Trust and Helpfulness in Amazon Reviews: Final Report

Dylan Shinzaki, Kate Stuckman, Robert Yates
Group 60

December 10, 2013

Abstract

On Amazon, many purchase reviews are dishonest spam entries written to skew product ratings [1]. Though users have the opportunity to rate reviews as helpful or unhelpful, sociological factors and prior ratings influence users to rate these reviews for reasons other than the truth of their content [2, 3]. Many studies have evaluated the content of these user reviews to detect spam entries by mining and classifying the text entry. However, [4] proposes a graph based algorithm to determine the honesty of reviews and trustworthiness of the reviewer for general product review data. In this project, we propose to apply this algorithm to Amazon review data and compare helpfulness data to the resulting quantitative assessments of honesty of reviews and trustworthiness of reviewers. Finally, we analyze the trust of the reviewers and their correlation with helpfulness in order to classify spam.

1 Background

Internet networks increasingly support the ability for users to express opinions and publicly evaluate website content. These types of user evaluations can be split into two groups: direct evaluation and indirect evaluation. In a direct evaluation scheme, users rate or express opinions about other users. For example, [5] discusses the process of Wikipedia admin elections in which admins vote on prospective admits either in support or opposition. In an indirect evaluation scheme, users rate or express opinions about user-generated content, such as a "Like" on a Facebook post.

The large, accessible, and sometimes anonymous nature of such social networks creates an opportunity for malicious users as it is often difficult to identify and punish the negative behavior. As a result, much research has been devoted to approaches used to determine whether a user is trustworthy, often based upon previous content and behavior. A notable example of this is the Eigentrust algorithm [6]. The algorithm is motivated by peer-to-peer file sharing networks in which malicious users could send programs such as viruses. The risk associated with such files creates the need to establish a peer's credibility. To establish the trustworthiness of user i, the algorithm uses direct evaluations from other users who have downloaded or uploaded from user i and creates a numerical value related to the probability that user i is malicious.

A similiar approach was proposed in [4]. [4] introduces a novel approach based upon a graph model of the reviewers, reviews, and products with the intent of identifying spammers. This algorithm is based upon a node reinforcement method that has been used in situations with conflicting information sources [7]. Like [6], this algorithm assigns a numerical value related the credibility of a given user and review based upon repeated iteration until convergence.

Term	Description	Symbol
Trustiness	Tendency of a user to supply honest reviews	T
Honesty	Tendency of a review agree with other honest reviews	H
Reliability	Tendency of a product to get good scores from users with high trustiness	R
Agreement	Tendency of a review to match other honest reviews	A

Figure 1: Terminology from [4]

With the increasing popularity of online shopping, sites such as Amazon increasingly support the ability for users to both rate and review products that they purchase. Such a system creates the opportunity for malicious users to post en-mass falsified reviews with the intent of influencing the opinion of potential buyers. This is a practice called review spam. In fact, [8] demonstrated that about 10% of reviews were influenced by review spam.

In a review of the field, [9] argues that one key tools of these fields include machine learning methods and natural language processing. The review spam detection approaches proposed by [10, 1, 11] are consistent with that description.

An implemented approach to fighting spam review and identifying reviewer trustworthiness is the Amazon helpfulness metric. Each review asks "Was this review helpful?" in the hope that users will vote truthful reviews more helpful. The actual dynamics of public opinion in these types of systems is much more complicated. [2] used the Amazon helpfulness measure as a case study. The results of this testing indicated that a review is likely to be rated more helpful if it is close to the mean and the variance is low. It also proposed other factors which may make a review helpful such grammatical correctness.

This paper is organized as follows. Section 2 explains the implementation of the algorithm. Section 3 discusses the dataset used for this project and preliminary analysis on it. Section 4 describes the methods used to evaluate the accuracy of the algorithm output. Section 5 describes compares and contrasts the algorithm output with the Amazon helpfulness metric. Section 6 proposes an extension to the algorithm which approximates the solution while using less space and time.

2 Algorithm Description

Adopting the terminology of [4], we wish to quantify the trustiness of the reviewers, the honesty of the reviews, and the reliability of the products, given the a set of reviewers, review scores, and the products that they review. Let review honesty be defined from -1.0 (dishonest) and 1.0 (honest), let reviewer trust be defined from -1.0 (untrustworthy) and 1.0 (trustworthy), and let item reliability be defined from -1.0 (bad item) and 1.0 (good item). A summary of this terminology is given in Figure 1.

The set of reviewers, reviews, and products are interpreted in a graph structure called a "review graph", shown in Figure 2. There is a node for each reviewer, each review, and each product. The topology of the graph is such that each reviewer node connects with one or more review nodes and each review node connects to exactly one product node.

The trustiness, reliability and honesty of each user, product and review is calculated though iterative update. Each value is initially set to 1 and iteratively updated using a set of equations which interdependently relate user trustiness, review honesty, and product reliability. [4] provides

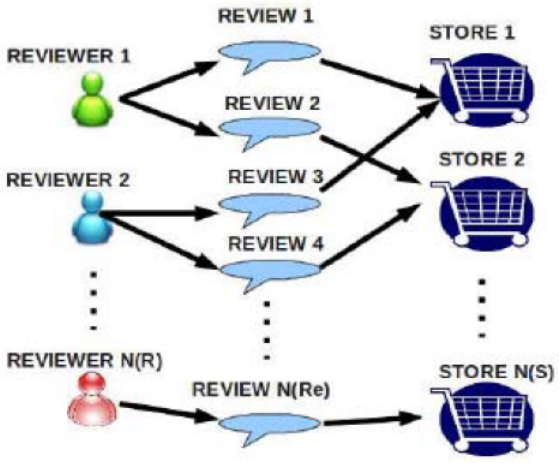

Figure 2: An example of the review graph structure [4]

a justification for the choice of these functions and a full description is rather lengthy. An example is the calculation of trustiness of a given user, r shown in Equation 1 where H_r is the sum of the honesty scores of over all of r's reviews. This equation has the intuitive properties that honest reviews increase the user's trustiness. Note that this equation depends not on the number of reviews, but the sum of honesty values.

$$T(r) = \frac{2}{1 + e^{-H_r}} - 1 \tag{1}$$

The update of all the equations is repeated until convergence is achieved. The pseudo-code is given in Algorithm 1 in the Appendix.

For initial testing, we ran our algorithm on test data with 3 products, 8 users, and 20 reviews. This included 2 spam users who had review scores that deviated significantly from the average for most reviews. This modeled a situation in which the spammer benefited from falsely promoting a bad product. The algorithm correctly detected 2 spam users. The reviewer trust was between 0.6 and 0.9 for the good users and between -0.7 and -0.8 for the spam users.

3 Dataset Analysis

The dataset to be used for this task is the "Web data: Amazon reviews" dataset available via the course website. We focused on a particular dataset used in [12] which is referred to as the "Fine Foods" dataset. It contains 568,454 reviews from 256,059 users. Each review has a number of positive ratings and negative ratings. The overall "helpfulness" of a user may be affected by the number of reviews they authored.

3.1 Method 1

A review with helpfulness 2/2 may not be as reliable as a review with helpfulness 23/25. Therefore, in Method 1, a user's helpfulness will be calculated as a weighted average: a user with 2 reviews of

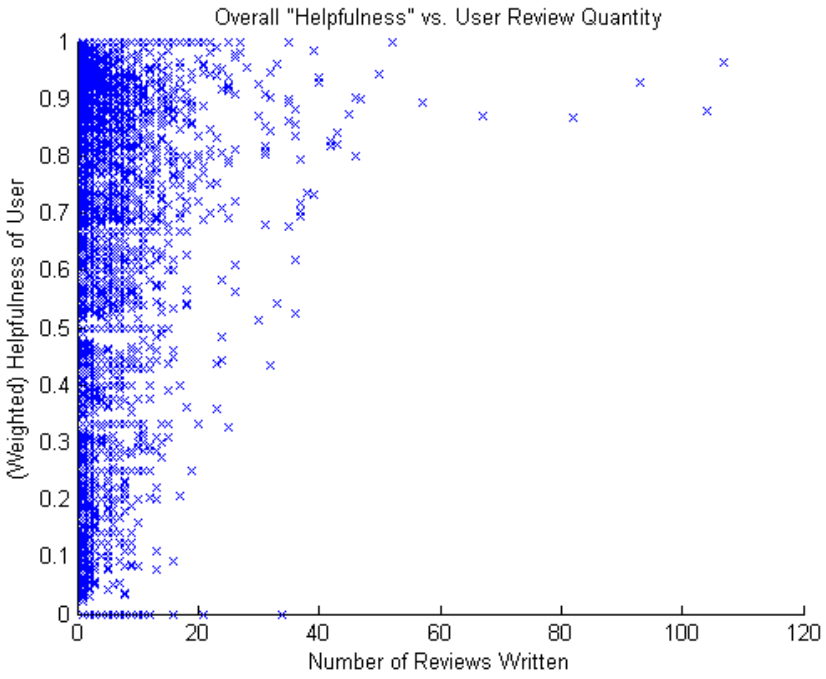

Figure 3: Number of Reviews vs. Net helpfulness

ratings 0/2 and 95/100 will receive a helpfulness rating of 95/102, ensuring that ratings with more votes are weighted more heavily. This relation is plotted in Figure 3.

The trends of users authoring more that 20 reviews is interesting. Though these users are outliers in the data, they consistently tend towards very high helpfulness ratings. The left-hand side of this plot indicates that the average Amazon user writes a small number of reviews. This frequency distribution of authored reviews was investigated and plotted in Figure 4. This figure plots the number of reviews written by a user, and the corresponding frequency of users that authored that number of reviews. As suspected, most Amazon users wrote a very small number of reviews, with 72% of users authoring only a single review. This plot also illustrates that this frequency of review authorship clearly follows a power law distribution.

3.2 Method 2

We can analyze how this helpfulness metric varies with the total number of votes, as shown in Figure 5. Most users that have received a very high number of votes are still achieving very high helpfulness ratings, as if there are "expert" reviewers that are consistently reviewing products well. This plot also shows that there are outliers with a large number of negative votes. For instance, there is one user in the plot who, over all products, was found helpful by 0/103 people. Sampling these consistently "unhelpful" reviewers could prove helpful in spam detection. This plot also indicates that users with a low number of total votes are still achieving these perfect helpfulness rating, exposing an issue with our helpfulness metric when analyzing more typical Amazon users.

Based on this, we propose an alternative method for measuring user helpfulness. A single review's helpfulness score of 95/100 represents 95 helpful votes and 5 unhelpful votes. In Method

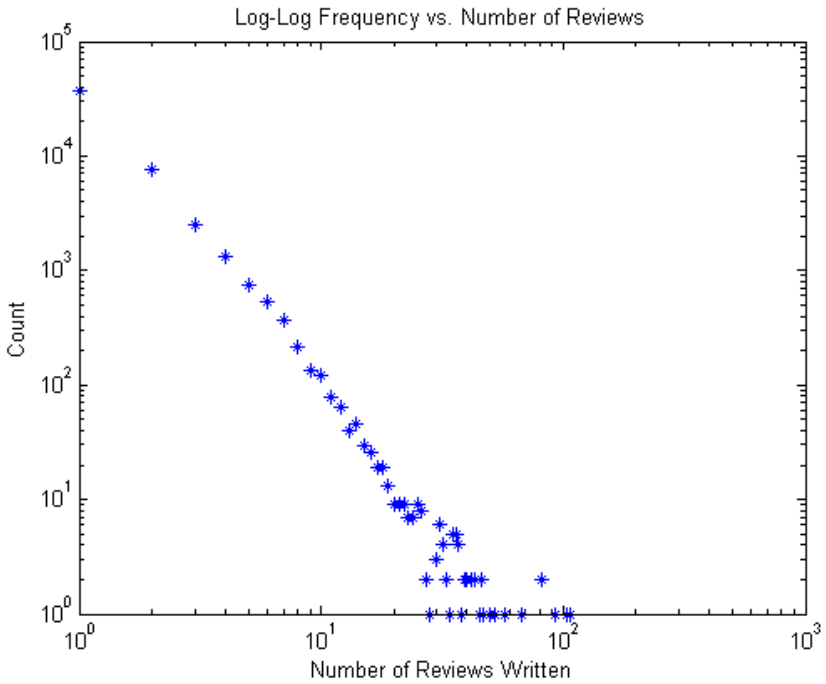

Figure 4: Number of Reviews vs. Number of Users who have done that many reviews

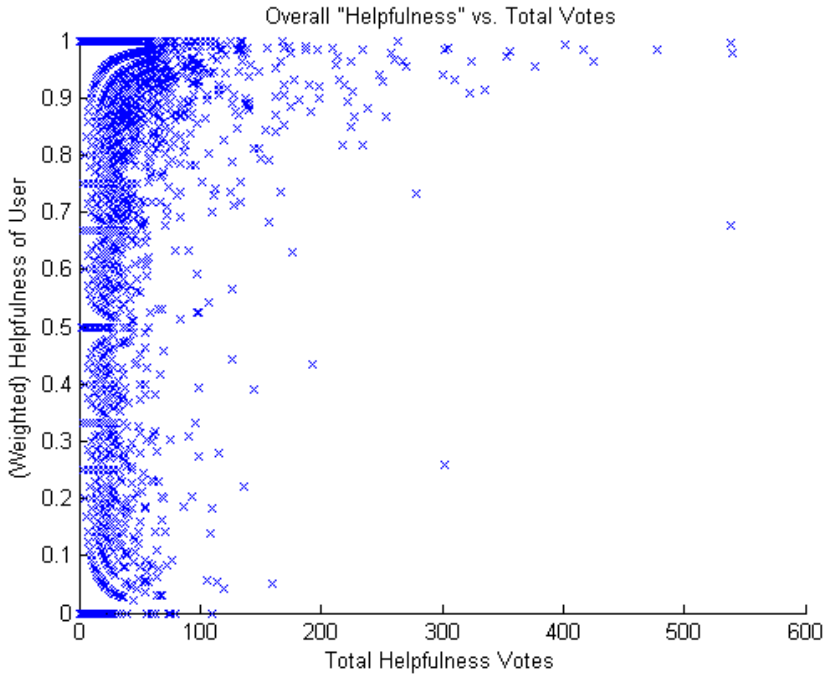

Figure 5: Weighted helpfulness vs. total helpfulness

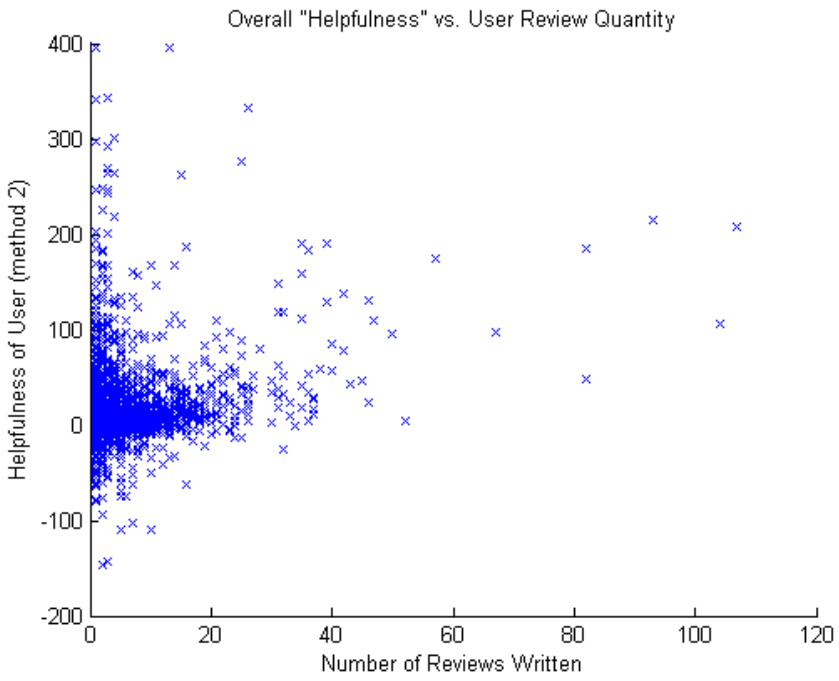

Figure 6: Number of Reviews vs. Alternative helpfulness

2, a user's helpfulness is calculated as the sum of the positive helpfulness votes minus the sum of the unhelpful votes over all of a user's reviews. So in our previous example, a user with ratings 0/2 and 95/100 would be rated 95 - 7 = 88. Conversely, a user with ratings 2/2 and 5/100 would be rated 7 - 97 = -90. This places our "typical" Amazon users on more neutral ground, where ratings of 1/1 = 1 are a more central score. This is plotted in Figure 6.

In contrast to Figure 3, Figure 6 shows that the use of Method 2 causes most users to achieve smaller ratings centered around zero (with the exception of some outliers). Here we achieve the desired quality where most users are achieving an average helpfulness metric, not an outstanding helpfulness metric. The problem of helpfulness scores for users with a low number of reviews written and ratings received is solved.

4 Algorithm Evaluation

One difficulty of evaluating the algorithm is that we do not know beforehand which users are spammers and which are legitimate reviewers. This makes it difficult to evaluate whether the user trustiness values are correct in the sense that they assign low trustiness values to spammers and high trustiness values to legitimate users.

Our approach is to insert additional reviews into the dataset. The reviews are constructed to be from new, known users and to model expected behaviors of certain types of users. The scores of these users can be easily extracted from the final output. We chose to implement 4 basic models of user behavior, which are described in Table 7. More specifically, N users are inserted. For each user, M items are chosen uniformly at random and a review is generated for that user with a score

Model	Description
Downvote Bot	Always review 1.0
Upvote Bot	Always reviews 5.0
Conformist	Always reviews the average score
Random	Reviews with score taken uniformly at random from 1.0 to 5.0 inclusive

Figure 7: User types modeled for evalution

Model	Average	Standard deviation	Median
Downvote Bot	-0.768	.381	-.094
Upvote Bot	0.941	0.096	0.073
Conformist	0.946	0.053	0.942
Random	-0.334	0.603	-0.613

Figure 8: Evaluation results for N = 20 and M = 10

based upon the selected model. The modified dataset is then run through the algorithm.

This approach has many weaknesses. For example, the modeled behaviors are very simple. Additionally, the set of chosen items is a preset size and populated at random. In practice, we might expect spammers to target certain items based upon underlying motivations. However, it provides a good baseline.

We chose $N = 20$ and $M = 10$. The results of this evaluation are chosen in Table 8. The results for the Downvote Bot and the Conformist make sense. The Downvote Bot gets a low score and the Conformist gets a high score. The Random and Upvote Bot models have results which do not seem to match our expectations. The Upvote Bot has a high trustiness and the Random trustiness is not near zero. This can partly be explained by the dataset. Table 9 indicates the breakdown of review scores for the Amazon Fine Foods dataset. The majority of reviews assign a score of 5.0, the highest score. This phenomenon can be seen in the other Amazon datasets. The reasons for this may be the result of self-selection as only enthusiastic users could be expected put in the extra effort of writing an online review. With this in mind, the results for the Upvote Bot and Random model make more sense. The Upvote bot almost always agrees with legitimate users, making it seem like a legitimate user itself. The Random model selected scores uniformly at random. While it sometimes agrees with the majority of users who vote 5.0, it often does not. This explains the Random models slightly negative score with a very large standard deviation.

This highlights a weakness in the implemented algorithm. A reputation based approach in the context of review score means that certain spammers can appear legitimate if they match the existing opinions of the majority of legitimate users. Additionally, this analysis assumes a potential spammer generates many reviews from the same account. A malicious user could subvert the above algorithm by spamming using a large number of single review spam accounts. While generating spam, such accounts would be indistinguishable from valid, new users. Other approaches, such as CAPTCHA, would be needed to compliment reputation based spam detection.

Score	1.0	2.0	3.0	4.0	5.0
Count	51691	29430	42090	79428	357730
Percentage	9.22%	5.25%	7.51%	14.2%	63.8%

Figure 9: Score breakdown for Fine Foods dataset

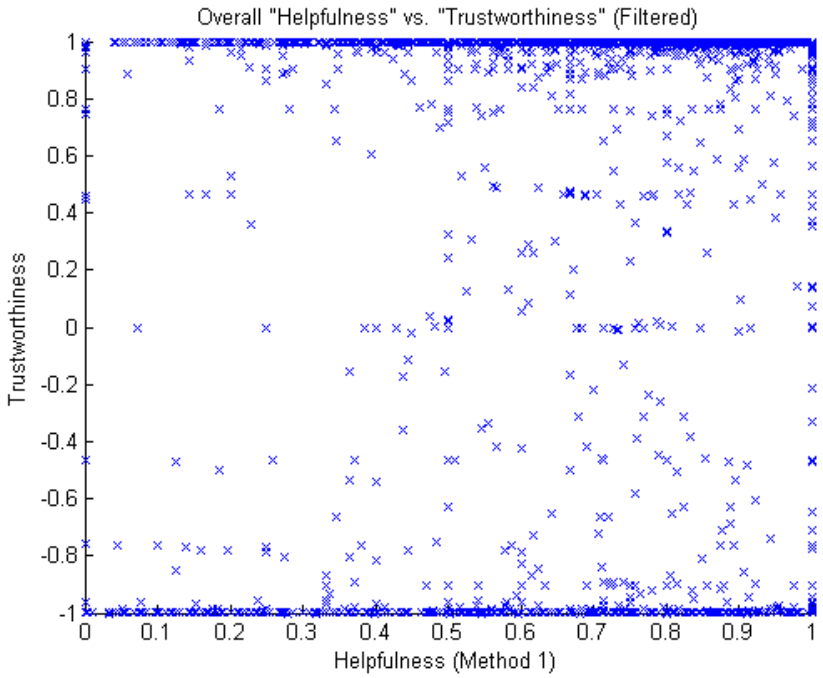

Figure 10: Helpfulness vs. Trustworthiness (Method 1)

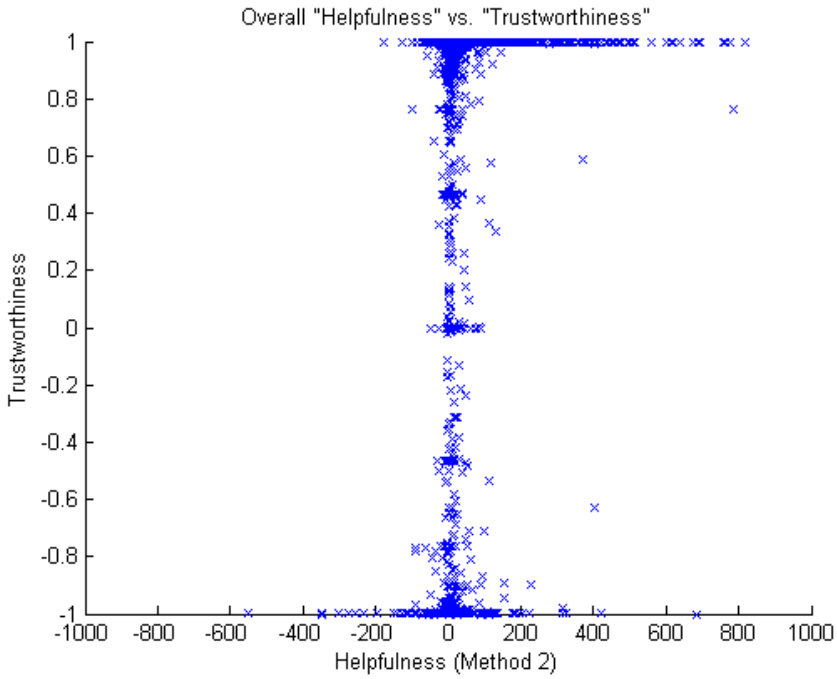

Figure 11: Helpfulness vs. Trustworthiness (Method 2)

5 Relating Helpfulness and Trustworthiness

After implementing the algorithm, the two methods of helpfulness were further evaluated. Figure 10 shows a user's helpfulness (calculated using Method 1) vs. trustworthiness. Note that the data was filtered to include only users with 20 or more reviews. Figure 10 illustrates that the relationship between helpfulness (Method 1) and trustworthiness is weak, wish a correlation value of 0.2414. This figure also indicates trustworthiness has a heavy distribution of 1s and -1s.

Figure 11 shows the relationship between helpfulness (calculated using Method 2) and trustworthiness. Perhaps surprisingly, the helpfulness metric calculated by method 2 is even more weakly tied to trustworthiness than that using Method 1, with a correlation value of 0.0676.

These weak correlations for both helpfulness methods indicate that factors beyond a review's star rating alone dictate whether a user finds it helpful. For instance, it is likely that Amazon users consider the textual content of a review, even if the review's star rating differs heavily from the average. Also, as indicated in background readings, users often conform to the masses when rating reviews.

With this said, the Method 2 helpfulness may provide useful insight in spam detection. Recall that the goal of method 2 was to consider most users as "average" reviewers with helpfulness ratings centered around zero. If we consider the practical applications of this metric, say to reward exceptional users or catch spammers, we are only concerned with outlier helpfulness scores. 88% of users with a helpfulness less than -100 had a trustworthiness less than -.9. 90% of users with a helpfulness greater than 100 had a trustworthiness greater than .9. Furthermore, 100% of users with a helpfulness less than -200 had a trustworthiness less than -.9. 92% of users with a helpfulness greater than 200 had a trustworthiness greater than .9. This method of considering helpfulness and

trustworthiness suggests that outlier values of helpfulness calculation using method 2 could act as a way to identify potential spammers.

6 Algorithm Changes

Size is a major consideration when doing calculations on this type of graph. We propose a method to produce an approximation of the trustiness, honesty, and reliability values with the consideration of limited space and computation time.

In Figure 4, we see that the distribution of user's review amounts approximately obeys a power law and the majority of users review once or twice. Users who review a small number of times generally have a neutral trustiness value (near 0) and thus have limited effect on the rest of the network. We propose an approximation scheme which modifies the graph to remove certain users (and their reviews) to save space. Since the algorithm run-time is linear in the number of users and reviews, we'd expect to also have a proportional effect on computation time.

Each user node is removed with probability $p(d)$ where d is the degree of that user node and p is a given function which returns a value from $(0, 1)$. Then, the original algorithm is run on the modified graph. We will focus on a specialization of this is an algorithm that we will call the k-approximation algorithm. A tunable value k is selected and $p(d)$ is given by:

$$p(d) = \begin{cases} 0 & d \leq k \\ 1 & d > k \end{cases}$$

This corresponds to ignoring users who have written less than k reviews by always assigning them a trustiness of 0. As shown in Figure 4, relatively small values of k could significantly reduce the size of the graph. Note that the case of $k = 0$ corresponds to the original algorithm.

The optimal value of k would depend on the specific topology of the graph. An important metric is the trade off between space and time saved and the error caused by the approximation. Let x_k denote a vector of user trustiness values when the algorithm is run with an approximation value of k. Note that if a user is ignored due to the value of k, this corresponds to a trustiness of 0. A possible measure of the error is weighted Euclidean distance between the results of the original algorithm of the k-approximation, $e(k)$. This is shown in Equation 2 where N indicates the number of users, $x_k(i)$ indicates the trustiness of the i^{th} user with value of k, and w_i indicates the weight associated with that user.

$$e(k) = \sqrt{\sum_{i=0}^{N} w_i (x_0(i) - x_k(i))^2} \tag{2}$$

One approach is to weight each user equally as in Equation 3.

$$w_i = \frac{1}{N} \tag{3}$$

Alternatively, we can weight users in proportion to the number of reviews they have done. This is based upon the observation that our algorithm is more interested with users who have done more reviews. This is shown in Figure 4 with c_i indicating the number of reviews that user i has done.

$$w_i = \frac{c_i}{\sum_{j=0}^{N} c_j} \tag{4}$$

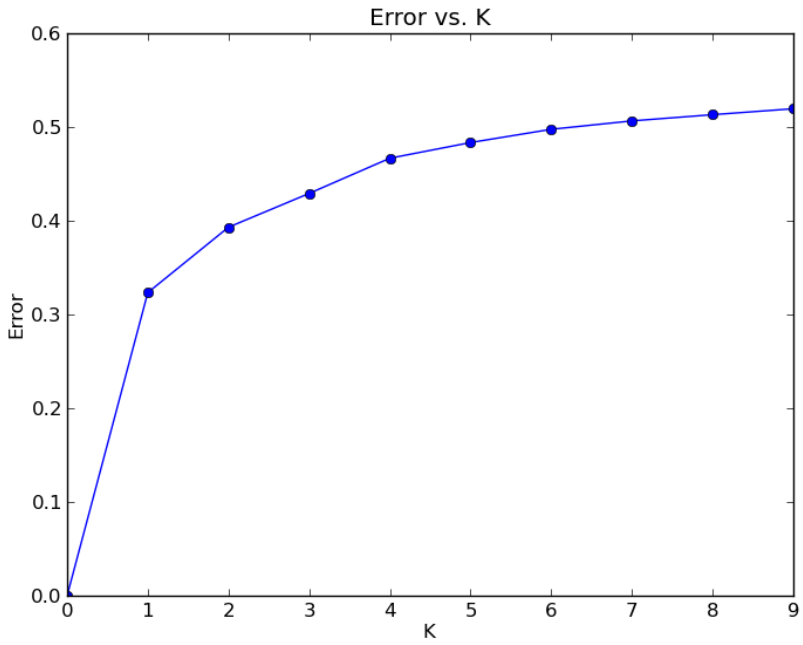

Figure 12: Unweighted error vs. K

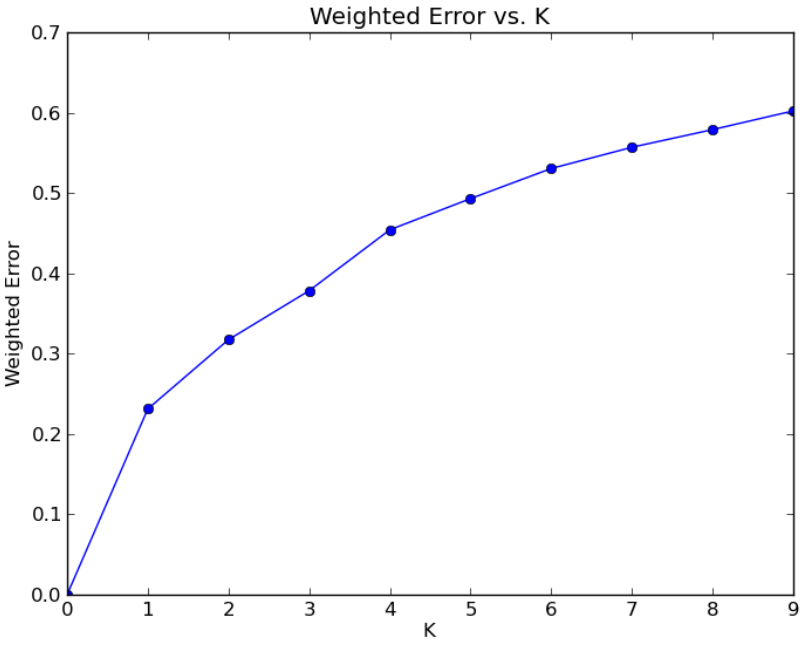

Figure 13: Weighted error vs. K

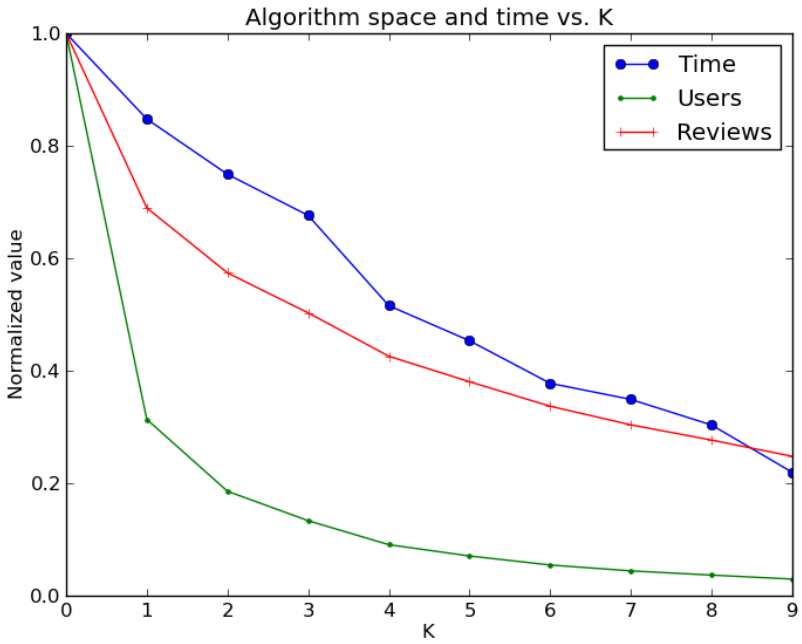

Figure 14: Problem size vs. K

The error for both of these cases are plotted in Figure 12 and 13. They show the expect relationship that increasing K increases the error. Additionally, we would like to characterize the amount of time and space that is saved as a result of this approximation. Normalized values for the time, number of reviews, and number of users vs. K is plotted in Figure 14. The values are normalized by the corresponding value in the original algorithm. This means all values for the case of $k = 0$ are 1. Additionally, the case $k = 1$ took about 80% of the time compared to the case of $k = 0$.

In general, it is difficult to discuss what is the optimal value of k to balance the trade-off between error and efficiency. The exact application will dictate what level of error is acceptable. Additionally, the time savings and error associated with this approach are heavily dependent on the topology of the graph. However, the above described approach provides a powerful tool for when one is primarily interested in high-degree users.

7 Conclusion

In this report, we discussed and implemented an algorithm to assign numerical values of trust to users reviewing items in an online shopping website. This algorithm was applied to a dataset taken from Amazon.com. The algorithm confirmed out expectations when fed with researcher generated users with certain models of behavior. However, this accuracy was limited by the uneven distribution of Amazon review scores. By analyzing Amazon review helpfulness, two metrics were created to calculate a user's overall helpfulness. Relating these metrics to the trustworthiness given by the algorithm showed how user helpfulness could be used to identify trusted and untrusted users in the case of outliers: users with very high or very low helpfulness ratings. Additionally, we

implemented an approximation algorithm which calculated the trust scores with reduced time and space requirements and limited loss in accuracy

A key challenge of this project was the distribution of scores in the dataset, as most of the scores were 5.0. This may not be the case in all review graph and merits future work. Additionally, extensions to this algorithm would incorporate data such as helpfulness or text in calculating trust. Manually tagged spammers could provide more convincing validation of the algorithm's ability to differentiate trusted and untrusted users.

References

[1] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, "Detecting product review spammers using rating behaviors," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 939–948, ACM, 2010.

[2] C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg, and L. Lee, "How opinions are received by online communities: a case study on Amazon.com helpfulness votes," in *Proceedings of the 18th international conference on World wide web*, pp. 141–150, ACM, 2009.

[3] L. Muchnik, S. Aral, and S. J. Taylor, "Social influence bias: A randomized experiment," *Science*, vol. 341, pp. 647–51, 2013.

[4] G. Wang, S. Xie, B. Liu, and P. S. Yu, "Review graph based online store review spammer detection," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pp. 1242–1247, IEEE, 2011.

[5] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg, "Governance in social media: A case study of the wikipedia promotion process.," in *ICWSM*, 2010.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*, pp. 640–651, ACM, 2003.

[7] X. Yin, J. Han, and P. S. Yu, "Truth discovery with multiple conflicting information providers on the web," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, no. 6, pp. 796–808, 2008.

[8] E. Gilbert and K. Karahalios, "Understanding deja reviewers," in *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pp. 225–228, ACM, 2010.

[9] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and trends in information retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.

[10] N. Jindal and B. Liu, "Review spam detection," in *Proceedings of the 16th international conference on World Wide Web*, pp. 1189–1190, ACM, 2007.

[11] F. Li, M. Huang, Y. Yang, and X. Zhu, "Learning to identify review spam," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pp. 2488–2493, AAAI Press, 2011.

[12] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews," in *Proceedings of the 22nd international conference on World Wide Web*, pp. 897–908, International World Wide Web Conferences Steering Committee, 2013.

8 Appendix

Algorithm 1 Calculate trustiness, honesty, and reliability

Require: Δt, δ, maxRound to be user defined
  Input: Items I, reviews RE, and reviewers R_i.
  Output: Reliability R, honesty H, and trustiness T.
  round = 0
  Assign all values in H, T, R, A to be 1
  while round < maxRound **do**
    for re ∈ RE **do**
      // Update honesty for each review
      compute H(re) using equations described in the paper
    end for
    for r ∈ R_i **do**
      // Update trustiness of each user
      compute T(r) using given equations described in the pape
    end for
    for i ∈ I **do**
      // Update reliability of each item
      compute R(i) using equations described in the paper
    end for
    for re ∈ RE **do**
      // Update how much each review agrees with other honest reviews
      compute A(re) using given equations described in the pape
    end for
    round++
  end while
  Output: R, H, T