Detecting Product Review Spammers using Rating Behaviors

Ee-Peng Lim
School of Information Systems
Singapore Management University
eplim@smu.edu.sg

Viet-An Nguyen
School of Information Systems
Singapore Management University
vanguyen@smu.edu.sg

Nitin Jindal
Department of Computer Science
University of Illinois at Chicago
nitin.jindal@gmail.com

Bing Liu
Department of Computer Science
University of Illinois at Chicago
liub@cs.uic.edu

Hady W. Lauw
Institute for Infocomm Research, Singapore
hwlauw@i2r.a-star.edu.sg

ABSTRACT

This paper aims to detect users generating spam reviews or review spammers. We identify several characteristic behaviors of review spammers and model these behaviors so as to detect the spammers. In particular, we seek to model the following behaviors. First, spammers may target specific products or product groups in order to maximize their impact. Second, they tend to deviate from the other reviewers in their ratings of products. We propose scoring methods to measure the degree of spam for each reviewer and apply them on an Amazon review dataset. We then select a subset of highly suspicious reviewers for further scrutiny by our user evaluators with the help of a web based spammer evaluation software specially developed for user evaluation experiments. Our results show that our proposed ranking and supervised methods are effective in discovering spammers and outperform other baseline method based on helpfulness votes alone. We finally show that the detected spammers have more significant impact on ratings compared with the unhelpful reviewers.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms, Measurement, Experimentation

1. INTRODUCTION

1.1 Motivation

Web spam refers to all forms of malicious manipulation of user generated data so as to influence usage patterns of the data. Examples of web spam include search engine spam [1], email spam [3], and video spam [2]. In this paper, we focus on spam found in online product review sites commonly known as **review spam** or **opinion spam** [9, 10]. Review spam is designed to give unfair view of some products so as to influence the consumers' perception of the products by directly or indirectly inflating or damaging the product's reputation. In [6], it was found that 10 to 15% of reviews essentially echo the earlier reviews and may potentially be influenced by review spam.

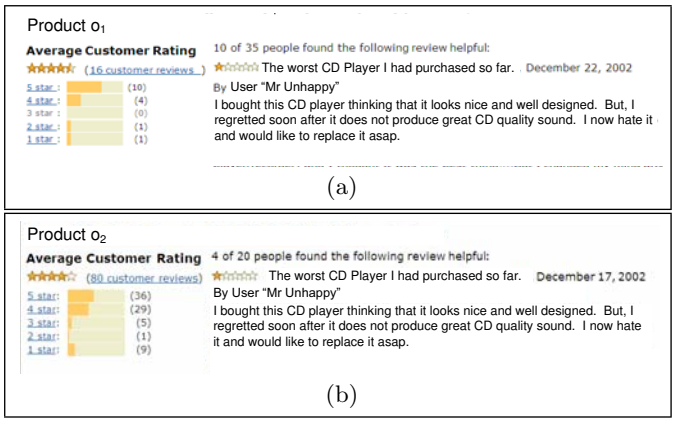


Figure 1: Review Spam Examples

Consider Figure 1a which illustrates a review for product o_1 by user "Mr Unhappy". The review is highly negative with 1-star rating in contrast with the high overall 4.5 star rating. This review does not cause any alarm until we find another highly negative review by the same user on a different product o_2 and both reviews are identical in content (see Figure 1b). Since identical review content for different products reflects a strong bias or a lack of seriousness, and the user's ratings are very different from the rest, we consider the two reviews likely to be spam and the user likely to be a spammer. Products o_1 and o_2 have 16 and 80 reviews respectively.

It is not clear how much review spam exists in online product review sites but their existence causes several problems including unfair treatment of products either independently

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

or in comparison with other similar products. Both under-rating (or "bad mouthing") and over-rating (or "ballot stuffing") affect the sales performance of the affected products especially for review sites that also offer buying and selling of products. When consumers rely on reviews from spammers to purchase products, they could be disappointed by purchased products not meeting their expectation, or misjudging good products. It is thus an important task to detect review spam and remove them so as to protect the genuine interests of consumers and product vendors.

Detecting review spam is a challenging task as no one knows exactly the amount of spam in existence. Due to the openness of product review sites, spammers can pose as different users (known as "sockpuppeting") contributing spammed reviews making them harder to eradicate completely. Spam reviews usually look perfectly normal until one compares them with other reviews of the same products to identify review comments not consistent with the latter. The efforts of additional comparisons by the users make the detection task tedious and non-trivial. One approach taken by review site such as Amazon.com is to allow users to label or vote the reviews as helpful or not. Unfortunately, this still demands user efforts and is subject to abuse by spammers.

The state-of-the-art approach to review spam detection is to treat the reviews as the target of detection [10]. This approach represents a review by review-, reviewer- and product-level features, and trains a classifier to distinguish spam reviews from non-spam ones. However, these features may not provide direct evidence against the spammed review. For the example in Figure 1, we rely on (a) comparison with other ratings on the same products by the contributor, and (b) identical review content for different products by the contributor. Both are behaviors of reviewer that deviate from normal practice and are highly suspicious of review manipulation. This suggests that one should focus on detecting spammers based on their spamming behaviors, instead of detecting spam reviews. In fact, the more spamming behaviors we can detect for a reviewer, the more likely the reviewer is a spammer. Subsequently, the reviews of this reviewer can be removed to protect the interests of other review users.

1.2 Review Spammer Detection

In this paper, we address the problem of **review spammer detection**, or finding users who are the source of spam reviews. Unlike the approaches for spammed *review* detection, our proposed review spammer detection approach is *user centric*, and *user behavior driven*. A user centric approach is preferred over the review centric approach as gathering behavioral evidence of spammers is easier than that of spam reviews. A review involves only one reviewer and one product. The amount of evidence is limited. A reviewer on the other hand may have reviewed a number of products and hence has contributed a number of reviews. The likelihood of finding evidence against spammers will be much higher. The user centric approach is also scalable as one can always incorporate new spamming behaviors as they emerge.

The main building blocks of the spamming behavior detection step are the **spamming behavior models** based on different review patterns that suggest spamming. Each model assigns a numeric spamming behavior score to each reviewer by measuring the extent to which the reviewer practises spamming behavior of a certain type. In this paper, we mainly rely on patterns of review content and ratings to define four different spamming behavior models, i.e., (a) *targeting product* (**TP**); (b) *targeting group* (**TG**); (c) *general rating deviation* (**GD**); and (b) *early rating deviation* (**ED**). To assign an overall numeric spam score to each user, we combine the spam scores of the user's different spamming behaviors using linear weighted combination. The weights on the different component spam scores can be empirically defined or learnt automatically.

Our proposed behavior models shy away from deep natural review text understanding and opinion extraction mainly to avoid high computational costs and performance pitfalls due to inaccurate text analysis. Should text analysis be accurate enough to extract opinions from review text, our spamming behavior models can be extended to consider review content. Nevertheless, content analysis can be computationally costly and we shall leave this to our future research.

The remainder of the paper is organized as follows. Section 2 covers the relevant related work. Section 3 describes the data representation and dataset used in this research. Our proposed characterizations of target-based and deviation-based spamming behaviors are given in Sections 4 and 5 respectively. Section 6 describes an user evaluation experiment of our proposed unsupervised spammer scoring methods. The supervised version of the spammer scoring method is given in Section 7. Section 8 concludes the paper.

2. RELATED WORK

In this section, we survey the related research in review spammer detection and compare them with that presented in this paper.

Review Opinion and Sentiment Mining. In the context of opinion mining, there are several efforts in extracting and aggregating positive and negative opinions from product reviews [7, 16]. These works focus mainly on text mining and extraction aspect of review content. While these extracted opinions can be useful in identifying bias opinions in reviews, they do not address spam detection unless being used to derive other more relevant features for the task.

Review Spam Detection. Review spam detection is a relatively new research problem which has not yet been well studied. A preliminary study was reported in [9]. A more in-depth investigation was given in [10]. In this work, three types of review spam were identified, namely *untruthful reviews* (reviews that promote or defame products), *reviews on brands* but not products, and *non-reviews* (e.g., advertisements). By representing a review using a set of review-, reviewer- and product-level features, classification techniques are used to assign spam labels to reviews. In particular, untruthful review detection is performed by using duplicate reviews as labeled data.

Reviews usually come with ratings. Detecting unfair or fraudulent ratings has been studied in several works including [5, 17]. The techniques used include: (a) clustering ratings into unfairly high ratings and unfairly low ratings, and (b) using third party ratings on the producers of ratings and ratings from less reputable producers are then deemed as unfair. Once unfair ratings are found, they can be removed to restore a fair item evaluation system. These works did not address review spammer detection directly. They usually did not conduct evaluation of their techniques on real data.

Review helpfulness prediction is closely related to review spam detection. The former aims to differentiate reviews of

different helpfulness. A helpful review is one that is informative and useful to the readers. The purpose of predicting review helpfulness is to help review sites to give feedbacks to the review contributors and to help readers choose and read high quality reviews. A classification approach to solving helpfulness prediction using review content and meta-data features was developed in [12]. The meta-data features used are review's rating and the difference between the review rating and the average rating of all reviews of the product. Liu et. al proposes to derive from reviews content features that correspond to informativeness, readability and subjectiveness aspects of review [15]. These features are then used to train a helpfulness classification method.

Item Spam Detection. Wu and others attempted to identify items that are targets of spamming by identifying singleton reviews on the reviewed items[17]. Singleton reviews are the reviews written by users who contribute only one review each. These users subsequently contribute no other reviews. Proportion of positive singleton reviews, concentration of positive singleton reviews, and rating distortion caused by singleton reviews are thus used to analyse possibly spammed hotels in TripAdvisor.

Comparison with Review Spam and Item Spam Detections. Instead of detecting review and item spams, this paper focuses on detecting reviewers who are spammers. The three problems are intimately related as solving one may help to solve the other two. For example, knowing which reviews are spam can provide pointers to review spammers and spammed items. Given that there are no perfect solutions for the three problems, it is still necessary to conduct research on each of the problems. This paper also takes a unique approach to detect review spammers by examining their rating behaviors and deriving features that can be used in review spammer detection methods.

To the best of our knowledge, this paper represents the first attempt to solve the review spammer detection problem. It uses rating behavior features to detect review spammers. The proposed method can further help to find review and item spam, or be extended to use the results of review and item spam detection to improve its accuracy. In our companion paper [11], rule mining is used to help find atypical review patterns, which may indicate spam activities.

Helpful Review Prediction. Amazon.com allows users to vote a review to be helpful or not. This helpfulness votes are manually assigned and are thus subjective and possibly abused. Danescu-Niculescu-Mizil et. al found that a strong correlation between proportion of helpful votes of reviews and the deviation of the review ratings from the average ratings of products [4]. This correlation illustrates that helpful votes are generally consistent with average ratings. The study is however conducted at the collection level and does not provide evidence to link between spam and helpfulness votes.

Detecting spam and predicting helpfulness are two separate problems since not all unhelpful reviews are spam. A poorly written review can be unhelpful but is not a spam. Spam reviews usually target specific products while unhelpful votes may be given to any products. Given the motive driven nature of spamming activities, review spam detection will therefore require an approach different from unhelpful review detection.

Outlier Detection. Finding review spammers is also related to outlier detection [13]. The spammers are 'outliers' in the sense of behaving differently from other reviewers who are non-spammers. Prior literature mostly attempts to define outliers by some notion of distance. However, spamming behaviors are complex and not easily captured by distance. For instance, the target-based spamming behavior described in Section 4 may in fact involve very similar or duplicate reviews. While the deviation-based behaviors described in Section 5 are in some sense distance-based, it has to incorporate spamming-specific factors such as how early the deviation occurs.

3. REVIEW DATASET

Amazon Dataset. In this research, we assume the product review data follows the database schema used by Amazon.com. Each product has a profile page that links to a set of reviews contributed by different users. Each product may also have a brand and zero or more product attributes assigned. These attributes may vary depending on the product type. Taking book product as an example, the relevant attributes include author, publisher, publication year, and price.

Each user can contribute one or multiple reviews to a product. Contributing multiple reviews by the same user is counter-intuitive but is allowed for updating previous reviews and to perhaps add new comments on the reviewed products. Each review consists of both a textual comment and a numeric rating score which takes a value between 1 to 5. For simplicity, we assume that every rating score is normalized to the [0,1] range. As a user reads a review, she can optionally cast a binary vote (helpful or not-helpful) to the review. The helpfulness voters' identities are assumed to be non-available.

For evaluation purposes, we use a Amazon's dataset downloaded using APIs. This dataset, known as *Manufactured Products (`MProducts`)*, has product attributes including: (a) ProductID; (b) Sales Rank; (c) Brand; (d) Sales Price; and (e) Product Categories.

Pre-processing. Several data preprocessing steps are performed on the above dataset before it is used.

- *Removal of anonymous users*: We first remove anonymous users and their reviews. Each anonymous user id may be used by multiple persons.

- *Removal of duplicate products*: We also identify sets of duplicate products in the dataset and remove them except for one representative product per set. This step is necessary since Amazon.com maintains duplicate products (essentially the same product with some very minor variations, e.g. color) and replicates reviews across them. In other words, given a set of duplicate products, a review written on a product will be replicated and added to other products in this set. Using the identical reviews, we detect sets of such products and randomly choose one representative product from each set to keep while removing others.

- *Removal of inactive users and unpopular products*: To focus on users who are active and products that attract some user attention, our dataset includes only users with no fewer than 3 reviews and products with no fewer than 3 reviews. This is done by repetitively applying minimum number of reviews threshold on

Table 1: Dataset Statistics

	Number (before preprocessing)
U: set of users	11,038 (313,120)
O: set of objects	5,693 (32,075)
V: set of reviews	48,894 (404,637)
E: set of ratings	as above

users and products in alternate order until all users and products meet the threshold.

- *Resolution of brand name synonyms*: We found out that the products' brand names suffer from the synonymy problem which involves multiple brand names assigned to the same brands. E.g., the brand "HP" may also be called "Hewlett Packard" or "HP Technology". Brand synonyms prevent us from grouping products by their brands. Fortunately, there are only few hundreds of brand names in `MProducts`. We therefore were able to resolve synonyms by manual inspection and replace them by the representative brand names.

Finally, we arrive at the preprocessed dataset with statistics shown in Table 1[1]. Another larger Amazon dataset on book items has also been downloaded and preprocessed in the similar manner. While our observations for the Book dataset generally are applicable to those of *Manufactured Products* dataset, this paper does not include the discussion here to conserve space.

Notations. In the following, we list the notations to be used in the subsequent sections.

- $\mathbf{U} = \{u_i\}$: set of users
- $\mathbf{O} = \{o_j\}$: set of objects[2]. Let $\mathbf{B} = \{b_k\}$ denote the set of brands and $b(o_j) \in \mathbf{B}$ denote the brand of object o_j.
- $\mathbf{V} = \{v_k\}$: set of reviews v_k's. The review v_k comes with a rating e_k.
- $\mathbf{E} = \{e_k\}$: set of ratings e_k's ($e_k \in [0, 1]$).
- $u(v_k) = u(e_k)$: user of review v_k and rating e_k.
- $o(v_k) = o(e_k)$: object of review v_k and rating e_k.
- $r(v_k) = r(e_k)$: order position ($\in \{1, \cdots\}$) of review v_k and rating e_k with respect to the reviewed object.
- $\mathbf{V}_{ij} = \{v_k | u(v_k) = u_i \wedge o(v_k) = o_j\}$: set of reviews from user u_i to object o_j.
- $\mathbf{E}_{ij} = \{e_k | u(e_k) = u_i \wedge o(e_k) = o_j\}$: set of ratings from user u_i to object o_j.
- $\mathbf{E}_{i*} = \cup_j \mathbf{E}_{ij}$: set of all ratings by user u_i
- $\mathbf{E}_{*j} = \cup_i \mathbf{E}_{ij}$: set of all ratings on object o_j

4. TARGET-BASED SPAMMING

To game the online review systems, we hypothesize that a spammer will direct most of his efforts to promote or victimize a few products or product lines which are collectively known as the *targeted products* or *targeted product groups*. He is expected to monitor targeted products and product groups closely and mitigate the ratings when time is appropriate. We thus define three spamming behaviors involving targeted products and product groups and derive their respective spam scores for each reviewer representing the extent to which he practises the behaviors.

[1]Numbers in the parentheses are the raw counts before applying filter.

[2]In this paper, *object* and *product* are used interchangeably.

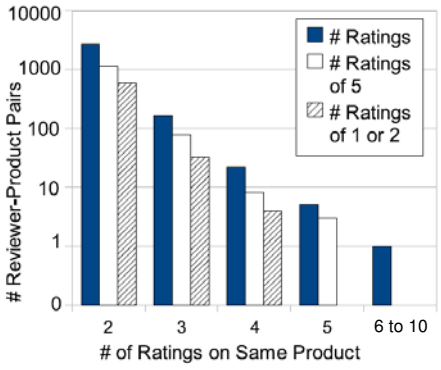


Figure 2: Single Product Multiple Reviews/Ratings in `MProducts`

4.1 Targeting Products

Spamming against a specific targeted product is an act that can be most easily observed by the number of reviews (also ratings) on the product. In the `MProducts` dataset, there are several reviewers who contribute multiple reviews to the same products, i.e., $|\mathbf{E}_{ij}| \geq 2$ (or $|\mathbf{V}_{ij}| \geq 2$), as shown in Figure 2. For each $|\mathbf{E}_{ij}| = x$ value, the figure shows the number of reviewer-product pairs (in log_{10} scale) having x ($x > 1$) reviews. The figure also shows the distribution of pairs with all 5 ratings and pairs with all 1 or 2 ratings.

In `MProducts`, 2874 reviewer-product pairs involve multiple reviews/ratings. This number is considered very small given the much larger numbers of reviewers and products. Most of these pairs, 1220 of them, involve only ratings of 5 compared with 624 of them involving only ratings of 1 or 2. This observation is consistent with the many high ratings in the dataset. Interestingly, several pairs involve both high and low ratings ($2874 - 1220 - 624 = 1030$ of them) suggesting that the corresponding reviewers changed their ratings on the same products, which may be caused by different reasons. For example, a reviewer may revise rating on a product after finding out the correct way to use it. Note that there are also very few pairs involving 3 to as many as 10 ratings.

4.1.1 Rating Spamming

Reviewers who are involved in reviewer-product pairs with larger number of ratings are likely to be spammers, especially when the ratings are not very different from one another. We thus define the following user spam score function $c_{p,e}(u_i)$ for user u_i based on *rating spamming behavior on some targeted products*:

$$c_{p,e}(u_i) = \frac{s_i}{Max_{u'_i \in \mathbf{U}} s_{i'}} \quad (1)$$

where s_i represents the unnormalized spammer score of u_i defined by:

$$s_i = \sum_{e_{ij} \in \mathbf{E}_{ij}, |\mathbf{E}_{ij}| > 1} |\mathbf{E}_{ij}| \cdot sim(\mathbf{E}_{ij}) \quad (2)$$

The denominator of (1) returns the maximum s_i among all

users. The function $sim()$ is a similarity function comparing ratings in a given set and is defined as:

$$sim(\mathbf{E}_{ij}) = 1 - \underset{e_k, e_{k'} \in \mathbf{E}_{ij}, k<k'}{Avg} |e_k - e_{k'}| \quad (3)$$

Based on the above spam score function, reviewers with large proportions of ratings involved as multiple similar ratings on products are to be assigned high spam scores.

4.1.2 Review Text Spamming

As a user spams a product with multiple ratings, he is also likely to spam the product with multiple review texts. Such review texts are likely to be identical or look similar so as to conserve spamming efforts. On the other hand, there are also users who contribute multiple reviews to the same products for genuine reasons. For example, some may divide their reviews into smaller reviews. Some make minor changes to their reviews to improve clarity. Hence, it is necessary to distinguish such genuine cases from the more dubious ones. We define the similarity between two reviews v_k and $v_{k'}$, $sim(v_k, v_{k'})$, by:

$$sim(v_k, v_{k'}) = cosine(v_k, v_{k'}) \quad (4)$$

where each review text is represented by a bag of bi-grams and $cosine(v_k, v_{k'})$ is the cosine similarity of the bi-gram TFIDF vectors of v_k and $v_{k'}$. When $sim(v_k, v_{k'}) = 1$, we consider the two reviews identical. TFIDF refers to the frequency $\times$ inverse document frequency of a bi-gram. By considering IDF, the dominance of popular bi-grams will be reduced. Given a set of reviews $\mathbf{V}_{ij}$ ($|\mathbf{V}_{ij}| > 1$), we derive a similarity score $sim(\mathbf{V}_{ij})$ as follows:

$$sim(\mathbf{V}_{ij}) = \underset{v_k, v_{k'} \in \mathbf{V}_{ij}, k<k'}{Avg} sim(v_k, v_{k'}) \quad (5)$$

Then, we define the user spam score function $c_{p,v}(u_i)$ for user u_i based on *review spamming behavior on some targeted products*:

$$c_{p,v}(u_i) = \frac{s'_i}{Max_{u'_i \in \mathbf{U}} s'_{i'}} \quad (6)$$

where

$$s'_i = \sum_{v_{ij} \in \mathbf{V}_{ij}, |\mathbf{V}ij|>1} |\mathbf{V}_{ij}| \cdot sim(\mathbf{V}_{ij}) \quad (7)$$

4.1.3 Combined Spam Score

We combine the above two spam scores due to single product multiple reviews by taking the average:

$$c_p(u_i) = \frac{1}{2}\left(c_{p,e}(u_i) + c_{p,v}(u_i)\right) \quad (8)$$

4.2 Targeting Product Groups

The above two spam models assign spam scores to each user based on their multiple ratings and review texts on the same products. In this section, we introduce a spam behavior model (also known as **single product group multiple ratings**) defined on the pattern of spammers manipulating ratings of a set of products sharing some common attribute(s) within a short span of time. For example, a spammer may target several products of *same brand* within a few hours. This pattern of ratings saves the spammers' time as they do not need to log on to the product review system many times. To achieve maximum impact, the ratings given to these target group of products are either very high or low. We can thus further define two types of single product group multiple ratings behavior, one for very high ratings and another for very low ones. The users who contribute ratings on the targeted product groups are likely to be spam.

4.2.1 Single Product Group Multiple High Ratings

To model rating behavior that involves very high ratings on products sharing the same attribute by the same user within a short span of time, we divide the whole time period into small disjoint time windows of fixed-size and derive clusters of very high ratings. Such rating clusters are suspicious of promoting a product group. The **high rating cluster** by user u_i to a product group b_k (identified by some product attribute) in time window w is defined as follows:

$$\mathbf{E}^{\mathcal{H}}_{ik}(w) = \{e_{ij} \in \mathbf{E}_{i*} \mid o_j \in b_k \wedge t(e_{ij}) \in w \wedge e_{ij} \in HRatingSet\} \quad (9)$$

where $HRatingSet$ is a set of ratings that are considered very high. At Amazon.com site, a raw rating score of 5 (or a normalized rating score of 1) is considered very high. Hence, we use $HRatingSet = \{1\}$ by default.

We assume that only sufficiently large $\mathbf{E}^{\mathcal{H}}_{ik}$'s can possibly be due to spams and impose a minimum size threshold of $minsize^{\mathcal{H}}$. Only clusters with size larger than $minsize^{\mathcal{H}}$ will thus be retained in $\mathbf{C}^{\mathcal{H}}_i$'s (defined below) for further spam scoring. The time window w should be chosen to be sufficiently large enough for multiple ratings to be given by the same user but small enough to capture the burstiness of ratings. In our experiments, we have empirically set w to be a day interval and $minsize^{\mathcal{H}} = 3$ which give reasonably good results (see Section 6). The product attribute used for `MProducts` and `Books` datasets are brand and publisher respectively.

$$\mathbf{C}^{\mathcal{H}}_i = \cup_{k,w}\{\mathbf{E}^{\mathcal{H}}_{ik}(w) | \ |\mathbf{E}^{\mathcal{H}}_{ik}(w)| \geq minsize^{\mathcal{H}}\} \quad (10)$$

The spam score a user u_i based on single product group multiple high ratings behavior is thus defined by:

$$c_{g,\mathcal{H}}(u_i) = \frac{\mathbf{C}^{\mathcal{H}}_i}{Max_{u'_i \in \mathbf{U}} \mathbf{C}^{\mathcal{H}}_{i'}} \quad (11)$$

4.2.2 Single Product Group Multiple Low Ratings

Instead of single product group multiple high ratings, spammers may adopt the opposite behavior, i.e., assigning several low ratings (usually the raw rating score of 1 or 2 out of 5) to demote a group of products. The motive here is to create a negative perception of the affected products so as to reduce their sales. We define a low rating cluster by user u_i to a product group b_k sharing some attribute a in time window w as follow:

$$\mathbf{E}^{\mathcal{L}}_{ik}(w) = \{e_{ij} \in \mathbf{E}_{i*} \mid o_j \in b_k \wedge t(e_{ij}) \in w \wedge e_{ij} \in LRatingSet\} \quad (12)$$

We use $LRatingSet = \{0, 0.25\}$ corresponding to the raw rating scores of 1 and 2. The set of low rating clusters by user u_i that meet the minimum size threshold $minsize^{\mathcal{L}}$ is captured by:

$$\mathbf{C}^{\mathcal{L}}_i = \cup_{k,w}\{\mathbf{E}^{\mathcal{L}}_{ik}(w) | \ |\mathbf{E}^{\mathcal{L}}_{ik}(w)| \geq minsize^{\mathcal{L}}\} \quad (13)$$

We empirically set w to be a day interval and $minsize^{\mathcal{L}} = 2$ in our experiments. A smaller $minsize^{\mathcal{L}}$ compared to

$minsize^{\mathcal{H}}$ is chosen as small low rating clusters look more suspicious than small high rating clusters due to an overall much smaller proportion of low ratings in the entire dataset.

We then define the spam score of a user u_i based on single product group multiple low ratings as:

$$c_{g,\mathcal{L}}(u_i) = \frac{\mathbf{C}_i^{\mathcal{L}}}{Max_{u'_i \in \mathbf{U}} \mathbf{C}_{i'}^{\mathcal{L}}} \tag{14}$$

4.2.3 Combined Spam Score

We again combine the above two spam scores due to single product group multiple ratings by simply taking the average:

$$c_g(u_i) = \frac{1}{2}(c_{g,\mathcal{H}}(u_i) + c_{g,\mathcal{L}}(u_i)) \tag{15}$$

5. DEVIATION-BASED SPAMMING

5.1 General Deviation

A reasonable rater is expected to give ratings similar to other raters of the same product. As spammers attempt to promote or demote products, their ratings could be quite different from other raters. General deviation is therefore a possible rating behavior demonstrated by a spammer. We define the deviation of a rating e_{ij} as its difference from the average rating on the same product:

$$d_{ij} = e_{ij} - \underset{e \in \mathbf{E}_{*j}}{\text{Avg}}\ e \tag{16}$$

The spam score of a user u_i based on general deviation is thus defined as the average deviation of his or her ratings:

$$c_d(u_i) = \underset{e_{ij} \in \mathbf{E}_{i*}}{\text{Avg}}\ |d_{ij}| \tag{17}$$

5.2 Early Deviation

Early deviation captures the behavior of a spammer contributing a review spam soon after product is made available for review. Such spams are likely to attract attention from other reviewers allowing spammers to manipulate the views of subsequent reviewers. It will take the victimized products several good ratings from other genuine reviewers to recover from these early low ratings.

The early deviation model thus relies on two pieces of information for assigning an early deviation behavior score: (a) deviation of each rating from the mean rating of the rated product, and (b) weight of each rating indicating how early the rating was given. The deviation of a rating of a user u_i on an object o_j is defined as the difference between e_{ij} from the mean of ratings on object o_j:

$$d_{ij} = e_{ij} - \underset{e \in \mathbf{E}_{*j}}{\text{Avg}}\ e \tag{18}$$

The weight of each rating e_{ij} is denoted by w_{ij} and is defined as follows:

$$w_{ij} = \frac{1}{(r_{ij})^{\alpha}} \tag{19}$$

where r_{ij} refers to the rank order of e_{ij} among ratings assigned to the rated product. We set $r_{ij} = i$ for the ith rating (or review) of a product. α is a parameter greater than 1 to accelerate the rate of decay for w_{ij}.

Thus, the spam score of a user u_i is defined as:

$$c_e(u_i) = \frac{\sum_{e_{ij} \in \mathbf{E}_{i*}} (|d_{ij}| \times w_{ij})}{\sum_{e_{ij} \in \mathbf{E}_{i*}} w_{ij}} \tag{20}$$

6. USER EVALUATION

6.1 Objectives

To evaluate the performance of different solution methods, we need some ground truth labels of users. Given that such labels do not exist in the public, we thus decide to conduct user evaluation on different methods derived from the spamming behaviors proposed in this paper. We want to know which methods are able to detect review spammers more accurately. Should any method fails to detect spammers correctly, we want to know how the method is led to the wrong conclusion. We also hope to elicit other spamming behaviors we may have missed out.

The spammer detection methods to be evaluated are based on: (a) single product multiple reviews behavior (**TP**) only; (b) single product group multiple reviews behavior (**TG**) only; (c) general deviation (**GD**) behavior; and (d) early deviation (**ED**) behavior with α=1.5. Each method works by ranking the users by decreasing behavior score (i.e., $c^p(u_i)$, $c_e^g(u+i)$, $c_d(u_i)$, and $c_e(u_i)$) order. The highly ranked users are more likely to be spammers. We also introduce a new combined method (**ALL**) that considers all the behaviors using a combined behavior score $c(u_i)$ as follows:

$$c(u_i) = \frac{1}{2}c_p(u_i) + \frac{1}{4}c_g(u_i) + \frac{1}{8}c_d(u_i) + \frac{1}{8}c_e(u_i) \tag{21}$$

The weighting scheme is empirically determined to give more emphasis to product specific spamming than group specific spamming. Deviations are generally weaker evidences and thus given the smallest weightage.

Baseline. We finally introduce a baseline method which ranks the reviewers by their unhelpfulness. Each review is assigned helpfulness votes of 1 (unhelpful) and 0 (helpful) by other users. We then derive the unhelpfulness score of a reviewer by averaging the helpfulness vote values received for his reviews. A unhelpfulness score of 1 thus implies that the reviewer receives only unhelpful votes. On the other hand, reviewers with unhelpfulness score of 0 receive helpful votes only.

6.2 Evaluation Methodology

There are several challenges in conducting the user evaluation experiments. Firstly, there are many reviewers and it is impossible for our human evaluators to judge everyone (see Table 1). Secondly, evaluating a reviewer alone can already be very time consuming especially when the reviewer has contributed many reviews. The number of reviews per user can be as large as 349 (235 after pre-processing) for our `MProducts` dataset. It will take human evaluators too much time to use the existing web interface of Amazon.com, which was not designed for spammer evaluation, to examine these reviews and the reviewed products. The existing Amazon's web interface is only good for users who want to search and contribute product reviews. Finally, to discourage sloppiness in user evaluation, we need to train our human evaluators and track their work ensuring that they provide high quality evaluation. This will subsequently give us trustworthy evaluation outcome that can be subsequently analysed and extracted to derive the ground truth labels.

We handle the first issue by choosing a relatively small subset of reviewers to be evaluated. These comprise both reviewers highly suspicious of spamming by each methods as well as those un-suspicious ones. Human evaluation of

a small sample is not new as it has been widely used in information retrieval (IR) task evaluation, e.g., evaluating the relevance of search results for only a small sample of search queries instead of all queries. Similar to IR evaluation which examines the top ranked results of search engines, we address the second issue by *selecting* a subset of reviews of each selected reviewer for the human evaluators to examine while still providing access to other non-highlighted reviews [8]. The purpose here is to identify some suspicious reviews, i.e., those contributing to spamming behaviors, as the highlighted subset that should be interesting for human evaluators to begin their evaluation. Should other reviews become relevant to the evaluation, the evaluators should still be able to view and compare them with other reviews.

Review Spammer Evaluation Software

In addition to selecting small numbers of reviewers and reviews, we also develop a *review spammer evaluation software* that ensures that the human evaluators can easily browse the reviewer profiles and their reviews (both selected and non-selected). The software is configured to mandate the human evaluators to examine all selected reviews (10 per reviewer in our experiments) before making their judgement on the associated reviewer. The human evaluators are also expected to enter textual comments and reasons during their evaluation. This reduces the chance of frivolous evaluation thus addressing the third issue in the evaluation process. Furthermore, the software provides easy visualization of reviews with exact and near-duplicates, review ratings among recent ratings on the same products, multiple reviews on same products, and multiple reviews on the same product groups (not necessary on the same day). These features also help to reduce the amount of evaluation efforts and time.

Figure 3 depicts the user interface of the software when a selected reviewer is examined. The reviewer has 19 reviews written for 17 products implying that he has multiple reviews for the two products (as shown in the right lower panel). The left lower panel shows the 10 selected reviews by the reviewer with the 1st review highlighted for display in the center upper panel. The center lower panel shows two products each with two reviews by the reviewer and one of the products has two near-duplicate reviews annotated with pink stars. The right upper panel shows the rating of the highlighted review along with other ratings on the same product concerned. The right lower panel shows the several brands reviewed by the reviewer, and his duplicate and near-duplicate reviews annotated with red and pink stars respectively.

Experiment Setup

The evaluation setup consists of the following steps:

- For each spammer detection method, we select 10 top ranked reviewers and 10 bottom ranked reviewers. We then merge all the selected spammers into a pool which consists of 62 reviewers. These reviewers are then sorted by their combined spamming behavior scores. 25 top ranked reviewers and 25 bottom ranked reviewers are then selected for user evaluation. This number of reviewers is quite reasonable for a human evaluator to examine. We further randomly order the reviewers so that there is no relationship between reviewers' order and their spammer scores.
- For each reviewer, select 10 of his/her reviews to be highlighted for human evaluator's attention. These selected reviews must be seen by the human evaluator before the latter makes judgement on whether the reviewer is a spammer. The reviews are selected based on their involvement in the spamming behaviors identified. Specifically, we select:
  - Reviews that are: (i) exact (or near-exact) duplicates of other reviews or (ii) have identical (or similar) ratings with other reviews on the same product by the evaluated reviewer (we call these the **TP** reviews);
  - Reviews of some products in product groups with multiple identical high or low ratings from the evaluated reviewer within the same day (**TG** reviews);
  - Reviews by the evaluated reviewer having ratings deviated from the average ratings of the reviewed products (**GD** reviews);
  - Reviews by the evaluated reviewer having ratings deviated from the average ratings of the reviewed products and are the early reviews of the reviewed products (**ED** reviews).

  For reviewers with less than 10 reviews under the above four categories, we randomly select other reviews to cover the shortfall (random reviews).
- Three human evaluators are recruited to examine the selected reviewers and their selected reviews using the review spammer evaluation software. They are tertiary students familiar with searching and reading product reviews at Amazon website. The software records the spammer label decisions of each evaluator and the reviews actually viewed by the evaluator. For each reviewer, the labeled decision is either "spammer" and "non-spammer" and the evaluators are not informed about the number of spammers to be labeled. The evaluators are also told to mark reviews "suspicious" if they find the reviews likely to be used to spam products.
- Based on the labeled decisions, we determine the **Normalized Discounted Cumulative Gain (NDCG)** [8] performance of each method. The **discounted cumulative gain (DCG)** of a ranked list of 50 reviewers $(u_{i_1}, u_{i_2}, \cdots, u_{i_{50}})$ is defined as:

$$DCG = \sum_{p=1}^{50} \frac{2^{f(i_p)} - 1}{log_2(1+p)} \tag{22}$$

  where $f(i_p)$ equals to the number of votes u_{i_p} received ($f(i_p) \in [0, 3]$). NDCG is simply DCG normalized by the DCG of the ideal rank order of the items that has spammers agreed by all 3 evaluators ranked before those spammers agreed by 2 evaluators who are in turn ranked before the remaining reviewers.

$$NDCG = \frac{DCG}{DCG\ of\ ideal\ ranked\ list} \tag{23}$$

  NDCG determines how well a method's rank order of reviewers follows the ideal rank order giving more weightage (determined by $log_2(1+p)$ in DCG) to the

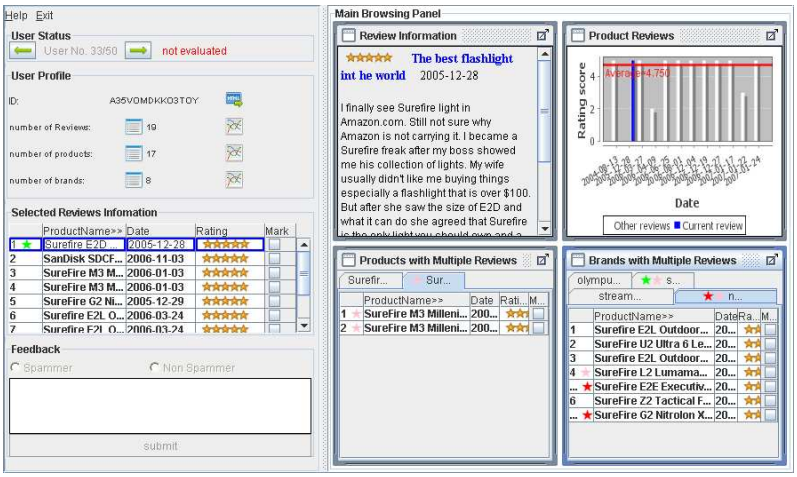


Figure 3: Screenshot of Review Spammer Evaluation Software.

Table 2: Evaluation Results

	Evaluator 1	Evaluator 2	Evaluator 3
# Spammers			
Evaluator 1	**26**	18	20
Evaluator 2	-	**23**	18
Evaluator 3	-	-	**23**
# Non-Spammers			
Evaluator 1	**24**	19	21
Evaluator 2	-	**27**	22
Evaluator 3	-	-	**27**

reviewers with higher importance (determined by $f(i_p)$ in DCG).

6.3 Results

Inter-evaluator consistency. We first show the results of evaluation by our three evaluators. Table 2 shows the number of spammers and non-spammers labeled by the evaluators in the diagonal cells (with boldface font type), and the number of overlapping spammers between each pair of evaluators. As shown in the table, the evaluators are largely consistent in their judgements of spammers and non-spammers. All the three evaluators agrees on 16 spammers and 18 non-spammers, constituting 78% of 50 evaluated reviewers. As the Cohen's kappa values of our evaluator pairs are between 0.48 and 0.64, we have moderate to substantial inter-evaluator agreement[3][14].

Spammer Ranking Performance. Given the results of the three evaluators, we assign a final label to each reviewer using majority voting. A reviewer is assigned a final label if the label is agreed by two or more evaluators. We ended up having 24 reviewers labeled as "spammers" (16 reviewers have 3 votes each and 8 reviewers have 2 votes each) and 26 labeled as "non-spammers" (8 reviewers have 1 vote each and 18 reviewers have no vote). The number of top 10 reviewers with the final spammer labels, and the number of bottom 10 reviewers with non-spammer labels for different methods are shown in Table 3. The results show that the ALL and **TP** methods correctly rank the spammers and non-spammers at the top and bottom ranks respectively. They are significantly better than the Baseline method based on helpful votes. This is followed by **TG**. **GD** and **ED** are slightly worse than Baseline.

[3]The kappa κ values can be interpreted as: no agreement (for $\kappa < 0$), slight agreement (for $\kappa \in (0, 0.2]$), fair agreement (for $\kappa \in (0.2, 0.4]$), moderate agreement (for $\kappa \in (0.4, 0.6]$), substantial agreement (for $\kappa \in (0.6, 0.8]$), and almost perfect agreement (for $\kappa \in (0.8, 1.0]$).

Table 3: Results of Top 10 and Bottom 10 Ranked Reviewers

	Baseline	**TP**	**TG**	**GD**	**ED**	ALL
# spammers in top 10	7	10	10	6	6	10
# non-spammers in bottom 10	7	10	9	6	7	10

Next, we examine the NDCG for different top k positions (k= 1 to 50) in the rank list produced by each method. As shown in Figure 4, ALL, **TP** and **TG** produce very good rank orders, and thus very high NDCG scores across the different k's. They are better than the **GD**, **ED** and Baseline methods. For small k values, **GD** and **ED** are able to rank better than Baseline hence giving the former two methods an edge over Baseline.

In summary, the experimental results show that our method is effective as indicated by the inter-evaluator agreement in labeling spammers and non-spammers, as well as by the higher NDCG values for the proposed strategies as compared to the baseline. In particular, the **TP** and **TG** strategies are very strong indicators of spam. In contrast, the baseline approach of using unhelpfulness votes is not a good indicator of spam.

7. SUPERVISED SPAMMER DETECTION AND ANALYSIS OF SPAMMED OBJECTS

7.1 Regression Model for Spammers

With the labeled spammers, we now train a linear regression model to predict the number of spam votes of a given reviewer's spamming behaviors, i.e., **GD**, **ED**, **TP**, **TG** scores. To ensure that the trained regression model make

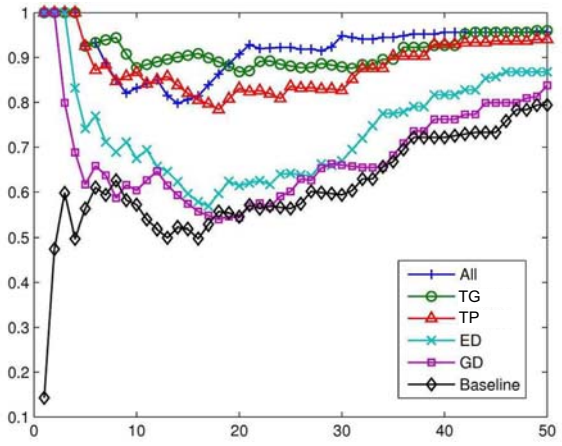


Figure 4: NDCG Results

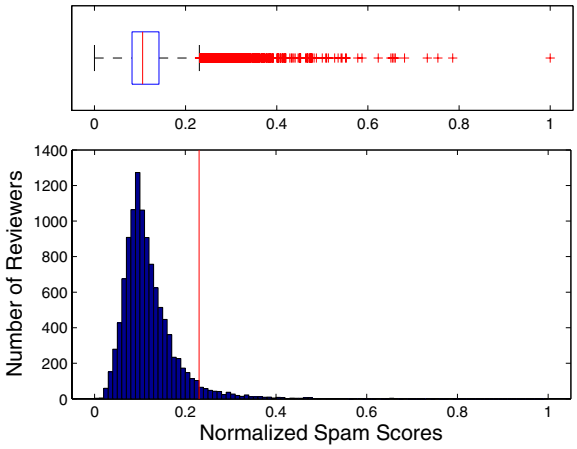


Figure 5: Distribution of Reviewers' New Spam Scores after Normalization

as little prediction errors as possible at the highly ranked reviewers, we consider the number of spam votes in the objective function for optimizing the regression model.

The regression model trained on the 50 labeled reviewers is shown below:

$$\text{Number of spam votes}(u_i) = w_0 + w_1 \cdot c_d(u_i) + w_2 \cdot c_e(u_i) + w_3 \cdot c_p(u_i) + w_4 \cdot c_g(u_i)$$

The regression model learnt by minimizing mean square error has these w_i values: $w_0 = 0.37$, $w_1 = -0.42$, $w_2 = 1.23$, $w_3 = 2.86$, $w_4 = 4.2$. It is interesting to note that the learnt weights are quite consistent with those used in Section 6.1. The only surprise is that general deviation **GD** is now given a negative weight. This suggests that having larger general deviation does not make a reviewer looks more like a spammer, although early deviation does.

We then apply the regression model on all 11,038 reviewers assigning each of them a new spam score normalized by the maximum score value and denote it by $s(u_i)$'s. The distribution of normalized spam scores is shown in Figure 5. Most reviewers have relatively small spam scores. There are 513 (4.65%) reviewers having spam scores greater than or equal to the upper-whisker of the distribution (0.23).

7.2 Analysis of Spammed Products and Product Groups

To show how the products and product groups are affected by spammers, we define a **spam Index** for a product o_j and a product group g_l as:

$$s(o_j) = Avg_{u_i \in \mathbf{U}_s(o_j)} s(u_i) \quad (24)$$

$$s(g_l) = Avg_{u_i \in \mathbf{U}_s(g_l)} s(u_i) \quad (25)$$

where $\mathbf{U}_s(o_j)$ denotes the set of reviewers of o_j each having spam scores computed by our trained regression model. $\mathbf{U}_s(g_l)$ denotes the set of reviewers of objects of product group g_l.

For comparison, two baseline indices are also defined for products and product groups. Instead of using the reviewer's spam score $s(u_i)$ like Equations 24 and 25:

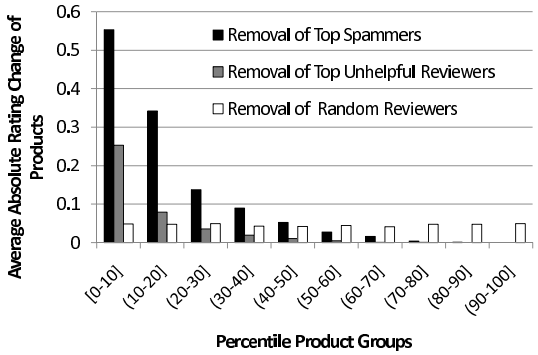
- **Unhelpfulness score**: Average of unhelpfulness scores of reviewers of the product or product groups.
- **Random Index**: Average of random score from 0 to 1 assigned to reviewers of the product or product groups.

One way to study the impact of spammers is to compare the average ratings of a product or a product group when spammers are included versus when they are excluded. Figure 6 shows how the average rating of a product changes after removing the top 4.65% users with highest spam scores. This is the fraction of users with spam scores greater than or equal to the upper-whisker of the distribution in Figure 5). Products are ranked from largest to smallest $s(o_j)$ values. The figure shows that the rating changes are more significant for the top 10 and 20 percentiles of products for the removed spammers or reviewers with highest unhelpful ratio indexes. The removed reviewers due to random index, on the other hand, leave similar rating changes on the products across the different percentile products. Figures 7 shows a similar figure for product brands, with similar observations.

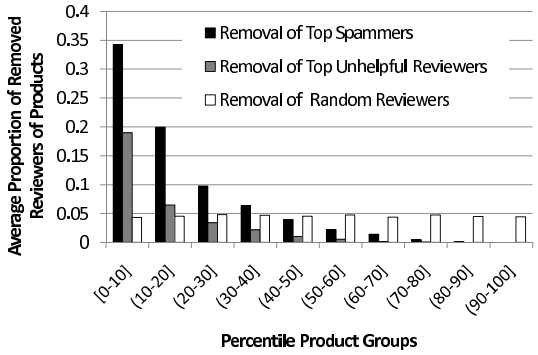
To see why the changes in ratings are more significant at higher percentiles, we plot the average proportion of reviewers removed from the products (Figure 6b) and product brands (Figure 7b) as a result of removing the top spammers. Both figures show that most of the reviewers removed by spam scores and unhelpful ratio index belong to the highly ranked products and brands. This explains more rating changes for these products and brands.

8. CONCLUSIONS

This paper proposes a behavioral approach to detect review spammers who try to manipulate review ratings on some target products or product groups. We derive an aggregated behavior scoring methods to rank reviewers according to the degree they demonstrate spamming behaviors. To evaluate our proposed methods, we conduct user evaluation on an Amazon dataset containing reviews of manufactured products. We found that our proposed methods generally outperform the baseline method based on helpfulness votes. We further learn a regression model from the user-labeled ground truth spammers, and apply the learnt model to score reviewers. It is shown that by removing reviewers with very high spam scores, the highly spammed products and prod-

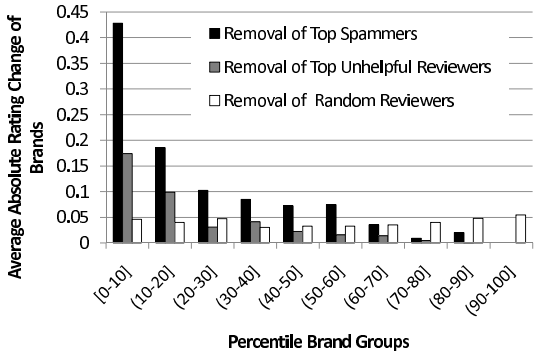


(a) Average Absolute Rating Change

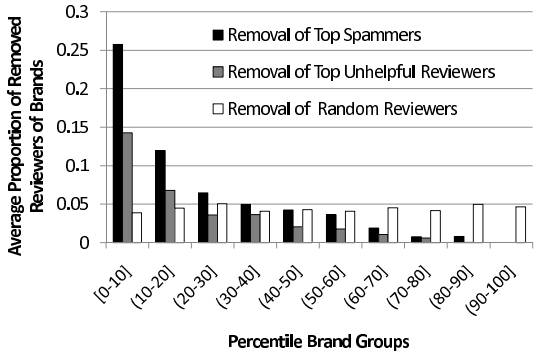


(b) Average Proportion of Removed Reviewers

Figure 6: Products after removing high spam score reviewers



(a) Average Absolute Rating Change



(b) Average Proportion of Removed Reviewers

Figure 7: Brands after removing high spam score reviewers

uct groups according to our approach will experience more significant changes in aggregate rating and reviewer count compared with removing randomly scored or unhelpful reviewers. As part of our future work, we can incorporate review spammer detection into review detection and vice versa. Exploring ways to learn behavior patterns related to spamming so as to improve the accuracy of the current regression model is also an interesting research direction.

9. ACKNOWLEDGMENTS

This work is supported by Singapore's National Research Foundation's research grant, NRF2008IDM-IDM004-036.

10. REFERENCES

[1] L. Becchetti, C. Castillo, D. Donato, R. Baeza-Yates, and S. Leonardi. Link analysis for web spam detection. *ACM TWeb*, 2(1), 2008.

[2] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, and M. Gonçalves. Detecting spammers and content promoters in online video social networks. In *SIGIR*, 2009.

[3] P.-A. Chirita, J. Diederich, and W. Nejdl. Mailrank: Using ranking for spam detection. In *CIKM*, 2005.

[4] 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 *WWW*, 2009.

[5] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *ACM EC*, 2000.

[6] E. Gilbert and K. Karahalios. Understanding deja reviewers. In *CSCW*, 2010.

[7] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD*, 2004.

[8] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR*, 2000.

[9] N. Jindal and B. Liu. Review spam detection. In *WWW (poster)*, 2007.

[10] N. Jindal and B. Liu. Opinion spam and analysis. In *WSDM*, 2008.

[11] N. Jindal, B. Liu, and E.-P. Lim. Finding unusual review patterns using unexpected rules. In *CIKM*, 2010.

[12] S.-M. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti. Automatically assessing review helpfulness. In *EMNLP*, 2006.

[13] E. Knorr and R. Ng. A unified notion of outliers: Properties and computation. In *KDD*.

[14] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1).

[15] J. Liu, Y. Cao, C.-Y. Lin, Y. Huang, and M. Zhou. Low-quality product review detection in opinion summarization. In *EMNLP-CoNLL*, 2007.

[16] A. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *EMNLP*, 2005.

[17] G. Wu, D. Greene, B. Smyth, and P. Cunningham. Distortion as a validation criterion in the identification of suspicious reviews. Technical Report UCD-CSI-2010-04, University College Dublin, 2010.