Spotting Opinion Spammers using Behavioral Footprints

Arjun Mukherjee[†], Abhinav Kumar[†], Bing Liu[†], Junhui Wang[†], Meichun Hsu[‡], Malu Castellanos[‡], Riddhiman Ghosh[‡]

[†]University of Illinois at Chicago, [‡]HP Labs

arjun4787@gmail.com, {akumar34, liub, junhui}@uic.edu, {meichun.hsu, malu.castellanos, riddhiman.ghosh}@hp.com

ABSTRACT

Opinionated social media such as product reviews are now widely used by individuals and organizations for their decision making. However, due to the reason of profit or fame, people try to game the system by opinion spamming (e.g., writing fake reviews) to promote or to demote some target products. In recent years, fake review detection has attracted significant attention from both the business and research communities. However, due to the difficulty of human labeling needed for supervised learning and evaluation, the problem remains to be highly challenging. This work proposes a novel angle to the problem by modeling *spamicity* as latent. An unsupervised model, called Author Spamicity Model (ASM), is proposed. It works in the Bayesian setting, which facilitates modeling spamicity of authors as latent and allows us to exploit various observed behavioral footprints of reviewers. The intuition is that opinion spammers have different behavioral distributions than non-spammers. This creates a distributional divergence between the latent population distributions of two clusters: spammers and non-spammers. Model inference results in learning the population distributions of the two clusters. Several extensions of ASM are also considered leveraging from different priors. Experiments on a real-life Amazon review dataset demonstrate the effectiveness of the proposed models which significantly outperform the state-of-the-art competitors.

Categories and Subject Descriptors

H.1.2 [**Information Systems**]: Human Factors; J.4 [**Computer Applications**]: Social and Behavioral Sciences

General Terms

Algorithms, Experimentation, Theory

Keywords

Abuse, Opinion Spam, Deceptive and Fake Reviewer Detection

1. INTRODUCTION

Online reviews of products and services are used extensively by consumers and businesses to make critical purchase, product design, and customer service decisions. However, due to the financial incentives associated with positive reviews, imposters try to game the system by posting fake reviews and giving unfair ratings to promote or demote target products and services. Such individuals are called *opinion spammers* and their activities are called *opinion spamming* [14]. The problem of opinion spam is widespread, and many high-profile cases have been reported in the news [40]. In fact the menace has become so serious that Yelp.com has launched a "sting" operation to publicly shame businesses who buy fake reviews [39].

In recent years, researchers have also studied the problem and proposed several techniques. However, the problem is still wide open. Unlike many other forms of spamming, the key difficulty for solving the opinion spam problem is that it is hard to find gold-standard data of fake and non-fake reviews for model building because it is very difficult, if not impossible, to manually recognize/label fake/non-fake reviews by mere reading [14, 34].

Since it was first studied in [14], various methods have been proposed to detect opinion spam. One of the main methods is supervised learning [10, 14, 34]. However, due to the lack of reliable ground truth label of fake/non-fake review data, existing works have relied mostly on ad-hoc or pseudo fake/non-fake labels for model building. In [14], duplicate and near duplicate reviews were assumed to be fake reviews, which is restrictive and can be unreliable. In [25], a manually labeled dataset was used, which also has reliability issues because it has been shown that the accuracy of human labeling of fake reviews is very poor [34]. In [34], Amazon Mechanical Turk (AMT) was employed to crowdsource fake hotel reviews by paying (US$1 per review) anonymous online workers (called *Turkers*) to write fake reviews for some hotels. Although these reviews are fake, they do not reflect the dynamics of fake reviews in a commercial website [31] as the Turkers do not have the same psychological state of mind when they write fake reviews as that of fake reviewers in a commercial website who have real business interests to promote or to demote. Also, Turkers may not have sufficient domain knowledge or experience to write convincing fake reviews. Due to the lack of labeled data, unsupervised methods have also been proposed for detecting individual [26, 42] and group [29, 30] spammers, time-series [46] and distributional [9] analysis, and mining reviewing patterns as unexpected association rules [15] and reviewing burstiness [8].

In a wide field, a study of bias and controversy of research paper reviews was reported in [24]. However, research paper reviews do not usually involve faking. Web spam [3, 38, 45], email spam [5], blog spam [22], clickbots [19], auction fraud [35], social networks [13], etc. have also been widely investigated. However, the dynamics of these forms of spamming are quite different from those of opinion spamming in reviews.

The above existing works in opinion spam have made good progresses. However, they are largely based on heuristics and/or hinge on ad-hoc fake/non-fake labels for model building. No principled or theoretical models have been proposed so far.

This paper proposes a novel and principled technique to model and to detect opinion spamming in a Bayesian framework. It transcends the existing limitations discussed above and presents an unsupervised method for detecting opinion spam. We take a fully Bayesian approach and formulate opinion spam detection as a clustering problem. The Bayesian setting allows us to model *spamicity* of reviewers as latent with other observed behavioral

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

features in our Author Spamicity Model (ASM). Spamicity here means the degree of being spamming. The key motivation hinges on the hypothesis that opinion spammers differ from others on behavioral dimensions [29]. This creates a separation margin between population distributions of two naturally occurring clusters: spammers and non-spammers. Inference in ASM results in learning the distributions of two clusters (or classes) based on a set of behavioral features. Various extensions of ASM are also proposed exploiting different priors.

In summary, this paper makes the following contributions:

1. It proposes a novel and principled method to exploit observed behavioral footprints to detect spammers (fake reviewers) in an unsupervised Bayesian framework precluding the need of any manual labels for learning which is both hard [14] and noisy [34]. A key advantage of employing Bayesian inference is that the model facilities characterization of various spamming activities using estimated latent variables and the posterior. It facilitates both detection and analysis in a single framework rendering a deep insight into the opinion spam problem. This cannot be done using existing methods. To our knowledge, this is the first principled model for solving this problem.
2. It proposes a novel technique to evaluate the results without using any labeled data. This method uses reviews of the top ranked and bottom ranked authors produced by the model as two classes of data to build a supervised classifier. The key idea is that the classification uses a complete different set of features than those used in modeling. Thus, if this classifier can classify accurately, it gives a good confidence that the unsupervised spamicity model is effective (details in §3.3).
3. It conducts a comprehensive set of experiments to evaluate the proposed model based on the classification evaluation method above and also human expert judgment. It also compares with a set of strong baseline techniques. The results show that the proposed models outperform the baselines significantly.

2. MODEL

This section details the proposed unsupervised model. We first discuss the basic intuition (§2.1) and the observed features (§2.2). In §2.3, we explain the generative process of our model and detail inference methods in §2.4 and §2.5.

2.1 Model Overview

As discussed above, the proposed model formulates spam detection as an unsupervised clustering problem in the Bayesian setting. It belongs to the class of generative models for clustering [6] based on a set of observed features. It models spamicity, s_a (degree/tendency of spamming in the range [0, 1]) of an author, a; and spam/non-spam label, π_r of a review, r as latent variables. π_r is essentially the *class* variable reflecting the cluster memberships (we have two clusters, K = 2, spam and non-spam) for every review instance. Each author/reviewer (and respectively each review) has a set of observed features (behavioral clues) emitted according to the corresponding latent prior class distributions. Model inference learns the latent population distributions of the two clusters across various behavioral dimensions, and also the cluster assignments of reviews in the unsupervised setting based on the principle of probabilistic model-based clustering [37].

As the generative process of ASM conditions review spam labels on author spamicites, inference also results in author spamicity estimates (probability of spamming) facilitating ranking of authors based on spamicity which is our main focus.

2.2 Observed Features

Here we propose some characteristics of abnormal behaviors which are likely to be linked with spamming and thus can be exploited as observed features in our model for learning the spam and non-spam clusters. We first list the author features and then the review features. The notations are listed in Table 1.

Variable/Functions	Description
a; A; r; $r_a = (a, r)$	Author a; set of all authors; a review; review r by author a
R_a, $p(r_a)$	All reviews by a, $R_a = \{r_a\}$; associated product p for r_a
R_p; $R_{a,p}$	Reviews on product, p; Reviews on product p by author a
$\star\,(r_a, p(r_a))$	The $\star$ rating of r_a on product $p(r_a)$ on the 5-$\star$ rating scale
$MaxRev(a)$	Maximum # of reviews posted in a day by an author, a
$F(a)$; $L(a)$	First posting date of a; last posting date of a
$L(a,p)$; $A(p)$	Last review posting date by a on p; product p's launch date
$k \in \{\hat{s}, \hat{n}\}$	Class variable k for Spam/Non-spam class (label)
$s_a \sim Beta(\alpha_{\hat{s}}, \alpha_{\hat{n}})$	Spamicity of an author, a, , $s_a \in [0, 1]$
$\alpha^a_{k\in\{\hat{s},\hat{n}\}}$	Beta shape parameters (priors) for s_a for each author a
$\pi_{r_a} \sim Bern(s_a)$	Spam/Non-spam class label for review r_a, $\pi_{r_a} \in \{\hat{s}, \hat{n}\}$
$\theta^f_{k\in\{\hat{s},\hat{n}\}} \sim Beta(\gamma^f_{\hat{s}}, \gamma^f_{\hat{n}})$	Per class prob. of exhibiting the review behavior, $f_{5\ldots9}$
$\gamma^f_{k\in\{\hat{s},\hat{n}\}}$	Beta shape parameters of θ^f for each review behavior, f
$\psi^f_{k\in\{\hat{s},\hat{n}\}} \sim Beta$	Per class probability of exhibiting the author behavior, $f_{1\ldots4}$
$\psi^f_{k,1}$; $\psi^f_{k,2}$	Beta shape parameters of class k for behavior f
$x^f_{a,r} \sim Bern(\theta^f_{k\in\{\hat{s},\hat{n}\}})$	Observed review feature, $f \in \{DUP, EXT, DEV, ETF, RA\}$
$y^f_{a,r} \sim \psi^f_{k\in\{\hat{s},\hat{n}\}}$	Observed author features $f \in \{CS, MNR, BST, RFR\}$
$n_{a,\hat{s}}$; $n_{a,\hat{n}}$	# of reviews of author a assigned to spam; non-spam class
$n^f_{k,P}$; $n^f_{k,A}$	# of reviews in class $k \in \{\hat{s}, \hat{n}\}$ which have review feature f (P)resent (f attains value 1); (A)bsent (f attains value 0)
n_a; $n_{k\in\{\hat{s},\hat{n}\}}$	# of reviews by author a; # of reviews in class k
K	Total number of clusters in the model

Table 1: List of notations

Author Features: Author features have values in [0, 1]. A value close to 1 (respectively 0) indicates spamming (non-spamming).

1. Content Similarity (CS): As crafting a new review every time is time consuming, spammers are likely to copy reviews across similar products. It is thus useful to capture the content similarity of reviews (using cosine similarity) of the same author. We chose the maximum similarity to capture the worst spamming behavior.

$$f_{CS}(a) = f_1(a) = \max_{r_i, r_j \in R_a, i<j} cosine(r_i, r_j) \quad (1)$$

2. Maximum Number of Reviews (MNR): Posting many reviews in a single day also indicates an abnormal behavior. This feature computes the maximum number of reviews in a day for an author and normalizes it by the maximum value for our data.

$$f_{MNR}(a) = f_2(a) = \frac{MaxRev(a)}{\max_{a\in A}(MaxRev(a))} \quad (2)$$

3. Reviewing Burstiness (BST): The study in [29] reports that opinion spammers are usually not longtime members of a site. Genuine reviewers, however, use their accounts from time to time to post reviews. It is thus useful to exploit the activity freshness of an account in detecting spamming. We define reviewing burstiness using the activity window (difference of first and last review posting dates). If reviews are posted over a reasonably long timeframe, it probably indicates normal activity. However, when all reviews are posted within a very short burst (τ = 28 days, estimated in §3.1), it is likely to be a spam infliction.

$$f_{BST}(a) = f_3(a) = \begin{cases} 0, L(a) - F(a) > \tau \\ 1 - \frac{L(a)-F(a)}{\tau}, otherwise \end{cases} \quad (3)$$

4. Ratio of First Reviews (RFR): Spamming early can impact the initial sales as people rely on the early reviews. Hence, spammers would try to be among the first reviewers for products as this enables them to control the sentiment [26]. We compute the ratio of *first reviews* to total reviews for each author. First reviews refer to reviews where the author is the first reviewer for the products.

$$f_{RFR}(a) = f_4(a) = \frac{|\{r \in R_a : r \; is \; a \; first \; review\}|}{|R_a|} \quad (4)$$

Review Features: We have 5 binary review features. Values of 1

indicate spamming while 0 non-spamming.

5. Duplicate/Near Duplicate Reviews (*DUP*): Spammers often post multiple reviews which are duplicate/near-duplicate versions of previous reviews on the same product to boost ratings [26]. To capture this phenomenon, we compute the review feature (f_{DUP}) *duplicate reviews* on the same product as follows:

$$f_{DUP}(r_a) = f_5(a,r) = \begin{cases} 1, \exists r \in R_{p=p(r_a)}\ cosine(r_a, r) > \beta_1 \\ 0, otherwise \end{cases} \quad (5)$$

f_{DUP} attains a value of 1 for a review r_a by an author a on product p, if it is similar (using cosine similarity based on some threshold, β_1= 0.7 (say)) to any other review on p. β_1 is estimated in §3.1.

6. Extreme Rating (*EXT*): On a 5-star ($\star$) rating scale, it reflects the intuition that to inflict spam, spammers are likely to give extreme ratings (1$\star$ or 5$\star$) in order to demote/promote products.

$$f_{EXT}(r_a) = f_6(a,r) = \begin{cases} 1, \star(r_a, p(r_a)) \in \{1,5\} \\ 0, \star(r_a, p(r_a)) \in \{2,3,4\} \end{cases} \quad (6)$$

7. Rating Deviation (*DEV*): Review spamming usually involves wrong projection either in the positive or negative light so as to alter the true sentiment on products. This hints that ratings of spammers often deviate from the average ratings given by other reviewers. This feature attains the value of 1 if the rating deviation of a review exceeds some threshold β_2. β_2 is estimated in §3.1. We normalize by the maximum deviation, 4 on a 5-star scale.

$$f_{DEV}(r_a) = f_7(a,r) = \begin{cases} 1, \frac{\left|\star(r_a,\ p(r_a)) - E\left[\star(r_{a' \neq a},\ p(r_a))\right]\right|}{4} > \beta_2 \\ 0, otherwise \end{cases} \quad (7)$$

The expectation is taken over all reviews on product $p = p(r_a)$ by other authors, $a' \neq a$, to get the average rating on p.

8. Early Time Frame (*ETF*): [26] argues that spammers often review early to inflict spam as the early reviews can greatly impact people's sentiment on a product. To capture this spamming characteristic, we propose the following feature:

$$f_{ETF}(r_a) = f_8(a,r) = \begin{cases} 1, ETF(r_a, p(r_a)) > \beta_3 \\ 0, otherwise \end{cases}$$
$$ETF(r_a, p) = \begin{cases} 0, L(a,p) - A(p) > \delta \\ 1 - \frac{L(a,p) - A(p)}{\delta}, otherwise \end{cases} \quad (8)$$

$ETF(r_a, p)$ captures how early an author a reviewed the product p. δ = 7 months is a threshold for denoting earliness (estimated in §3.1). The definition says that if the latest review is beyond 7 months of product launch, it is no longer considered to be early. At the other extreme, if reviews are posted just after launch this feature attains a value of 1. β_3 is the corresponding threshold indicating spamming and is estimated in §3.1.

9. Rating Abuse (*RA*): This feature captures the abuse caused by multiple ratings on the same product. Multiple ratings/reviews on the same product are unusual. Although this feature is similar to DUP, it focuses on the rating dimension rather than content. Rating abuse, $RA(a,p)$ is defined by the similarity of ratings of an author, a towards a product, p across multiple reviews by the author weighted by the total number of reviews on the product.

$$f_{RA}(r_a) = f_9(a,r) = \begin{cases} 1, RA(a, p(r_a)) > \beta_4 \\ 0, otherwise \end{cases};$$
$$RA(a,p) = |R_{a,p}| \left(1 - \frac{1}{4}\left(\max_{r \in R_{a,p}}(\star(r,p)) - \min_{r \in R_{a,p}}(\star(r,p))\right)\right) \quad (9)$$

The similarity of multiple star rating is computed using the difference between maximum and minimum star rating on a 5-star scale to capture consistent high/low ratings. The normalization constant is 4 as it is the maximum possible rating difference. For multiple ratings in genuine cases where ratings change (e.g., after correct use), the feature attains lower values. β_4 is the rating abuse threshold indicating spamming and is estimated in §3.1.

2.3 Process

In ASM, spam detection is influenced by review and author features. Normalized continuous author features in [0, 1] are modeled as following a Beta distribution ($y^f_{a,r} \sim \psi^f_{k \in \{\hat{s},\hat{n}\}}$) (Table 1). This enables ASM to capture more fine grained dependencies of author's behaviors with spamming. However, review features being more objective, we found that they are better captured when modeled as binary variables being emitted from a Bernoulli distribution ($x^f_{a,r} \sim Bern\left(\theta^f_{k \in \{\hat{s},\hat{n}\}}\right)$ (Table 1). $\theta^f_{k \in \{\hat{s},\hat{n}\}}$ for each review feature $f \in \{DUP, EXT, DEV, ETF, RA\}$ and $\psi^f_{k \in \{\hat{s},\hat{n}\}}$ for each author feature $f \in \{CS, MNR, BST, RFR\}$ denote the per class/cluster (spam vs. non-spam) probability of emitting feature f.

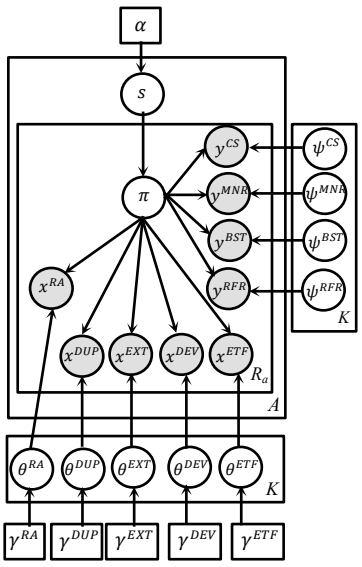

Figure 1: Plate Notation of ASM.

Latent variables s_a and π_r denote the spamicity of an author, a and the (spam/non-spam) class of each review, r. The objective of ASM is to learn the latent behavior distributions for spam and non-spam clusters (K = 2) along with spamicites of authors from the observed features. We now detail its generative process.

1. For each class/cluster, $k \in \{\hat{s}, \hat{n}\}$:
   Draw $\theta_k^{f \in \{DUP,\ldots,RA\}} \sim Beta(\gamma^f)$
2. For each (author), $a \in \{1 \ldots A\}$:
   i. Draw spamicity, $s_a \sim Beta(\alpha^a)$;
   ii. For each review, $r_a \in \{1 \ldots R_a\}$:
      a. Draw its class, $\pi_{r_a} \sim Bern(s_a)$
      b. Emit review features $f \in \{DUP, \ldots, RA\}$:
         $x^f_{r_a} \sim Bern\left(\theta^f_{\pi_{r_a}}\right)$;
      c. Emit author features $f \in \{CS, .., RFR\}$:
         $y^f_{r_a} \sim \psi^f_{\pi_{r_a}}$;

We note that the observed author features are placed in the review plate (Figure 1). This is because each author behavior can be thought of as percolating through reviews of that author and emitted across each review to some extent. Doing this renders two key advantages: i) It permits us to exploit a larger co-occurrence domain. ii) It paves the way for a simpler sampling distribution providing for faster inference.

2.4 Inference

We employ approximate posterior inference with Monte Carlo Gibbs sampling, and use Rao-Blackwellization [2] to reduce sampling variance by collapsing on the latent variables s and θ^f. As observed author features obtaining values in [0, 1] are modeled as continuous Beta distributions, sparsity is considerably reduced as far as parameter estimation of $\psi^{f \in \{CS,\ldots RFR\}}$ is concerned. Hence, to ensure speed, we estimate ψ^f_k using the method of moments,

once per sweep of Gibbs sampling. The Gibbs sampler is given by:

$$p(\pi_i = k|\boldsymbol{\pi}_{\neg i} \dots) \propto \frac{n_{a,k_{\neg i}}+\alpha_k^a}{(n_a+\alpha_{\hat{s}}^a+\alpha_{\hat{n}}^a)_{\neg i}} \times \prod_{f\in\{DUP,EXT,DEV,ETF,RA\}} \left(g\left(f,k,x_{a,r}^f\right)\right) \times \prod_{f\in\{CS,MNR,BST,RFR\}} \left(p\left(y_{a,r}^f|\psi_{\pi_i}^f\right)\right) \quad (10)$$

where functions g and $p\left(y_{a,r}^f|\psi_{\pi_i}^f\right)$ are defined below:

$$g\left(f,k,x_{a,r}^f\right) = \begin{cases} \frac{\left(n_{k,P}^f+\gamma_k^f\right)_{\neg i}}{\left(n_k+\gamma_{\hat{s}}^f+\gamma_{\hat{n}}^f\right)_{\neg i}}, & if \quad x_{a,r}^f = 1 \\ \frac{\left(n_{k,A}^f+\gamma_{\neg k}^f\right)_{\neg i}}{\left(n_k+\gamma_{\hat{s}}^f+\gamma_{\hat{n}}^f\right)_{\neg i}}, & if \quad x_{a,r}^f = 0 \end{cases}$$

$$p\left(y_{a,r}^f|\psi_{\pi_i}^f\right) \propto \left(y_{a,r}^f\right)^{\psi_{i_1}^f-1}\left(1-y_{a,r}^f\right)^{\psi_{i_2}^f-1}$$

The subscript $\neg i$ denotes counts excluding review $i = r_a = (a,r)$. The Beta shape parameter updates for ψ_k^f using method of moments are given as follows:

$$\psi_k^f = (\psi_{k,1}^f, \psi_{k,2}^f) = \left(\mu_k^f\left(\frac{\mu_k^f(1-\mu_k^f)}{\sigma_k^f}-1\right), (1-\mu_k^f)\left(\frac{\mu_k^f(1-\mu_k^f)}{\sigma_k^f}-1\right)\right) \quad (11)$$

where μ_k^f and σ_k^f denote the mean and biased sample variance for feature f corresponding to class k. Algorithm 1 details the full inference procedure for learning ASM using the above Gibbs conditional. Omission of a latter index denoted by [ ] corresponds to the row vector spanning over the latter index.

2.5 Hyperparameter EM

Algorithm 1 performs inference using uninformed priors (i.e., hyperparameters α and γ are set to $(1,1)$). Posterior estimates of spamicity can be improved if hyperparameters α and γ are estimated from the data. This is because the priors for author spamicity and latent review behaviors (α and γ) directly affect spam/non-spam cluster assignment to reviews. Algorithm 2 details hyperparameter estimation using the single sample Monte Carlo EM, which is recommended by [4] as it is both computationally efficient and often outperforms multiple-sample Monte Carlo EM.

Algorithm 2 learns hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ which maximize the model's complete log-likelihood, $\mathcal{L}$. We employ an L-BFGS optimizer [48] for maximization. L-BFGS is a quasi-Newton method which does not require the Hessian matrix of second order derivatives. It approximates the Hessian using rank-one updates of first order gradient. A careful observation of the model's complete log-likelihood reveals that it is a separable function in $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ allowing the hyperparameters to be maximized independently. Due to space limits, we only provide the final update equations:

$$\alpha_k^a = \text{argmax}_{\alpha_k^a} \begin{pmatrix} \log\Gamma(\alpha_{\hat{s}}^a+\alpha_{\hat{n}}^a) + \log\Gamma\left(\alpha_{\hat{s}}^a+n_{a,\hat{s}}\right) + \log\Gamma\left(\alpha_{\hat{n}}^a+n_{a,\hat{n}}\right) \\ -\log\Gamma(\alpha_{\hat{s}}^a) - \log\Gamma(\alpha_{\hat{n}}^a) - \log\Gamma(n_a+\alpha_{\hat{s}}^a+\alpha_{\hat{n}}^a) \end{pmatrix} \quad (12)$$

$$\frac{\partial\mathcal{L}}{\partial\alpha_k^a} = \Psi(\alpha_{\hat{s}}^a+\alpha_{\hat{n}}^a) + \Psi\left(\alpha_k^a+n_{a,k}\right) - \Psi(\alpha_k^a) - \Psi(n_a+\alpha_{\hat{s}}^a+\alpha_{\hat{n}}^a) \quad (13)$$

$$\gamma_k^f = \text{argmax}_{\gamma_k^f} \begin{pmatrix} \log\Gamma\left(\gamma_{\hat{s}}^f+\gamma_{\hat{n}}^f\right) + \log\Gamma\left(\gamma_{\hat{s}}^f+n_{k,P}^f\right) + \log\Gamma\left(\gamma_{\hat{n}}^f+n_{k,A}^f\right) \\ -\log\Gamma\left(\gamma_{\hat{s}}^f\right) - \log\Gamma\left(\gamma_{\hat{n}}^f\right) - \log\Gamma\left(n_k+\gamma_{\hat{s}}^f+\gamma_{\hat{n}}^f\right) \end{pmatrix} \quad (14)$$

$$\frac{\partial\mathcal{L}}{\partial\gamma_{\hat{s}}^f} = \Psi\left(\gamma_{\hat{s}}^f+\gamma_{\hat{n}}^f\right) + \Psi\left(\gamma_{\hat{s}}^f+n_{\hat{s},P}^f\right) - \Psi\left(\gamma_{\hat{s}}^f\right) - \Psi\left(n_k+\gamma_{\hat{s}}^f+\gamma_{\hat{n}}^f\right) \quad (15)$$

$$\frac{\partial\mathcal{L}}{\partial\gamma_{\hat{n}}^f} = \Psi\left(\gamma_{\hat{s}}^f+\gamma_{\hat{n}}^f\right) + \Psi\left(\gamma_{\hat{n}}^f+n_{\hat{n},A}^f\right) - \Psi\left(\gamma_{\hat{n}}^f\right) - \Psi\left(n_k+\gamma_{\hat{s}}^f+\gamma_{\hat{n}}^f\right) \quad (16)$$

where, $\Psi(\cdot)$ denotes the digamma function.

3. EXPERIMENTAL EVALUATION

We now evaluate the proposed ASM model. We use the reviews of manufactured products from Amazon.com. We consider only authors/reviewers with at least 3 reviews as authors with fewer reviews have few behavior characteristics. For reviewers with fewer reviews, the method in [46] can be applied. Our final data comprises of 50,704 reviewers, 985,765 reviews, and 112,055 products. Below we describe parameter estimation, baseline systems, and evaluation results.

Algorithm 1 Inference using MCMC Gibbs Sampling

1. Initialization:

Randomly assign review clusters, $\pi_{r_a} = \begin{cases} \hat{n}, z < 0.5 \\ \hat{s}, z \geq 0.5 \end{cases}; z \sim U(0,1)$

2. Iterate $\boldsymbol{n = 1}$ to $\boldsymbol{N_{max}}$: // $N_{max} = 3000$

For author, $a = 1$ to A:

For review $r_a = 1$ to R_a:

i. Flush cluster assignment, π_{r_a};

ii. Sample $\pi_{r_a} \sim p(\pi = k| \dots)$ using (10);

iii. Update $n_{k,[\,]}^{f=DUP}$, $n_{k,[\,]}^{f=EXT}$, $n_{k,[\,]}^{f=DEV}$, $n_{k,[\,]}^{f=ETF}$, $n_{k,[\,]}^{f=RA}$ for $k \in \{\hat{s}, \hat{n}\}$

End for

End for

If $n > N_{Burn\_In}$: // $N_{BurnIn} = 250$

For author, $a = 1$ to A:

For review $r_a = 1$ to R_a:

Update $\psi_k^{f=CS}$, $\psi_k^{f=MNR}$, $\psi_k^{f=BST}$, $\psi_k^{f=RFR}$; $k \in \{\hat{s}, \hat{n}\}$ using (11)

End for

End for

End if

Algorithm 2 Single-sample Monte Carlo EM

1. Initialization:

Start with uninformed priors: $\alpha^a \leftarrow (1,1)$; $\gamma^f \leftarrow (1,1)$

2. Repeat:

i. Run Gibbs sampling to steady state (Algorithm 1) using current values of α^a, γ^f.

ii. Optimize α^a using (12) and γ^f using (14)

Until convergence of α^a, γ^f

3.1 Learning Observed Feature Thresholds

As noted in §2.2, the proposed feature constructions contain thresholds. The thresholds can either be set heuristically or learned using some weak supervision. In this work, we use weak supervision to learn the thresholds from the Amazon group spam dataset in [29], which provides a small set of labeled spammer groups and their reviews. Using this small set of available data is not a limitation of our method because the actual spamicty modeling of ASM still remains unsupervised as it does not use any spamicty labels for authors/reviews in model building. In fact, ASM does not have any response variable (e.g., [28, 36]) where supervision can be fed using labels. To keep evaluation fair, the labeled data in [29] is not used for our evaluation in §3.3.

Thresholds of burstiness, $\tau = 28$ days and earliness, $\delta = 7$ months (in §2.2) were estimated using greedy hill-climbing search maximizing the likelihood of the data in [29]. It is also important to note that above thresholds apply to feature constructions in §2.2 and not specific to ASM. As we will see in §3.2, the same features are used by other baseline methods, so improvements of ASM are attributed to its process and not the choice of feature thresholds.

Thresholds for binary discretization of continuous review features $\beta_{1\dots4}$ (in §2.2) are learned using Recursive Minimal Entropy Partitioning (RMEP) [7]. The estimated values are as follows: $\beta_1 = 0.72$, $\beta_2 = 0.63$, $\beta_3 = 0.69$, $\beta_4 = 2.01$.

3.2 Systems in our Experiments

Although ASM clusters reviews and estimates author spamicities, in this work, our focus is to evaluate the ranking of authors based on estimated author spamicities.

3.2.1 Generative Models: ASM Variants

ASM with Uninformed Priors (ASM-UP): This is the fully unsupervised version of ASM. Uninformed priors are used for Beta distributed variables, s^a, θ^f, $f \in \{DUP, .., RA\}$, i.e., $\forall a \in A$, $\alpha^a \leftarrow (1,1)$; $\gamma^f \leftarrow (1,1)$. This setting is called uninformed

because any value in [0, 1] is equally likely to be assigned to the Beta distributed variables and the model doesn't benefit from any domain knowledge based priors. Posterior estimates are drawn after 3000 iterations with an initial burn-in of 250 iterations.

ASM with Informed Priors (ASM-IP): Here we employ guidance using some domain knowledge heuristics. Amazon tags each review with AVP (Amazon Verified Purchase) if the reviewer actually bought the product. Keeping other settings the same as ASM-UP, for each author, a, if none of his reviews have AVP tags, then we set $s_a \sim Beta(5,1)$, else we set $s_a \sim Beta(1,5)$[1]. The rationale is that authors who have not bought a single product on Amazon are likely to be less reliable (hence probably spamming) than those who also review products that they have purchased on Amazon (i.e., receiving AVP tags).

ASM with Hyperparameter Estimation (ASM-HE): This setting estimates the hyperparameters α^a, $a \in A$ and γ^f, $f \in \{DUP, EXT, DEV, ETF, RA\}$ using Algorithm 2 keeping all other settings fixed as ASM-UP.

3.2.2 Unsupervised Rank Aggregation

ASM estimates reviewer spamicities as scores in [0, 1] (the posterior on $s \sim Beta$). We can also regard the observed behaviors as ranking functions in [0, 1] with extreme values 0 (respectively 1) indicating non-spamming (spamming) on various behavior dimensions. Then, estimating the final spamicites of authors becomes unsupervised rank learning using aggregation [21]. The problem setting is described as follows.

Let $x \in X$ denote an item (e.g., author) in the instance space X (e.g., set of authors/reviewers) that need to be ranked relative to each other according to some criterion (e.g., spamicity). Let $q \in Q$ denote a query and $r : Q \times X \rightarrow \mathbb{R}$ denote a ranking function whose output governs the rank position of an item, i.e., $r(q,x) > r(q,x')$ specifies that x is ranked higher than x' on query q using ranking function r. The notation $r_u \succ_\epsilon r_v$ signifies that the ranking function r_u is better than r_v with respect to a certain evaluation criterion (e.g., NDCG, MAP, L_2 loss, etc.). Given a set of ranking functions, $\{r_i\}_{i=1}^N$, rank aggregation learns the optimal ranking function, r_{Opt} (using a weighted combination of $\{r_i\}_{i=1}^N$) such that $\forall i, r_{Opt} \succ_\epsilon r_i$. This problem setting is also called *unsupervised rank fusion* or *ensemble ranking* [44]. In the supervised setting, one usually employs regression to learn the weights [41]. In the unsupervised setting, the approach is two-fold: i) Derive an *incidental*/*surrogate* supervision signal. ii) Employ a learning algorithm to learn the parameters of r_{Opt}. In [21], the *surrogate* supervision signal was computed using some ranker agreement heuristics and iterative gradient descent was used as the learning algorithm.

In our case of ranking reviewers according to spamicites, we have 9 ranking functions $\{r_i\}_{i=1}^{N=9}$ corresponding to the 9 behavior feature dimensions (in §2.2). For each author feature, $f_{1\ldots4} \in \{CS, MNR, BST, RFR\}$, we directly use the value of the feature as the ranking function while for each review feature, $f_{5\ldots9} \in \{DUP, EXT, DEV, ETF, RA\}$, we take the expected value of the feature across all reviews of an author to compute the corresponding author feature. We then directly use the value of each author feature as a ranking function to produce 9 training ranks. Given no other knowledge, this is a reasonable approach since $f_{1\ldots9}$ being anomalous reviewing behaviors are likely to be correlated with spammers (the nature of correlation will be detailed using posterior density analysis in §4). Thus, the training ranking produced by each feature function is based on a certain spamicity dimension. However, none of the training rankings may be optimal. We employ learning to rank [27] to learn an optimal ranking function by aggregating 9 ranking functions $\{f_i\}_{i=1}^{N=9}$.

Learning to rank [27] is a supervised technique that takes a set of rankings for training and produces a single optimal aggregated ranking function. Each of our training rankings is produced by sorting instances (authors) based on values of a feature in $\{f_i\}_{i=1}^{N=9}$. Each instance/author in a training ranking is represented as a vector of $f_{1\ldots9}$ spam features. We experimented with two popular learning to rank algorithms: SVMRank [17] and RankBoost [47]. For SVMRank, we used the system in [17]. RankBoost was from RankLib[2]. We use the pair-wise L_2 loss metric as the optimization criterion for RankBoost and SVMRank. We also experimented with RankNet, AdaRank, Coordinate Ascent in RankLib, but their results were poorer and hence not included.

3.2.3 Feature Value Sum and Helpfulness

For a more comprehensive comparison, we also experiment with the following baseline approaches:

Feature Sum (FSum): As each feature $f_{1\ldots9}$ measures spam behavior on a specific dimension, an obvious approach is to rank the authors in descending order of the sum of all feature values.

Helpfulness Score (HS): In many review sites (e.g., Amazon), readers can provide helpfulness feedback to each review. It is reasonable to assume that spam reviews should get less helpfulness feedback. HS uses the mean helpfulness score (percentage of people who found a review helpful) of reviews of each reviewer to rank reviewers in ascending order of the scores.

Finally, we note that although in ASM spam detection is modeled as clustering (§2.3), our key task is to estimate author spamcities (the posterior on the latent variable, $s_a \sim Beta \in [0,1]$) for ranking authors according to their spamicities.. Standard clustering algorithms (e.g., *k*-means) are not suitable baselines because they only produce clusters, but do not generate a ranking of authors.

3.3 Evaluation Results

As noted in §1, we are not aware of any gold-standard ground truth labeled data for opinion spammers. Hence, this work focuses on Bayesian inference in the unsupervised setting. To evaluate the author spamicities computed by different systems, we use two methods: *review classification* and *human evaluation*.

Running the systems in §3.2 on our data generates a ranking of 50,704 reviewers. However, human evaluation on all authors is clearly impossible. Even for review classification, the number of reviews is huge for such a large number of reviewers. We thus sample the rank positions 1, 10, 20,…, 50000 (with a sampling lag/interval of 10 ranks) to construct the evaluation set, E of 5000 rank positions. Using a fixed sampling lag ensures that performance on E is a good approximation over the entire range. 5000 is reasonable for review classification, but for human evaluation we need to use a subset (see below).

3.3.1 Review Classification

This is a new evaluation method in the unsupervised setting. The idea is that if ASM is effective, it must rank the highly likely spammers at the top and highly likely non-spammers at the bottom. We use a supervised classification of likely spammers and likely non-spammers to evaluate this ranking. Instead of classifying reviewers, we classify their reviews.

Prior works in psycholinguistic research (e.g. [32]) have shown that faking or lying usually involves more use of personal

[1] A potential non-spammer (NS), $s^{NS} \sim Beta(1,5)$ has an expected value, $E[s^{NS}] = \frac{1}{6}$ which is much smaller than a potential spammer (S), $s^{NS} \sim Beta(5,1)$, with expected value, $E[s^S] = \frac{5}{6}$ in the range [0, 1].

[2] http://www.cs.umass.edu/~vdang/ranklib.html

Table 2 (a)

k	ASM-UP				ASM-IP				ASM-HE				SVMRank			
(%)	P	R	F1	A	P	R	F1	A	P	R	F1	A	P	R	F1	A
5	77.7	74.0	75.8	75.5	77.9	74.8	76.3	75.7	79.6	75.1	77.3	77.4	72.1	74.7	73.4	73.1
10	68.5	62.9	65.6	63.5	72.1	69.5	70.8	72.8	76.8	70.3	73.4	73.4	67.9	70.3	69.1	70.4
15	62.9	59.9	61.4	60.2	66.8	64.5	65.6	66.1	68.9	67.4	68.1	66.7	57.2	60.9	58.9	59.2

Table 2 (b)

k	RankBoost				FSum				HS			
(%)	P	R	F1	A	P	R	F1	A	P	R	F1	A
5	74.6	75.1	74.8	74.6	76.1	73.6	74.8	75.2	57.8	61.7	59.7	59.8
10	68.1	71.6	69.8	71.2	67.3	60.2	63.6	61.4	58.9	60.8	59.8	60.6
15	58.3	57.8	58.0	59.8	60.2	55.3	57.6	57.2	61.7	58.0	59.8	58.2

Table 2 (a, b): 5-fold SVM CV for review classification using top *k* (%) authors' reviews as the spam (+) class and bottom *k* % authors' reviews as the non-spam (-) class. P: Precision, R:Recall, F1:F1-Score, A:Accuray.

	ASM-UP			ASM-IP			ASM-HE			SVMRank			RankBoost			FSum			HS		
	B_1	B_2	B_3	B_1	B_2	B_3	B_1	B_2	B_3	B_1	B_2	B_3	B_1	B_2	B_3	B_1	B_2	B_3	B_1	B_2	B_3
J_1	31	15	3	36	11	1	43	5	0	36	19	1	37	13	1	34	13	0	6	14	17
J_2	28	14	3	31	6	1	36	6	0	32	16	4	34	8	2	32	11	0	5	12	14
J_3	29	13	2	33	8	0	39	3	0	33	11	2	34	11	0	31	8	0	8	9	10
Avg.	29.3	14.0	2.67	33.3	8.33	0.67	39.3	4.67	0	33.7	15.3	2.33	35.0	10.7	1	32.3	10.7	0	6.33	11.7	13.7
κ_{Fleiss}	0.73			0.68			0.74			0.71			0.72			0.76			0.73		

Table 3: Number of spammers detected in each bucket (B_1, B_2, B_3) by each judge (J_1, J_2, J_3) across each method. Last row reports the agreement of judges using Fleiss' multi-rater kappa (κ_{Fleiss}) for each method.

pronouns and associated verb actions to justify fake statements eventually resulting in more use of positive sentiments and emotion words. The hypothesis has been attested in [34] where text classification using *n*-gram features have been shown quite effective in detecting spam and non-spam reviews. Thus, if our classification of reviews based on text features is good, it implies that the ASM ranking of reviewers according to spamicites is effective because text classification concurs with the abnormal behavior spam detection of ASM. The key characteristic of this classification is that the text features have not been used in ASM. Clearly, using the same set of features will not be meaningful.

For this set of experiments, we consider the reviews from the top *k*% ranked authors to be in the spam (+) class while reviews from the bottom *k*% of the ranked authors to be in the non-spam (–) class. Although the actual percentage of spammers is unknown, deception prevalence studies [33, 43] have reported 8-15% spam rate in online review sites. We thus report results for *k* = 5%, 10%, and 15%. We induce a linear kernel SVM[3] using SVM^{Light} [16] and report 5-fold cross validation results in Table 2. The features are 1-3-grams. We note the following observations:

1. As *k* increases, we find a monotonic degradation in review classification performance (except HS) which is expected as the spamicities of top and bottom authors get closer which makes the corresponding review classification harder.
2. For *k* = 5%, ASM models performs best on F1 and Accuracy metrics. Next in order are FSum, RankBoost, and SVMRank. It is interesting that the simple baseline FSum performs quite well for *k* = 5%. The reason is attributed to the fact that the top positions are mostly populated by heavy spammers while the bottom positions are populated by genuine reviewers and hence a naïve un-weighted FSum could capture this phenomenon.
3. For *k* = 10, 15%, FSum does not perform so well (SVMRank and RankBoost outperform Fsum). This is because for *k* = 10, 15%, the ranked positions involve more difficult cases of authors/reviewers and a mere sum is not able to balance the feature weights as not all features are equally discriminating.
4. For *k* = 10%, SVMRank and RankBoost outperform ASM-UP and perform close to ASM-IP. ASM-HE still outperforms SVMRank and RankBoost by 4% in F1 and 2-3 % in accuracy.
5. For *k* = 15%, ASM variants outperform other methods and increase F1 by a margin of 2-10% and accuracy by 3-7%.
6. Performance of HS remains much poorer and similar for each *k* showing that it is not able to rank spammers well, indicating that helpfulness is not a good metric for spam detection. In fact, helpfulness votes are subject to abuse.

We note that no significance test is applied here to compare performance of different methods because each method uses *different* reviews in classification as the output rankings from the systems are different. However, we experimented with multiple and different randomized binning in 5-fold CV which showed the same trend as in Table 2.

To further analyze the nature of spam and non-spam reviews by spammers and non-spammers (based on the rankings of ASM), we do an additional experiment. We consider the top 15% authors ranked by ASM-HE and randomly split them into two sets of authors and construct two classes of reviews coming from each set of authors. Classification on this set of reviews (coming from two sets of authors belonging to spammers) yielded 57% accuracy. Similarly, classification of reviews by two sets of authors belonging to non-spammers (bottom 15% ranked authors of ASM-HE) yielded 58% accuracy. We tried different random selection of authors which also yielded similar accuracies. We note that the accuracy is greater than 50% but less than the classification accuracy using ASM-HE (Table 2). This shows that the above experiment is likely to separate authors and not spam vs. non-spam reviews. As different authors have differences in writing, the accuracy is greater than 50% (random guessing). But it is lower than spam and non-spam review classification because reviews of both classes come from spammers (or non-spammers) and hence noisy. Our result of author classification is supported by prior studies in [20] which showed that authorship attribution using n-grams is not very effective. It also indirectly shows that a lower accuracy obtained by various competitors in Table 2 have more noise in ranking authors based on spamicity.

These results render confidence in the proposed models. It also indirectly shows that there are some linguistic differences between spam and non-spam reviews.

3.3.2 Human Evaluation

Our second evaluation is based on human expert judgment, which is commonly used in research on spam, e.g., Web [3, 38], email [5], and even blogs and social spam [22]. Human evaluation has also been used for opinion spam in prior works [42, 46]. It is however important to note that just by reading a single review without any context, it is very hard to determine whether a review is fake (spam) or not [34]. However, it has been shown in [29] that when sufficient context is provided e.g., reviewing patterns, ratings, type/brand of products reviewed, posting activity trails, etc., human expert evaluation becomes easier.

For this work, we used 3 domain expert judges: employees at Rediff Shopping; eBay.in for evaluating our ranked reviewers based on spamicites. The judges had sufficient background in reviews of products and sellers due to their nature of their work in online shopping. The judges were briefed with many opinion spam signals: i) Having zero caveats, and full of empty adjectives. ii) Purely glowing praises with no downsides. iii) Suspicious brand affinity/aversion, posting trails, etc., from prior findings and

[3] Other kernels, e.g., polynomial, rbf, sigmoid didn't perform so well. Linear kernel has been shown effective for text classification by many researchers, e.g., [18].

consumer sites [1, 12]. These signals are sensible as they have been compiled by consumer domain experts with extensive know-how on fake reviews. Our judges were also familiar with Amazon reviews and given access to additional metadata, e.g., reviewing profile, demographic information, etc. Although the judges were not provided the proposed features, they were encouraged to use their own signals along with the above existing signals and reviewer metadata. It is important here to note that providing various signals compiled from prior works and domain experts in consumer sites [1, 12] to the judges does not introduce a bias but enhances judgment. Without any signals, as mentioned above, it is very difficult to judge by merely reading reviews. It is also hard for anyone to know a large number of signals without extensive experience in opinion spam detection. Given a reviewer and his reviews, the judges were asked to independently examine his entire profile and to provide a label as spammer or non-spammer.

Due to the large number (5000) of reviewers in the evaluation set, E, it would have taken too much time for human judges to assess all the reviewers in E. We selected the following three ranked buckets (B_1, B_2, B_3) for evaluation by our judges:

B_1 (Top 50): Reviewers ranked from 1 to 50 by each system in E.
B_2 (Middle 50): Reviewers ranked from 2501 to 2550.
B_3 (Bottom 50): Reviewers ranked at bottom 50 ranks in E.

This is reasonable because these rank-positions in E reflect the performance trend on the entire range. Table 3 reports the results of each judge (as the count of reviewers labeled as spammers) for each bucket across each method. Additionally, we report the agreement of judges using Fleiss multi-rater kappa [11] for each method in the last row of Table 3. We note the following observations:

1. All 3 judges perform similarly with slight variation (e.g., J_1 seems to be stricter and identifies more spammers in each bucket than J_3). This shows that the judges have consensus in spam judgments. Further, the kappa values being in the range of substantial agreement according to scale[4] in [23] bolsters confidence in the judgments.
2. For all methods except HS, we find the average number of spammers detected by judges decreasing monotonically for buckets B_1, B_2, and B_3 which is expected from the ranking produced by the algorithms. HS performs poorly (placing more spammers in B2, B3 which is undesireable) as also observed in §3.3.1. Thus, helpfulness votes aren't useful for spam detection.
3. For B_1, ASM-HE performs best. On average, it is able to place 39 spammers in the top 50 rank positions. ASM-IP, RankBoost, and SVMRank perform poorer than ASM-HE. ASM-UP does not perform well as uninformed priors are weaker.
4. For B_2, the performance order is ASM-HE → ASM-IP → RankBoost → FSum → ASM-UP → SVMRank. At this bucket, a good method should place fewer spammers as B_2 is in the middle. This is indeed the case in Table 3. Note that an ideal ranking should not place any spammers in the middle range as it is very unlikely that 50% of the reviewers are spammers.
5. For the last bucket, B_3, we find ASM-HE and FSum performing best by not placing any spammers in the bottom ranks. ASM-IP does quite well too. Next in performance order are RankBoost → SVMRank → ASM-UP. FSum performed very well in B_3 because its ranking (based on descending order of feature sum value) placed authors who ranked very low in all features. As the features are all abnormal and indicate suspicious behaviors, reviewers attaining very low FSum values are likely to be genuine reviewers which explain the good results of FSum.

In summary, we can conclude that proposed ASM-HE is effective and outperforms other methods and baselines. ASM-IP and ASM-UP are slightly inferior which is reasonable as they use weaker priors. Learning to rank methods SVMRank and RankBoost using proposed behaviors are strong competitors showing that the proposed behavior features are effective.

4. POSTERIOR ANALYSIS

Apart from generating a ranking of authors based on spamicity (s_a), ASM also estimates the latent distributions of spam and non-spam reviews and authors corresponding to each observed author and review feature dimension f as reflected in the spam vs. non-spam clusters or classes, i.e., $\theta^f_{k\in\{\hat{s},\hat{n}\}}$, $\psi^f_{k\in\{\hat{s},\hat{n}\}}$ for $K = 2$ clusters, $k \in \{\hat{s}, \hat{n}\}$ (Table 1). It is thus interesting to analyze the posterior on the learned distributions $\theta^f_{k\in\{\hat{s},\hat{n}\}}$, $\psi^f_{k\in\{\hat{s},\hat{n}\}}$ for each feature dimension f. Although these are model's estimates of spam and non-spam distributions for each anomalous/spamming dimension or feature (§2.2), it is still worthwhile to study the estimates to see whether they reflect our understanding of spamming behaviors.

We report the posterior on the latent spam and non-spam distributions under each feature f ($\theta^f_{k\in\{\hat{s},\hat{n}\}}$, $\psi^f_{k\in\{\hat{s},\hat{n}\}}$) estimated by ASM-HE using MC-EM (Algorithm 2). Due to space constraints, we only focus on ASM-HE as it performed best. We plot the estimated densities of $\theta^f \sim Beta$ and $\psi^f \sim Beta$ for spam and non-spam clusters in Figure 2. For each estimated latent distribution ($\theta^f_{k\in\{\hat{s},\hat{n}\}}$, $\psi^f_{k\in\{\hat{s},\hat{n}\}}$), we show the density of spam in red (solid) lines and non-spam in blue (dotted) lines. Furthermore, the spam and non-spam densities being different are shown in their respective scales (secondary y-axes, left: blue/dotted for non-spam and right: solid/red for spam). The x-axis is the domain of the Beta distribution, [0, 1] which is also the range of the corresponding observed (author and review) feature where values close to 0 and 1 show extremes of non-spamming and spamming (author and review). Also shown are the expected values for each latent behavior for spam (red/dashed lines) and non-spam (blue/dash-dotted lines) in respective scales and figure captions. We compare and explain the estimated trends of each behavior feature (see §2.2) across spam and non-spam classes below.

Content Similarity (ψ^{CS}): From the densities in Figure 2 (a), we see that most spammers have a lot of content similarity in their reviews while non-spammers exhibit much less review similarity. The expected value of content similarity (using cosine similarity) for non-spammers is 0.09, much lower than 0.7 for spammers.

Maximum Number of Reviews (ψ^{MNR}): Figure 2 (b) shows the difference of densities for this behavior. The density curve for non-spammers attaining the peak towards the left of the plot has majority of the probability mass concentrated towards the left. This shows that non-spammers attain much lower values for MNR. Spammers on the other hand have their masses concentrated at the middle with the density curve attaining a peak at a feature value of 0.34. We recall that this feature, MNR is normalized by the largest value for this feature (maximum number of reviews in a day) in our data which happens to be 21. This imples that a significant percentage of spammers have written close to $0.34 \times 21 \approx 7$ reviews in a single day. In expectation, we see that the maximum number of reviews in a day for non-spammers is 0.11, i.e., $0.11 \times 21 \approx 2$ reviews while $0.28 \times 21 \approx 5$ reviews for spammers.

Reviewing Burstiness (ψ^{BST}): Reviewing busrtiness measures the account activity (according to difference between last and first post date) within the estimated threshold $\tau = 28$ days. At one extreme, reviewers with overall posting activity more than 28 days obtain a value of 0 while at other extreme reviewers whose last and first post are very close attain value close to 1. From

[4] No agreement ($\kappa < 0$), slight agreement ($0 < \kappa \leq 0.2$), fair agreement ($0.2 < \kappa \leq 0.4$), moderate agreement ($0.4 < \kappa \leq 0.6$), substantial agreement ($0.6 < \kappa \leq 0.8$), and almost perfect agreement for $0.8 < \kappa \leq 1.0$.

Figure 2 (c), we find that density of non-spam class attains a peak at the feature value 0, showing a large percentage of non-spammers post reviews spanned over more than 28 days. This is intuitive as genuine reviewers use their accounts from time to time to post reviews. For spammers, the density attains a peak at feature value 0.89 (i.e., 28×(1-0.89) ≈ 3 days showing that a considerable percentage of spmamers never use their accounts after 3 days from the first post date. Further the expected account acitvity period for spammers is 0.75, i.e., 28 × (1-0.75) ≈ 7 days. The expected value for non-spammers is 0.01 which is negligible showing most non-spammers have account activities (by posting reviews) for more than 28 days.

Ratio of First Reviews (ψ^{RFR}): As defined earlier, this behavior computes the ratio of first reviews to total reviews per account (reviewer). First review means that a review was posted first for a given product. From Figure 2 (d), we see that unlike previous behaviors, the separation between spammers and non-spammers is not so disticnt. Although in expectation, the estimated behavior attains a value of 0.36 for spammers which is higher than 0.29 for non-spammers, the density profile for non-spammers remains somewhat high. A plausible reason for this phenomenon is that there are many enthusiastic genuine reviewers who review newly launched products. Spammers obtain higher values in expectation as reviewing first/early can give them an edge as early reviews have more impact [26].

Duplicate/Near Dulpicate Reviews on Products (θ^{DUP}): This review feature measures whether a review posted on a product is similar to other reviews on that product. This behavior can detect some sockpuppets where a person posts similar reviews on a product with multiple ids. There can also be multiple similar review posts by a single id. In either cases, this behvarior would return high values for a review when it is a duplicate/near dulpicate to other reviews on that product. From Figure 2 (e), we see stark difference of densities of spam and non-spam reviews based on this behavior. Spam reviews attain higher values (with density peak at extreme right) while non-spam reviews attain very low values (with peak density at extreme left).

Extreme Rating (θ^{EXT}): Extreme review rating is a boolean review feature. Its corresponding latent behavior, θ^{EXT} measures the extent to which reviews are rated with extreme rating (1 or 5 star rating) on the scale [0, 1]. Form Figure 2 (f), we see that the density of spam reviews is concentrated towards the extreme right. Its expected value is 0.86 which implies that about 86% percent of spam reviews are rated with either 1 or 5 stars. For non-spam reviews, we find a somewhat evenly distributed density profile because genuine reviewers usually have different rating levels. The expected value for non-spam reviews is 0.36, i.e., about 36% of non-spam reviews are extreme while the rest 64% is distributed across 2, 3, and 4 star ratings according to the density.

Rating Deviation (θ^{DEV}): Recall that rating deviation for a review is the extent to which its rating deviates from the average rating (general rating consensus) on the product. Its corresponding normalized latent behavior is θ^{DEV}. From the densities in Figure 2 (g), we find that spam reviews deviate a great deal in rating from the general rating consensus. For non-spam reviews we find the density peak is attained at the feature value of 0.12 showing that a considerable percentage of non-spam reviews don't deviate very much in ratings from the average rating. The expected rating deviation is 0.68 for spam reviews which is almost double of the expected deviation of 0.34 for non-spam reviews.

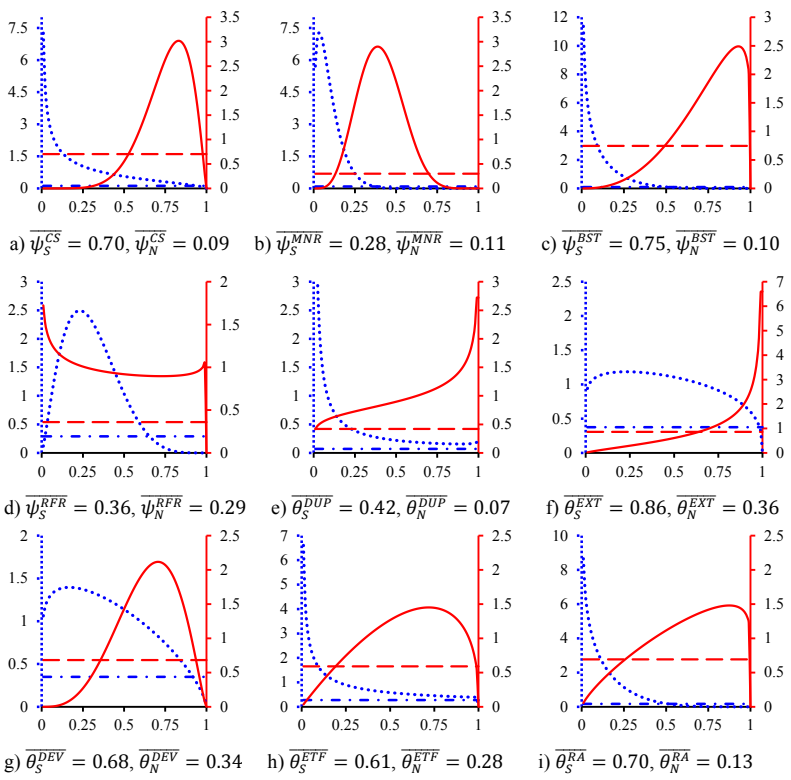

Figure 2: Density function (PDF) of estimated latent behvaior variables, $\psi^f \sim Beta$, $\theta^f \sim Beta$ corresponding to each author and review behavior feature, $f \in \{CS, MNR, BST, RFR\} \cup \{DUP, EXT, DEV, ETF, RA\}$. Estimated posterior densities for spam (in red/solid) and non-spam (in blue/dotted) are plotted with their respective scales (left:blue/dotted for non-spam and right: solid/red for spam). Also shown are the expected values for each latent behavior for spam (red/dashed) and non-spam (blue/dash-dot) in respective scales. Expected values are also reported in plot captions.

Early Time Frame (θ^{ETF}): ETF behavior measures how early was a review posted in reference to the product's launch date. We find something very interesting in its corresponding latent bevarior, θ^{ETF}. We recall from §3.1 that the estimated threshold for this feature is δ = 7 months. So, 0 values indicate that reviews were posted after 7 months of launch date and values close to 1 indicate review posts close to the lauch date. From the density plot in Figure 2 (h), we find that the mass for non-spam cluster is concentrated close to 0 with the expected value of 0.28 showing that relatively fewer (28%) of non-spam reviews for a product are posted early within 7 months of launch. This is possible as not all genuine reviews for a product are posted early within 7 months of launch. In fact, the bulk of the genuine reviews get accumulated overtime. We note that these statistics are in expectation and not for any single product. The 28% of genuine early reviews can be attributed to reviewing enthusiasts with keen interests in the products. For spam reviews, we find that the density tends to lean towards the right, but not extremely concentrated towards the right. It attains an expected value of 0.61 showing that about 61% spam reviews are posted within 7 months of product lauch date while the rest spam reviews accumulate later in the timeline. This dovetails with the findings in [26] that spammers usually post reviews early enough to cast a bigger impact.

Rating Abuse (θ^{RA}): Rating abuse measures the extent products are spammed with multiple ratings. The densities for its corresponding latent behavior, θ^{ETF} are shown in Figure 2 (i). We find that a reasonably large percentage of spam reviews (70% in expectation) have been instances of imparting rating abuse (i.e., among the multiple reviews/ratings for the same product by the same reviewer-id). However for non-spam reviews, the expected

value for this behavior is 0.13. Thus, most non-spam reviews (≈ 100-13 = 87%) are not instances of rating abuse (i.e., are single reviews by a reviewer on a product). Again, we note that our model estimates 13% of non-spam reviews being instances of rating abuse (i.e., multiple ratings on a product by a reviewer) which is a small percentage. However, from the non-spam density we can say that the feature values for those reviews must be very low as the density drops to almost 0 beyond the feature value of 0.5. Again, this is understandable as sometimes a reviewer might review a prodcut more than once when the product experience changes from the first review (e.g., a installation fault, not knowing the correct method of using, etc.).

Lastly, we note that although it is interesting to analyze the relative discriminative strength of each feature, it is not directly possible as we do not have ground-truth spamicity labels for reviewers in our data. Instead, the posterior denisity estimates of latent bahviors give us some clues to relative feature strengths.

5. CONCLUSIONS

This paper proposed a novel and principled method to exploit observed reviewing behaviors to detect opinion spammers (fake reviewers) in an unsupervised Bayesian inference framework. To our knowledge, this is the first such attempt. Existing methods are mostly based on heuristics and/or ad-hoc labels for opinion spam detection. The proposed model has its basis in the theoretic foundation of probabilistic model based clustering. The Bayesian framework facilitates characterization of many behavioral phenomena of opinion spammers using the estimated latent population distributions. It also enables detection and posterior density analysis in a single framework. This cannot be done by any of the existing methods. The paper also proposed a novel way to evaluate the results of unsupervised opinion spam models using supervised classification without the need of any manually labeled data. Finally, a comprehensive set of experiments based on the proposed automated classification evaluation and human expert evaluation have been conducted to evaluate the proposed model. The results across both evaluation metrics show that the proposed model is effective and outperforms strong competitors.

6. ACKNOWLEDGEMENTS

This project is supported in part by a grant from HP Labs Innovation Research Program and a grant from National Science Foundation (NSF) under grant no. IIS-1111092.

7. REFERENCES

[1] Popken, B. 2010. 30 Ways You Can Spot Fake Online Reviews. The Consumerist.

[2] Bishop, C.M. 2006. *Pattern Recognition and Machine Learning*. Springer.

[3] Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M. and Vigna, S. 2006. A reference collection for web spam. *SIGIR Forum*. (2006).

[4] Celeux, G., Chaveau, D., & Diebolt, J. 1996. Stochastic versions of the em algorithm: an experimental study in the mixture case. *Journal of Statistical Computation and Simulation*. (1996).

[5] Chirita, P.A., Diederich, J., and Nejdl, W. 2005. MailRank : Using Ranking for Spam Detection. *CIKM* (2005).

[6] Duda, R. O., Hart, P. E., and Stork, D.J. 2001. *Pattern Recognition*. Wiley.

[7] Fayyad, U., & Irani, K. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. *UAI* (1993), 1022–1027.

[8] Fei, G., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M. and Ghosh, R. 2013. Exploiting Burstiness in Reviews for Review Spammer Detection. *ICWSM*. (2013).

[9] Feng, S., Xing, L., Gogar, A. and Choi, Y. 2012. Distributional Footprints of Deceptive Product Reviews. *ICWSM* (2012).

[10] Feng, S., Banerjee R., Choi, Y. 2011. Syntactic Stylometry for Deception Detection. *ACL* (2011).

[11] Fleiss, J. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*. (1971), 378–382.

[12] Frietchen, C. 2009. How to spot fake user reviews. Consumersearch.com.

[13] Ghosh, S., Viswanath, B., Kooti, F., Sharma, N.K., Korlam, G., Benevenuto, F., Ganguly, N. and Gummadi, K.P. 2012. Understanding and combating link farming in the twitter social network. *WWW*. (2012).

[14] Jindal, N. and Liu, B. 2008. Opinion Spam and Analysis. *WSDM* (2008).

[15] Jindal, N., Liu, B. and Lim, E.-P. 2010. Finding Unusual Review Patterns Using Unexpected Rules. *CIKM* (2010).

[16] Joachims, T. 1999. Making large-scale support vector machine learning practical. *Advances in Kernel Methods*. (1999).

[17] Joachims, T. 2002. Optimizing Search Engines Using Clickthrough Data. *KDD* (2002).

[18] Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. *ECML* (1998).

[19] Kang, H., Wang, K., Soukal, D., Behr, F. and Zheng, Z. 2010. Large-scale bot detection for search engines. *WWW* (2010).

[20] Keselj, V., Peng, F., Cercone, N., Thomas, C. 2003. N- Gram-Based Author Profiles for Authorship Attribution. *PACL* (2003), 255–264.

[21] Klementiev, A., Roth, D. and Small, K. 2007. An Unsupervised Learning Algorithm for Rank Aggregation. *ECML* (2007).

[22] Kolari, P., Java, A., Finin, T., Oates, T. and Joshi, A. 2006. Detecting Spam Blogs : A Machine Learning Approach. *AAAI* (2006).

[23] Landis, J. R. and Koch, G.G. 1977. The measurement of observer agreement for categorical data. *Biometrics*. (1977), 159–174.

[24] Lauw, H.W., Lim, E. and Wang, K. 2007. Summarizing Review Scores of " Unequal " Reviewers. *SIAM SDM* (2007), 539–544.

[25] Li, F., Huang, M., Yang, Y. and Zhu, X. 2011. Learning to Identify Review Spam. *IJCAI* (2011), 2488–2493.

[26] Lim, E.-P., Nguyen, V.-A., Jindal, N., Liu, B. and Lauw, H.W. 2010. Detecting product review spammers using rating behaviors. *CIKM* (2010)

[27] Liu, T.Y. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval*. (2009), 225–331.

[28] McAuliffe, D.B. and J. 2007. Supervised Topic Models. *NIPS* (2007).

[29] Mukherjee, A., Liu, B. and Glance, N. 2012. Spotting Fake Reviewer Groups in Consumer Reviews. *WWW* (2012).

[30] Mukherjee, A., Liu, B., Wang, J., Glance, N. and Jindal, N. 2011. Detecting Group Review Spam. *WWW* (2011).

[31] Mukherjee, A., Venkataraman, V., Liu, B. and Glance, N. 2013. What Yelp Fake Review Filter might be Doing? *ICWSM*. (2013).

[32] Newman, M.L., Pennebaker, J.W., Berry, D.S., Richards, J.M. 2003. Lying words: predicting deception from linguistic styles. *Personality and Social Psychology Bulletin*. (2003), 665–675.

[33] Ott, M., Cardie, C. and Hancock, J. 2012. Estimating the prevalence of deception in online review communities. *WWW* (2012).

[34] Ott, M., Choi, Y., Cardie, C. and Hancock, J.T. 2011. Finding Deceptive Opinion Spam by Any Stretch of the Imagination. *ACL* (2011), 309–319.

[35] Pandit, S., Chau, D.H., Wang, S. and Faloutsos, C. 2007. NetProbe : A Fast and Scalable System for Fraud Detection in Online Auction Networks. *WWW*.

[36] Ramage, D., Hall, D., Nallapati, R., & Manning, C.D. 2009. A supervised topic model for credit attribution in multi-labeled corpora. *EMNLP* (2009).

[37] Smyth, P. 1999. Probabilistic Model-Based Clustering of Multivariate and Sequential Data. *AISTATS* (1999).

[38] Spirin, N. and Han, J. 2012. Survey on Web Spam Detection : Principles and Algorithms. *ACM SIGKDD Explorations*. 13, 2 (2012), 50–64.

[39] Streitfeld, D. 2012. Buy Reviews on Yelp, Get Black Mark. (2012).

[40] Streitfeld, D. 2012. Fake Reviews, Real Problem. New York Times.

[41] Vogt, C.C., Cottrell, G.W. 1999. Fusion via a linear combination of scores. *Information Retrieval*. (1999), 151–173.

[42] Wang, G., Xie, S., Liu, B. and Yu, P.S. 2011. Review Graph Based Online Store Review Spammer Detection. *ICDM* (2011), 1242–1247.

[43] Wang, Z. 2010. Anonymity, Social Image, and the Competition for Volunteers: A Case Study of the Online Market for Reviews. *The B.E. Journal of Economic Analysis & Policy*. 10, 1 (Jan. 2010), 1–34.

[44] Wei, F., Li, W., Liu, S. 2010. iRANK: A Rank-Learn-Combine Framework for Unsupervised Ensemble Ranking. *Journal of the American Society for Information Science and Technology*. (2010).

[45] Wu, B., Goel V. & Davison, B.D. 2006. Topical TrustRank: using topicality to combat Web spam. *WWW* (2006).

[46] Xie, S., Wang, G., Lin, S. and Yu, P.S. 2012. Review spam detection via temporal pattern discovery. *KDD*. (2012).

[47] Freund, Y., Iyer, R., Schapire, R. and Singer, Y. 2003. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*. 4 (2003), 933–959.

[48] Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. 1997. L-BFGS-B: Fortran routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*. (1997).