratings of other users, but users do not rate each other directly. Instead, they rate though their replies, implicitly the content other users create.

To place the effectiveness of our sentiment based approach in context, we implemented several other algorithms, including HITS and PAGERANK. We also use the Weka framework [26] and an artificial neural network to boost the correctness of our author reputations.

Experimental Results

To evaluate the effectiveness of our algorithms, two testers each labeled every responding email in fifty threads from the `pig-user@hadoop.apache.org` discussion group with the percentage that the tester believed the email contributed to the overall resolution of the thread. Every message was labeled by both testers, and the mean of these two labels were used. We chose `pig-user@hadoop.apache.org` because the conversation graphs in many of the threads are relatively complex compared to those we observed in other lists.

The output of each algorithm is a mapping from each author to a contribution score. Our evaluation compares these programmatically generated results against those given by our manual contribution labels. We use three measurements of numeric similarity: vector distance, the correlation coefficient, and relative entropy. Intriguingly, the relative rankings of our algorithms using each of these metrics are largely similar. Our final sentiment-based algorithm performs better than the pure link analysis based approaches. Interestingly, our naive approach of giving each author in a thread equal credit for contributing out-performs all of the more sophisticated approaches. This occurs because, for certain types of threads in our sample, answers are all equally correct; our human labelers respond accordingly while our automatic systems fail to. This shows that for best results one needs to determine the thread type before analyzing the contributions of its authors.

In addition, we look at the precision of each algorithm in spotting the "best" response to each thread. HITS is able to outperform our best-sentiment based algorithm in this setting, but the naive algorithm still provides the best result, depending on how ties are broken.

Our Contributions

To the best of our knowledge, our work is the first to apply sentiment analysis to the challenge of evaluating the relative quality of discussion posts. Additionally, our target area of email based discussion groups is very large and well established[1], yet few studies have looked at ways of separating better posts, and better authors, from poorer ones. As these types of discussion groups function as key secondary information repositories in many areas such as open source software, an effective way of scoring user contributions is needed. Our system allows the rapid processing and evaluation of such discussions, outputting a contribution reputation for each author involved in the discussion group. From this, experts in a subject can be easily identified and users can be warned about posts of dubious quality.

2. RELATED WORK

Our work bridges several areas including expert finding [7, 4], discourse mapping [22, 23], and the prediction of future quality via reputation systems [1, 21]. In this section we provide a brief summary of related work in each of these areas.

Expert Finding and CQA Forums

Expert finding is the problem of identifying a subset of the total population who are highly knowledgeable on a given subject. This is useful in organization mapping and performance evaluation. Campbell et al. [4] evaluate the effectiveness of finding experts based solely on a corpus of email text as compared to using both text and a graph of communication patterns. They find that by using the link analysis-based HITS (Hyperlink-Induced Topic Search) algorithm [15], they achieve a more accurate result than counting topic frequency in the email corpus and assigning reputation for a topic based on how frequency the topic is mention.

Bian et al. [2] apply machine learning to more effectively rate the quality of questions and answers in CQA forums. The authors break reputation down into "question" and "answer" quality values. Using a set of 250 question/answer sets from Yahoo! Answers labeled as good/bad, the authors apply logistic regression to get an accurate prediction of future question and answer quality with limited training. They achieve good precision in selecting the best answers in a set. However, their approach requires explicit user ratings of all answers.

Zhang et al. [27] also measure expertise in a user forum, looking at the effectiveness of HITS vs the PageRank [19] algorithm. They only use the graph structure for their algorithms, and do not look at the posts' text. Dom et al.[7], also apply link analysis algorithms to evaluating email expertise.

We find experts tangentially; we rate user's contributions to a discussion dedicated to a particular product, but we do not attempt to explicitly validate the top scorers as being experts in the discussion group's topic. We are exploring this connection in future work.

Discourse Mapping

Most prior research on email-based discussion groups has focused on discourse mapping. Several groups have explored visualizing and re-constructing the flow of conversation in discussion groups and email. In [22], the authors created a tool which parses Usenet groups to find who is citing and replying to whom, as well as the main themes of a discussion thread. Carenini et al. [5] look at summarizing discussions by studying the patterns of quotation in email exchanges. They also attempt to rank the importance of difference sentences in a conversation. The GroupLens project [16] is another attempt to filter Usenet posts. Using collaborative filtering, the project tries to predict conversations of interest to a user. Karagiannis et al. [14] reconstructs the flow of email in the more free-form environment of a large corporation rather than a discussion group such as Usenet. On a broader level, Sinatra et al. [23] look at ways of visualizing how the major and minor ideas of a written work are related.

Sentiment Analysis

Sentiment analysis is the process of extracting an option about a topic from a body of text. This opinion is codified as three numbers expressing the positive, negative and objective connotations of a particular word. We combine these "word scores" to get positive and negative scores for an entire discussion post. The sentiment dictionary provided by the SentiWordNet project [8] provides the basis of our analysis.

A recent book by Bo Pang and Lillian Lee [20] provides a good rundown of current research in this area. Of particular interest, Godbole et al. [10] have looked at the sentiment analysis of newspapers and blogs. Also, Somasundaran et al. [25] take up the challenge of figuring out what the target of an opinion is. While we currently resolve this issue in our work by arbitrarily spreading opinion across all quoted lines, a more fine grained approach has the

[1]Usenet has been going strong since 1979.

potential to greatly increase the accuracy of our algorithms. None of these authors use sentiment analysis as the basis for a reputation system.

Ghose et al. [9] come close to this. They look at the product review system of Amazon.com They show that those products which benefit the most from reviews have reviews which are strongly positive. Over time, this results in these products increasing in price.

Reputation Systems

We use reputation as a way to weight the positive and negative statements made by authors. Without reputation, a statement which is overwhelmingly positive or negative lacks context. Does the author have a history of making such statements? Do other authors agree when the author makes such statements? With reputation, we turn judging quality into a repeated game: we know what has happened in the past so when a new event occurs we are able to weight the event accordingly. As with any reputation system, the validity of reputation is defined by how much a user's reputation is a good predictor of the user's future work

Specifically in our work, we use a form of reputation know as *content-driven* reputation. As laid out in several recent works [13, 3], there are two major types of reputation systems. The first, far more prevalent type are *user-driven*: they are based on users directly rating each other [21, 6]. In contrast, a content-driven reputation system is built up from the rating of *work* which users contribute, rather than rating other users directly. Our system is content-driven because we update an author's reputation based on how other authors rate this author's posts.

One recent example of a content driven reputation system is that used by the WikiTrust project. In [1], the authors show that by looking at the history of an author's edits on the Wikipedia, accurate predictions can be made about the life expectancy of the author's future edits. In other words, those who have done good will most likely continue to do good.

Lastly, in [11], Guha et al. look at how trust spreads in a social network and show that even with each individual having a small number of interactions, the composite graph of interactions allows for accurate prediction of trust between any two persons. Since relatively few authors directly quote one another in our evaluation set, this property of trust inference is key to us.

3. PROBLEM DESCRIPTION

Our goal in this work is to automatically measure the contribution each author makes to an email-based discussion group. We define broadly what types of discussions our tool works on: any setting with an initial posting followed by responses, as long as responses can quote other responses. Therefore, we look at threads which include single question and multiple answer style postings, as well as threads which begin with a statement, such as the announcements of new features.

Following [2], we look at both quality and reputation. For CQR forums in particular, Bain et al. define the following measurements for individual questions:

- **Question Quality:** a score between 0 and 1 indicating a questions effectiveness at attracting high-quality answers.
- **Answer Quality:** a score between 0 and 1 indicating the responsiveness, accuracy, and comprehensiveness of the answer to a question.

And from the question metrics, Bian et al. build up these author reputation metrics:

- **Answer-reputation:** a score between 0 and 1, indicating the expected quality of the answers posted by a user.
- **Question-reputation:** a score between 0 and 1, indicating the expected quality of the questions posted by a user.

We keep the devision between question quality and answer quality, but choose to focus our effort on accurately calculating answer quality. In this work, we fix question quality as being the number of answers a question attracts. The intuition is that authors will only respond to questions which they believe are worth their time answering. By fixing question quality, we are able to focus on accurately determining answer quality. In the future, we hope to explore more reactive ways of gauging question quality. Therefore, we define:

- **Question-quality:** a score between 0 and ∞, indicating the *number* of answers the question attracted.
- **Relative answer-quality:** a score between 0 and 1, indicating the *relative* quality of the answer, compared to other answers in the thread. For each thread, the sum of all relative answer-quality scores is 1.

We take these two and combine them, generating a single value which reflects the more ambiguous nature of general-purpose email discussions, where what is a question and what is an answer is not always well defined.

- **Contribution-reputation:** a score between 0 and ∞, indicating the expected quality of a user's future contributions to a discussion.

In practice, we arrive at an author's contribution reputation score by summing up the weighted question quality and unweighted answer quality scores for each message that the author writes. In this manner, prolific authors only gain reputation quickly if they produce high quality work.

- **Predicting Author Contribution:**

  Given a general discussion group archive, composed of multiple discussion threads, determine the question and relative answer quality for all messages in all threads in the archive. From these message quality measurements, produce a contribution reputation for each contributing author.

Formally, let α_p be the author contribution of the pth author over all threads in a group. Q_p is the set of questions asked by α, while A_p is the set of answers. m_i^q is the question quality of message i, while m_i^a is the answer quality of message i. A^t is the set of answers for the tth thread. κ is the weight given to question quality. Then,

$$\alpha_p = \kappa \sum_{m \in Q_p} m_i^q + \sum_{m \in A_p} m_i^a \tag{1}$$

Where $m_t^q = |A^t|$ and $\sum_{m \in A^t} m_i^a = 1.0$.

This equation above holds for all of the algorithms we present in Section 5. The difference between them lies in how answer quality is calculated.

4. EXPERIMENTAL SETUP

In order to conduct an empirical study of the effectiveness of our algorithms, we choose to focus on one particular discussion group, `pig-user@hadoop.apache.org`, which is archived online by the website Mail Archive [18]. Mail Archive is an online collection of 67370804[2] postings in a variety of technical and social discussion groups. We chose this repository because it provides a very large collection of messages, all of which are presented in a standard HTML format.

To establish quality metrics, we annotated all of the responding messages from 50 threads from the `pig-user@hadoop.apache.org` list with an answer resolution percentage[3]. To make this score less subjective, we had two people label the same corpus. Overall, the two labelers labeled 193 individual messages. Their instructions were to assign a percentage to each responding thread in the corpus, representing their opinion of how much each message contributes to the thread. We then take the mean of the two percentages for each message as our canonical MANUAL value. This manually annotated score provides a baseline against which to measure our objective algorithms.

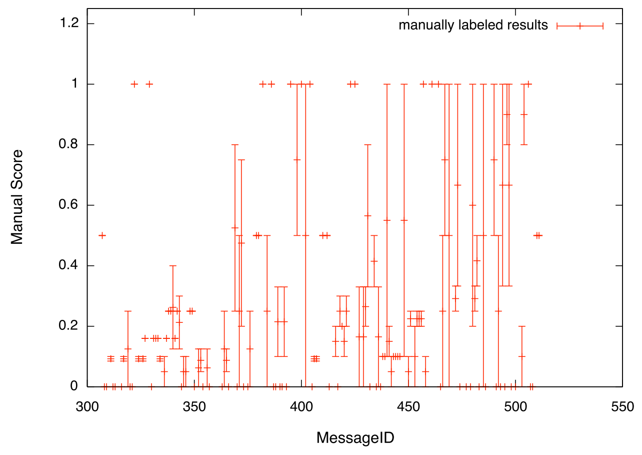

Figure 1: In the graph above, the scores of the two labelers for each message are represented by the y error bars. The "plus" in the middle is the mean, used for scoring in the MANUAL algorithm and as a baseline to judge other algorithms against.

Figure 1 shows the distance from this mean for each labeled message; the two message labelers achieved a correlation of 0.692773. 79 messages were labeled exactly the same while 18 had labelings which were off by 0.5.

For initial questions, we provide a weighted score based on how many responses there were on the thread. We believe sparking discussion is also a contribution to discussion. Note that this initial score is easily computable, and the same values are used in all of our scoring algorithms.

4.1 Evaluation Criteria

We use four metrics for evaluation. Three of these compare the sets of numerical values (author reputations) returned by each algorithm, while the last, precision, is a measurement of binary classification soundness.

- **Euclidian Vector Distance:** We started with this metric, which measures the distance between two vectors. We generate the vectors by mapping each author in the result set to a position in a vector, while the author's contribution reputation provides the value. By comparing each algorithm's results and the manually calculated results in turn, the Euclidian distance formula provides a relative quality ranking. The assumption here being that an algorithm's quality increases as its vector's distance to the baseline vector decreases.

- **Kullback-Leibler Divergence:** Also known as relative entropy [17], this is a common technique to measure the distance between two probability distributions. Kullback Leibler Divergence is calculated as $D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)}$. Its benefit over Euclidian distance is that, since it measures the distance between probability distributions, it does't grow as our sample size increases. Additionally, it is less responsive to a few highly divergent values. To calculate the relative entropy, we normalize our result vectors, generating a probability distribution for each algorithm. We map our manual labelings to $P(i)$, while each algorithm supplies $Q(i)$.

- **Correlation Coefficient:**

  Borrowing a common calculation from statistics, calculating the correlation between two random variables indicates the strength and direction of their relationship. Correlation is frequently used in Machine Learning [12] to assess the quality of an algorithm's results. We use the correlation coefficient as the primary comparison between the machine learning approaches we use and our other algorithms.

- **Precision:**

  Precision is the ratio of true-positives to false-positives. We treat the response in each thread with the highest manual label as being the "best" answer. Ties are broken in two ways: we first favored the last response, since earlier responses still needed the followup answer to completely answer the question. Our results also look at how favoring the first response in a tie effects the quality rankings of our algorithms. In each case, true positives are defined by what the manual algorithm returns for each thread.

5. IMPLEMENTATION

We have developed a framework for automatically parsing and evaluating message threads implemented in the Objective Caml language. Our framework works in two phases. The first phase takes as input a url representing a discussion archive, for example `http://www.mail-archive.com/pig-user@hadoop.apache.org`, as well as the number of threads to process. This phase outputs a tree of discussion posts where a simple syntax of each message and the overall discussion has been parsed.

Table 1: The grammar of a discussion group

Group	⇒	Thread list
Thread	⇒	Message tree
Message	⇒	Line list
Line	⇒	Original | Quote
Original	⇒	String
Quote	⇒	Message ref

Our message syntax breaks down messages on a per-line bases into original content and quotes. Original content are lines which appear for the first time in the thread in the current message. Quotes are lines which appeared in other, previous messages. Commonly,

[2]as of July 19, 2009

[3]This dataset is available upon request.

these are prefixed with a "¿" or "—" character. The compete grammar is defined in Table 1. Our system matches up each quote it finds with the message from which the quote was taken. Quotes of quotes are matched up with the original message, not the intermediary message which was subsequently quoted again.

Once we have the basic syntax parsed, we move on to the semantics of each message. For this, we use a dictionary of word sentiment provided by SentiWordNet[8]. This assigns a positive, objective and negative value to each word in the dictionary.

To use the dictionary, first we map every word back to its stemmed form using the libstemmer C library. For example, both running and ran stem to run. Ignoring objective values, we then sum up the positive and negative values for every stemmed word of every original line in each message. If a word is not found, we assign it a score of 0 positive, 0 negative. This gives us a positive and negative score for each line. In order to enhance the speed of our tool, we created a SentiServer web API that takes a stemmed word and returns its positive, objective, and negative scores.

At this point, our algorithms diverge. Every algorithm described below takes as input the parsed discussion group. Each algorithm then outputs an array of author reputations. From these we calculate the algorithm's quality as described in Section 4.1.

The remainder of this section details how each of the eleven algorithms we analyze do this. Using the sentiment scores and message graph, we started out doing the simplest thing we could think of. Each subsequent attempt tries to rectify some of the failings we observed in the previous algorithms. Lastly, we describe our attempts to boost the performance of our algorithms using supervised machine learning.

5.1 Algorithms

All of these algorithms are ways of computing relative answer quality (m_q^i) for all answers in a thread. A user's reputation is computed from answer quality as described in Section 3. Note that many threads terminate with a "thank you" message from the initial author. To prevent this from skewing our results, we do not give a score to posts by the initial author in any of our algorithms.

- MANUAL

  For our manual analysis we simply use the mean of hand-calculated scores for each message. This algorithm is the baseline against which we evaluate all of the others.

Naive Approaches

- BASIC

  Here, all answers are given an equal percentage of the credit. In this way, BASIC simply measures the frequency of an author's contributions. While there is no attempt to measure the quality of an author's contributions, this algorithm performs better than any other approach. We believe that this non-intuitive result is due to the question type mixture observed in our sample set. We discuss this result more in Section 6.4.

- LAST

  The LAST algorithm awards answers a score based on how few messages are posted after them. The idea is that terminal messages must be complete. Conversely, if there are a large number of follow ups, there are still things to clear up.

Sentiment Analysis Algorithms

- BASIC-SENTI

  BASIC-SENTI is our first attempt to evaluate and use the semantic value of posts, rather than simply counting posts. First, the ratio of positive to negative sentiment in each message's original lines is calculated. This is done by summing up all of the positive values for each word and dividing by the sum of the negative values.

  We map discussion threads as trees. The original question provides the root. As with email, each message is in response to only one other message. This is the message's parent. The children of each message are those messages who respond to it.

  The ratio of positive sentiment to negative is used to boost the score of the message's parent, where the higher the proportion of positive to negative sentiment is, the more reputation this message gains. Messages cannot lose quality; a strong negative ratio simply results in zero reputation gain.

- BASIC-PROP

  The problem with BASIC-SENTI is that no attempt is made to turn the raw message quality scores into proportional gains over each thread. Normalizing the ratio results in the BASIC-PROP (basic-proportional) algorithm. Here, each answer gets a percentage of the total raw quality available corresponding to the author's percentage ownership of all of the awarded contribution points. This normalization is used in all of the later algorithms as well.

- SENTI-QUOTES

  The problem with BASIC-PROP is that it ignores the information found in quotes. In SENTI-QUOTES, the benefit of the sentiment of the original lines are spread proportionally over all messages quoted, rather than being given entirely to the direct parent of a post. Additionally, the benefit is weighted by the reputation of the current email's author. SENTI-QUOTES thus provides a content driven reputation system, where content is judged by other users and a positive judging results in increased reputation.

  For example, if message one is, "`Set $foo to 5 for best results`", while message two says, "`> Set $foo to 5 -- this is true, $foo should be 5`", our tool would extract a sentiment score of (0,0,1) (positive, negative, objective) for message one and (1,0,1) for message two. Since message two is quoting message one and has a strong positive component, message one would gain reputation in proportion to the reputation of message two's author.

- SENTI-NC

  SENTI-NC (non-content-senti in figures) provides a counterpoint to SENTI-QUOTES. Rather than splitting the benefit up among the quoted lines, it gives all of the quality boost to the current message. We expected this algorithm to perform poorly, but in our experiments it closely tracks the performance of SENTI-QUOTES. We hypothesize that this is because authors who are certain of their response will write message which are strongly positive in tone, and in our sample these positive messages are quality messages. Another reason is that other authors are more likely to agree with strongly positive messages.

Link Analysis Algorithms

- HITS

Other work [2, 4] use the HITS family of algorithm as a way of estimating message quality. We implemented HITS using the notion that parents have out-links to all of their children. HITS, associates two non-negative scores with each node in a graph: the authority score and the hub score [15]. The hub score measures the value of a node as an in-link, while the authority score measures the value of the node to its author. We take the final authority score as the quality for each message, normalized over the thread.

- PAGERANK

  Additionally, we implemented the PAGERANK[19] algorithm. In-links and out-links are the same as in HITS. This algorithm calculates the probability that a random walk will result in a user getting to a page. For our purposes, we use this value as the measure of a message's quality. Normalizing the resulting values over a thread results in fairly good performance.

Machine Learning Algorithms

- ANN

  We next looked at training an artificial neural network on the previous algorithms' scores, attempting to closer approximate the manual scores. Our approach uses a ANN with one input neuron for each author in the set considered. There is also one output for each author. Our input is the author reputations provided by the previous algorithms. Our output is a new author reputation set. We train the resulting ANN on the first half of the threads in our dataset. We validate it on the second.

- M5P and LINEAR REGRESSION

  We extracted the information described in Table 2 for use in the popular machine learning tool Weka [26]. We fed the features into two common techniques used for predicting numeric values: M5P and LINEAR REGRESSION. Using the correlation values provided by Weka, we show that neither of these approaches performs especially well.

6. EXPERIMENTAL RESULTS

In this section we first present the relative effectiveness of our textual and link analysis algorithms, measured in three different ways. We then introduce the results from our machine learning approaches, followed by a section evaluating the precision of all of our algorithms. Lastly we discuss the impact of conversation type on algorithm effectiveness.

Gratifyingly, the relative quality ranking of all of the algorithms, shown in Table 3, mostly agree at the 50 thread mark. The ordering for precision is strikingly different than that of the numeric quality measurements above it. We discuss the reasons for this in Section 6.3.

6.1 Numeric Comparisons

As described in Section 5, we present three main measurements of numeric quality: Vector Distance, Relative Entropy and Correlation.

Overall, SENTI-NC and SENTI-QUOTES algorithms very closely track one another, PAGERANK is the clear winner over HITS, and

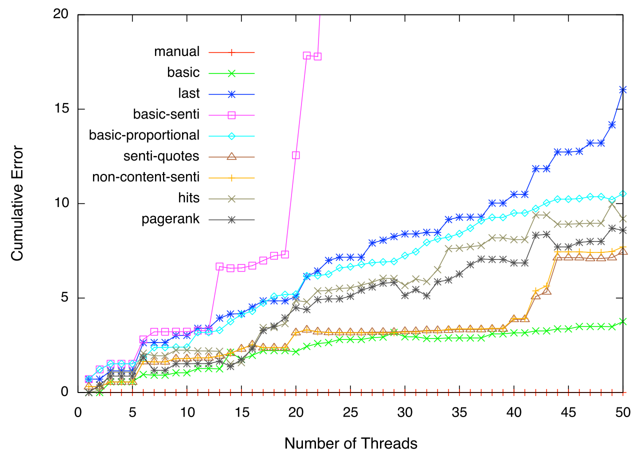

Figure 2: The euclidian vector distance for each algorithm from MANUAL as the number of examined threads goes from 1 to 50.

BASIC is able to pull off a surprise win. This is because the dominant type of conversation changes during the latter threads in our sample towards a type which favors BASIC.

Our first experiment, show in Figure 2, looks at how vector distance from MANUAL increases as the number of threads considered grows from one to fifty. Observe that since reputation is additive across threads, vector distance tends to grow as the number of threads considered increases. What defines the quality of an algorithm is the rate of increase.

In Table 4, we present the standard deviation of all of the algorithms from the MANUAL baseline at fifty threads. This is calculated by first taking the difference of each author's reputation from the MANUAL reputation score and then finding the standard deviation of the resulting series. Interestingly, the low standard deviation posted by BASIC shows that in most cases the MANUAL labelings spread quality equally across a thread's answers.

Table 4: The table below shows each algorithm's relative entropy in comparison with MANUAL. It also shows the standard deviation of the difference of each author's contribution reputation from that calculated by MANUAL. These calculations are all performed after evaluating fifty threads

Algorithm	Relative Entropy	σ
MANUAL	0.0	0.0
BASIC	0.077	0.583
LAST	0.221	1.868
BASIC-SENTI	0.349	17.3
BASIC-PROP	0.343	1.64
SENTI-QUOTES	0.127	1.07
SENTI-NC	0.139	1.12
HITS	0.555	1.303
PAGERANK	0.305	1.21

Figure 3 shows the algorithm's relative entropy from MANUAL as the number of threads considered goes from 1 to 50. Here, the link analysis algorithms perform more poorly than in other metrics. This is due to the fact that the entropy metric is more forgiving of

Table 2: Features Spaces: $X(A)$

Answer Quality Feature Space $X(A)$	
A: Subject Length	Number of words in the message subject
A: Body Length	Number of lines in the message body
A: Positive Tone	Sum of the positive values for all original words in the body
A: Negative Tone	Sum of the negative values for all original words in the body
A: Objective Tone	Sum of the objective values for all original words in the body
A: Number of Children	Number of direct responses to the message
A: Message Id	Id of the message

Table 3: The relative quality orderings of each of our algorithms using different evaluation criteria.

Metric	Best $\Rightarrow$ Worst
Vector Distance	MANUAL, BASIC, SENTI-QUOTES, SENTI-NC, PAGERANK, HITS, BASIC-PROP, LAST, BASIC-SENTI
Relative Entropy	MANUAL, BASIC, SENTI-QUOTES, SENTI-NC, LAST, PAGERANK, BASIC-PROP, BASIC-SENTI, HITS
Correlation Coefficient	MANUAL, BASIC, SENTI-QUOTES, SENTI-NC, LAST, PAGERANK, HITS, BASIC-SENTI, BASIC-PROP
Precision (last)	MANUAL, HITS, PAGERANK, BASIC-SENTI, SENTI-QUOTES, SENTI-NC, BASIC, LAST, BASIC-PROP
Precision (first)	MANUAL, BASIC, HITS, PAGERANK, SENTI-QUOTES, SENTI-NC, LAST, BASIC-SENTI, BASIC-PROP

sets with a few large outliers, due to the *log* factor in the evaluation. As Table 4 shows, the standard deviation in the textual algorithms is higher than in the link analysis ones.

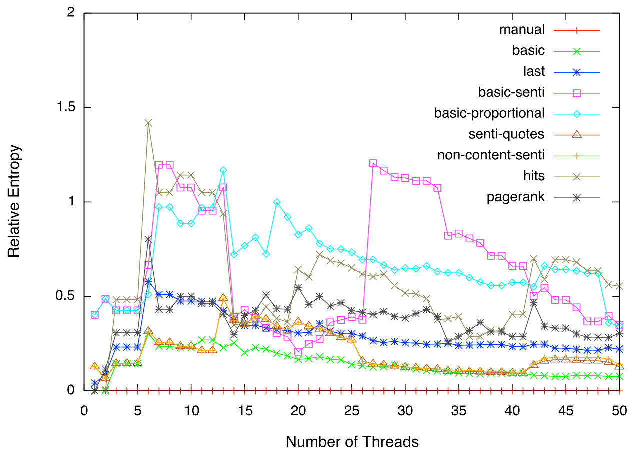

Figure 3: Each algorithm's relative entropy, from MANUAL, as the number of examined threads goes from 1 to 50.

The correlation of the other algorithms' results to MANUAL's results is shown in Figure 4. Both the link analysis and textual algorithms show high correlation. The relative rankings are similar to those seen in Figure 3. As we will see in the next section, all of the correlations presented above are better than those achieved by the Weka based machine learning algorithms. This shows that reputation is not distributed amongst authors in a form which is tractable to the linear approximation our Weka approaches use.

Reputation is only useful because it gives users a reason to trust. In our last numeric experiment, we explore how good a predictor of future answer quality a user's reputation is. That is, how much should we trust that a high reputation user will continue to contribute highly to a discussion group? We calculate this by generating a reputation for each user on the first 25 threads in our sample. We then compare this to the MANUAL reputation generated by each user in the last 25 threads. Only users who posted at least one answer in both sets are considered. In Table 5, we see that while all of the algorithms return lower scores than before, an author's reputation is still an effective way of estimating future contribution

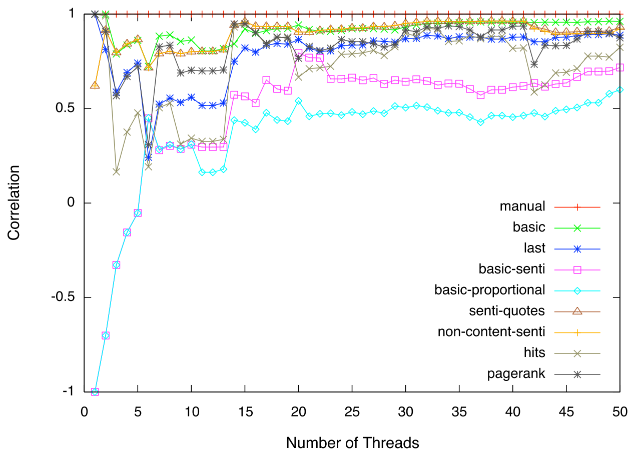

Figure 4: Each algorithm's correlation coefficient as the number of examined threads goes from 1 to 50.

quality.

6.2 Machine Learning

Because of the nature of having to train and validate on different sets, we exclude the machine learning algorithms from the first section. In this section, we present the effectiveness of these algorithms and place this in context with the other approaches.

The results from the Weka algorithms show disappointing correlation coefficients. Since both of these provide linear approximations to the data, this was expected. ANN is able to do much better, being competitive with all of the link analysis and textual algorithms across metrics. We trained the LINEAR REGRESSION, M5P and ANN algorithms on the first 25 threads, consisting of 87 messages, and used the last 25 threads, with 106 messages, as validation. As Table 6 shows, neither LINEAR REGRESSION nor M5P are able to predict the quality labeling with any degree of accuracy. ANN shows a correlation in line or better with the other algorithms, which were ran over the same validation set. Note that both Weka algorithms found the same predictive formula of:

Table 5: Reputation as a predictor of future quality. To be valid as a reputation system, a user's reputation must be an accurate indicator of the quality of a user's future work. In the table below, we present how the reputation calculated from the first 25 threads fares against the manual reputation gained by each user over the last 25 threads. We measure this using VD (vector distance), CC (correlation coefficient) and RE (relative entropy).

Algorithm	VD	CC	RE
MANUAL	0.0	1.0	0.0
BASIC	4.5	.69	.36
LAST	7.17	.65	.46
BASIC-SENTI	25.3	.38	.47
BASIC-PROP	6.59	.34	.65
SENTI-QUOTES	5.54	.66	1.3
SENTI-NC	5.53	.66	1.3
HITS	5.89	.54	1.2
PAGERANK	5.64	.62	.85

$$\begin{aligned}\text{Manual Score} = {} & -0.1593 * \text{Number of Children} \\ & +0.8166 * \text{Objective Tone} \\ & +0.1745\end{aligned}$$

This results in the same correlation values for both algorithms.

Table 6: Machine Learning Results

Algorithm	Correlation Coefficient
MANUAL	1.0
BASIC	0.9423
LAST	0.7920
BASIC-SENTI	0.4825
BASIC-PROP	0.4666
SENTI-QUOTES	0.8375
SENTI-NC	0.8139
HITS	0.6975
PAGERANK	0.7276
ANN	0.8663
M5P	0.2469
LINEAR REGRESSION	0.2469

Our next experiment observes how the results of the ANN algorithm change as its inputs are generated by each of the other algorithms. We see in Figure 5 that while there is some variation, the input algorithm is largely irrelevant to the output.

Using the best input of PAGERANK, we see how ANN compares to the other algorithms using all of our evaluation criteria. For this comparison, all of the other algorithms are evaluated using the same validation set as ANN.

Looking at vector distance in Figure 6, we see that ANN is able to beat all of the other algorithms except for BASIC. The number of threads considered here goes from 1 to 50. in every case, we train ANN on the first half of the threads and validate on the second. The results show are those produced by running each algorithm on the validation set.

Interestingly, using the relative entropy metric, show in Figure 7, ANN does much poorer in comparison to the other algorithms than when we use vector distance.

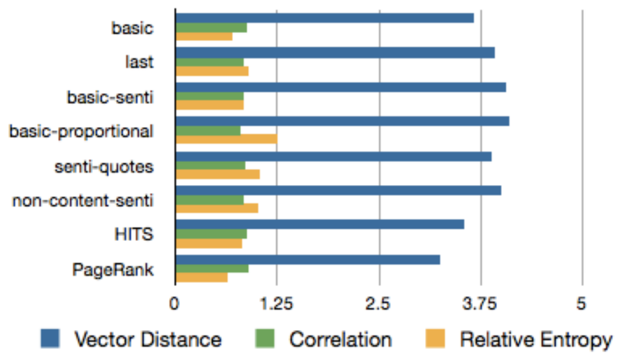

Figure 5: The quality of our ANN solution, using different algorithm's outputs as the input to ANN. While there is not much change, we see that ANN does best using PAGERANK as its input. The quality is evaluated on the last 25 threads in the test set, after training on the first 25.

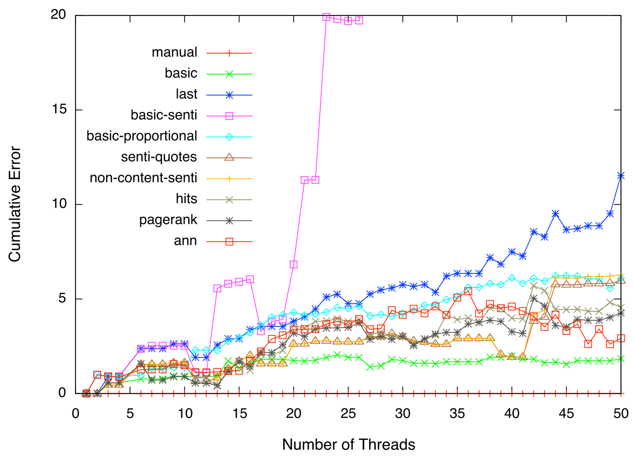

Figure 6: The Euclidian vector distance for each algorithm from MANUAL as the number of examined threads goes from 1 to 50.

6.3 Precision

Our last experimental section moves from regression to classification. We wished to evaluate how closely our algorithms can predict the *most* useful response in a thread. For this, we assume that the message with the highest manual score is the most useful. Ties are broken in preference to the last message, our reasoning being that the thread was not complete until the last high score was added. Figure 9 shows the results, which are quite surprising to us. The non-textual algorithms PAGERANK, and to a lessor extent HITS, are the clear winners here.

This is due to these algorithms nature of aggressively favoring the last message in a thread, while the textual algorithms tend to spread credit out more evenly over all contributors. Since we break ties in favor of the last message, this policy benefits the link analysis theoretic algorithms at the expense of the textual. Note that because of the way the best message is calculated, BASIC and LAST return exactly the same results. In our graph, BASIC is hidden by LAST.

We then re-ran this experiment, this time breaking ties in favor of

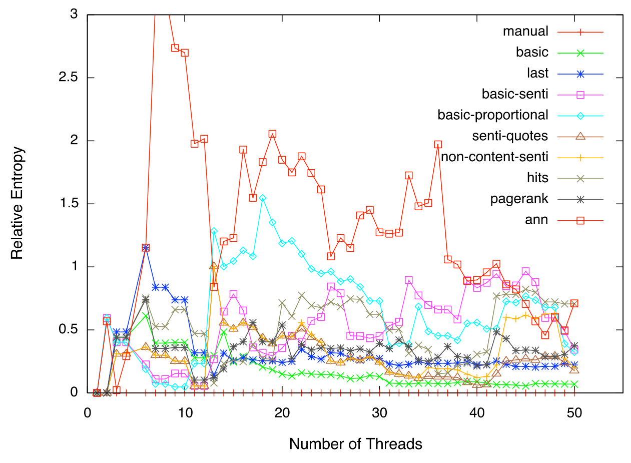

Figure 7: Each algorithm's relative entropy, from MANUAL, as the number of examined threads goes from 1 to 50.

the first message. Figure 8 gives these results. As expected, BASIC-SENTI and LAST do much poorer here, while the the link analysis theoretic algorithms still dominate the sentiment-based ones.

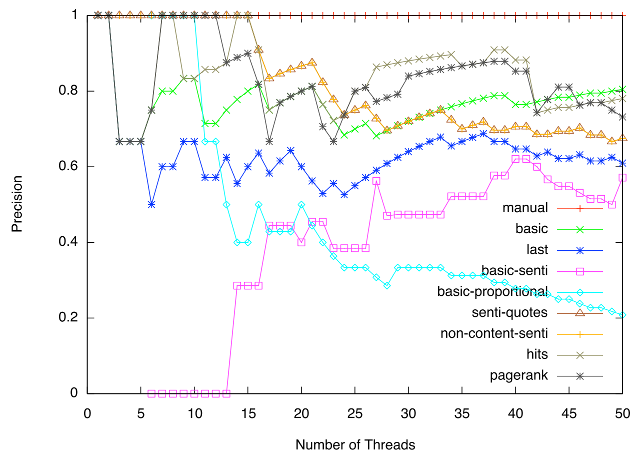

Figure 8: Each algorithm's precision as the number of examined threads goes from 1 to 50. Ties are broken in favor of the first message.

6.4 Conversation Types

After observing how our algorithms perform over different threads, we came to realize that our sample contains three separate thread types. The first, a straight Technical Question Answer (TQA), describes threads where one initial author asks a technical question. All of the responding threads are attempts to answer the question, guided by possible followup questions asked by the original author. In this type of thread, later responses are more likely to be the best, as these are the ones which finished the conversation, presumably because the questions was completely answered to the satisfaction of the original poster. Algorithms like PAGERANK, LAST, and SENTI-NC do better in this type of thread because of their bias towards later posts. Conceptually, the magnitude of the valid answer space for TQA is 1.

Other threads followed a pattern where the initial post asks a question which does not have a definite "true" answer. One example of such a question is:

> If you have done some interesting work using Pig, we would like to know! Could you please, send a brief description of the kind of work you are doing with Pig and what you have learned from working woth [*sic*] Pig: things that worked well, things that you would like to be improved.

For NonSpecific (NS) questions like this, the manual grader scored all responses equally, as they are all equally valid. Here, the magnitude of the valid answer space is ∞. This throws off algorithms which do well in the previous type of thread but rewards BASIC. The preponderance of this type of question in the final threads accounts for the comeback BASIC makes in all of our evaluations.

Lastly, there are questions which are a blend of TQA and NS. For these, the magnitude of the valid answer space is finite but greater than 1. SlashDot[24] article threads often take this form, where all of the highly rated responses tend to be towards top of the responses stack. From this, we deduce that everything of value to be said is said quickly.

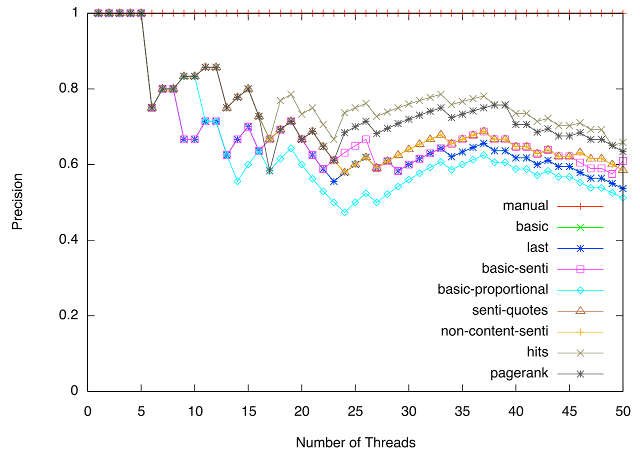

Figure 9: Each algorithm's precision as the number of examined threads goes from 1 to 50. Ties are broken in favor of the last message.

7. CONCLUSIONS AND FUTURE WORK

Email-based newsgroups are frequently the first place a technical user goes to find a solution when they get stuck trying to solve a technical problem. Our test set for example is full of problem descriptions and requests for help regarding the Hadoop project's Pig platform. Other users reply to these problem descriptions and eventually, hopefully, a working solution is arrived at. Archived online, these discussions help many more users than those originally participating. In most cases, all archived posts are presented without external annotation. Our system allows for high quality posts, as well as high quality authors, to be found quickly. Additionally, our system can track the benefit each author provides to a discussion.

In a traditional email-based group, users reply to other posts, but not always the initial post. They frequently quote one another, ask for clarification, and express an opinion on other user's proposed solutions. We assume no outside quality metrics: there is no handy way for later users to rate posts. However, the conversation is structured in a way that general emails are not: all posts in a thread are by definition on the same topic.

Our work looks at ways of evaluating these discussions, outputting a reputation for each participating user. For this task, we developed several algorithms leveraging sentiment analysis and the internal quote structure of a post. We compare these to the link analysis algorithms HITS and PAGERANK which do not use any information about message's text, in addition to naive algorithms BASIC and LAST. We perform this comparison using a variety of metrics, including relative entropy or KullbackLeibler divergence, vector distance, precision, and the correlation between each algorithm's results and a manually calculated baseline. We find that a user's reputation is a good predictor of the quality of a user's future posts. Of the algorithms we explore, SENTI-QUOTES performs better on numeric comparisons than HITS and PAGERANK, although these have a higher precision. We are able to boost the quality our automatic labelings using the ANN algorithm. Ironically, the naive algorithm BASIC performs the best across our entire sample set.

In the future, we plan to use an expanded set of labeled emails to validate against. Additionally, we plan to explore the benefits and practicalities of a multi-phase evaluation: first the thread is categorized as being of the form TQA, NS or blended. After this, the best algorithm for each type is applied to posts in the thread. A refinement may be also using each algorithm in a panel of experts format, adjusting the weights of the experts as the type of thread changes. Lastly, we are looking at moving from thread contribution to expert finding. This would involve calculating total contributions to a subject, as this subject is discussed across multiple threads.

8. REFERENCES

[1] B. Adler and L. de Alfaro. A content-driven reputation system for the wikipedia. *Proc. of the 16th Intl. World Wide Web Conference*, Jan 2007.

[2] 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. *Proceedings of the 18th international conference on the World Wide Web*, Jan 2009.

[3] A. Bouajjani, J. Esparza, and S. Schwoon. SDSIrep: A reputation system based on SDSI. *LECTURE NOTES IN COMPUTER SCIENCE*, Jan 2008.

[4] C. Campbell, P. Maglio, A. Cozzi, and B. Dom. Expertise identification using email communications. *Proceedings of the twelfth international conference on Information and Knowledge Management*, Jan 2003.

[5] G. Carenini, R. Ng, and X. Zhou. Summarizing email conversations with clue words. *Proceedings of the 16th international conference on World Wide Web*, Jan 2007.

[6] C. Dellarocas. The digitization of word-of-mouth: Promises and challenges of online reputation systems. *Management Science*, October 2003.

[7] B. Dom, I. Eiron, A. Cozzi, and Y. Zhang. Graph-based ranking algorithms for e-mail expertise analysis. *Proceedings of the 8th ACM SIGMOD workshop on Research Issues in Data Mining and Knowledge Discovery*, Jan 2003.

[8] A. Esuli and F. Sebastiani. Sentiwordnet: A high-coverage lexical resource for opinion mining. *tcc.itc.it*, Jan 2007.

[9] A. Ghose, P. Ipeirotis, and A. Sundararajan. Opinion mining using econometrics: A case study on reputation systems. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 416, 2007.

[10] N. Godbole, M. Srinivasaiah, and S. Skiena. Large-scale sentiment analysis for news and blogs. *Proceedings of the International Conference on Weblogs and Social Media*, Jan 2007.

[11] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. *Proceedings of the 13th international World Wide Web Conference*, Jan 2004.

[12] M. Hall and L. Smith. Practical feature subset selection for machine learning. *Computer Science*, Jan 1998.

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

[14] T. Karagiannis and M. Vojnovic. Email information flow in large-scale enterprises. *research.microsoft.com*, Jan 2008.

[15] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, Jan 1999.

[16] J. Konstan, B. Miller, D. Maltz, and J. Herlocker. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, Jan 1997.

[17] S. Kullback and R. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, Jan 1951.

[18] The mail archive. `http://www.mail-archive.com`.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *lipn.fr*, Jan 1998.

[20] B. Pang and L. Lee. Opinion mining and sentiment analysis. *books.google.com*, Jan 2008.

[21] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kiwabara. Reputation systems. *Comm. ACM*, 43(12):45–48, 2000.

[22] W. Sack. Conversation map: An interface for very large-scale conversations. *Journal of Management Information Systems*, Jan 2000.

[23] R. Sinatra, J. Stahl-Gemake, and N. Morgan. Using semantic mapping after reading to organize and write original discourse. *Journal of Reading*, Jan 1986.

[24] Slashdot.org. `http://www.slashdot.org`.

[25] S. Somasundaran, J. Ruppenhofer, and J. Wiebe. Discourse level opinion relations: An annotation study. *SIGdial Workshop on Discourse and Dialogue*, Jan 2008.

[26] I. Witten and E. Frank. Data mining: practical machine learning tools and techniques with java implementations. *ACM SIGMOD Record*, Jan 2002.

[27] J. Zhang, M. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. *Proceedings of the 16th international conference on the World Wide Web*, Jan 2007.