Collective Opinion Spam Detection: Bridging Review Networks and Metadata

Shebuti Rayana
Stony Brook University
Department of Computer Science
srayana@cs.stonybrook.edu

Leman Akoglu
Stony Brook University
Department of Computer Science
leman@cs.stonybrook.edu

ABSTRACT

Online reviews capture the testimonials of "real" people and help shape the decisions of other consumers. Due to the financial gains associated with positive reviews, however, opinion spam has become a widespread problem, with often paid spam reviewers writing fake reviews to unjustly promote or demote certain products or businesses. Existing approaches to opinion spam have successfully but separately utilized linguistic clues of deception, behavioral footprints, or relational ties between agents in a review system.

In this work, we propose a new holistic approach called SpEagle that utilizes clues from all metadata (text, timestamp, rating) as well as relational data (network), and harness them collectively under a *unified* framework to spot suspicious users and reviews, as well as products targeted by spam. Moreover, our method can efficiently and seamlessly integrate semi-supervision, i.e., a (small) set of labels if available, without requiring any training or changes in its underlying algorithm. We demonstrate the effectiveness and scalability of SpEagle on three real-world review datasets from Yelp.com with filtered (spam) and recommended (non-spam) reviews, where it significantly outperforms several baselines and state-of-the-art methods. To the best of our knowledge, this is the largest scale quantitative evaluation performed to date for the opinion spam problem.

1. INTRODUCTION

Online product and business reviews are increasingly valuable sources for consumers to make decisions on what to purchase, where to eat, which care provider to see, etc. They are powerful since they reflect testimonials of "real" people, unlike e.g., advertisements. Financial incentives associated with reviews, however, have created a market of (often paid) users to fabricate *fake reviews* to either unjustly hype (for promotion) or defame (under competition) a product or business, the activities of whom are called *opinion spam* [9].

The problem is surprisingly prevalent; it is estimated that more than 20% of Yelp's reviews are fake [3], with steady growth [16], while one-third of all consumer reviews on the Internet are estimated to be fake [24]. While widespread, it is a hard and mostly open problem. The key challenge is obtaining large ground truth data to learn from, however manual labeling of reviews is extremely difficult by merely reading them, where humans are only slightly better than random [22], unlike e.g., labeling email spam. This renders supervised methods inadmissible to a large extent.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
KDD '15, August 10-13, 2015, Sydney, NSW, Australia.
Copyright 2015 ACM. ISBN 978-1-4503-3664-2/15/08...$15.00
DOI: http://dx.doi.org/10.1145/2783258.2783370

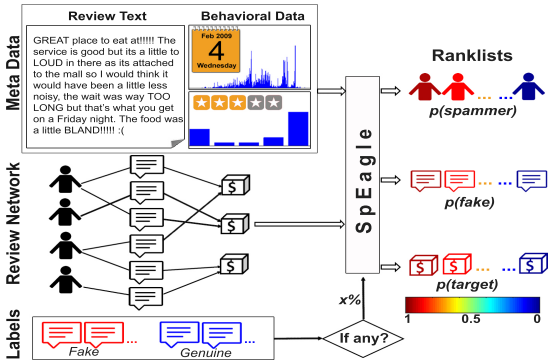


Figure 1: SpEagle collectively utilizes both metadata (review text, timestamp, rating) and the review network (plus available labels, if any) under a *unified* framework to rank all of users, reviews, and products by spamicity.

Since the seminal work of Jindal *et al.* on opinion spam [9], a variety of approaches have been proposed. At a high level, those can be categorized as linguistic approaches [6, 21, 22] that analyze the language patterns of spam vs. benign users for psycholinguistic clues of deception, behavioral approaches [7, 9, 10, 13, 15, 18, 28] that utilize the reviewing behaviors of users (e.g., temporal and distributional footprints), and graph-based methods [1, 5, 14, 26] that leverage the relational ties between users, reviews, and products with minimal to no external information. Current approaches can also be grouped as those that detect fake reviews [6, 7, 9, 13, 14, 22, 26, 28], spam users [1, 5, 15, 18, 26], or spam user groups [19, 29]. (See §4 for details)

These have made considerable progress in understanding and spotting opinion spam, however the problem remains far from fully solved. In this work, we capitalize on our prior work [1] to propose a new method, SpEagle (for Spam Eagle), that can utilize clues from all of *metadata* (text, timestamp, rating) as well as *relational data* (review network), and harness them collectively under a unified framework to spot spam users, fake reviews, as well as targeted products. Moreover, SpEagle can seamlessly integrate labels on any subset of objects (user, review, and/or product) when available, without any changes in its algorithm (See Figure 1). We summarize the contributions of this work as follows.

- We propose SpEagle, a new approach for the opinion spam detection problem, which *ties together* relational data with metadata, i.e., it utilizes all of graph, behavioral, and review content information collectively.
- SpEagle considers the user–review–product graph to formulate the problem as a network-based classification task, in which users are labeled as spammer or benign, reviews as fake or genuine, and products as target or non-target (of spam) (§2.2). This formulation accepts prior knowledge on the class distribution of the nodes, which is estimated from metadata. In particular, SpEagle uses the metadata (text, timestamps, ratings) to design and extract indicative features of spam which are converted into a spam score to be used as part of class priors (§2.2.1).
- SpEagle works in a completely *unsupervised* fashion. However, it is amenable to easily leverage label information (when available). As such, we introduce a *semi-supervised* version of our method called SpEagle$^+$, which accepts as input labels for a (small) set of user, review, and/or product nodes in the graph (§2.2.2). The integration of labels is efficient and seamless; it requires *no* training on the labeled data and the main inference steps of the algorithm remain intact (§2.2.3).
- SpEagle extracts features for all user, review, and product nodes using metadata. To reduce computational overhead, we investigate the effectiveness of our features and their categories (user vs. review vs. product and text vs. behavior), and design SpLite (for SpEagle-Light), which uses only a few review features (and completely avoids feature extraction for users and products) (§2.2.4).

We evaluate our method on three real-world datasets collected from Yelp.com, containing filtered (spam) and recommended (non-spam) reviews. To the best of our knowledge, our work provides the largest scale *quantitative* evaluation to date for the opinion spam problem. Our results demonstrate that (*i*) SpEagle outperforms several baselines and state-of-the-art techniques [1, 26], (*ii*) performance can be boosted significantly by SpEagle$^+$ with only limited supervision, and (*iii*) SpLite provides significant speed-up with only a slight compromise in detection performance (§3).

2. METHODOLOGY

In this work, we build on our FraudEagle framework [1] and extend it in two main directions; (1) we expand its graph representation (structure and semantics), and (2) we incorporate meta-information into its network-only solution.

In a nutshell, we formulate the spam detection problem as a network classification task on the user-review-product network. In this task, users are to be classified as *spammer* or *benign*, products as *targeted* or *non-targeted*, and reviews as *fake* or *genuine*. To aid the network classification, we utilize additional metadata (i.e., ratings, time stamps, and review text) to extract indicative features of spam, which we incorporate into the inference procedure. Our proposed method works in a completely unsupervised fashion, however it can easily accommodate labels when available. As such, it is amenable to semi-supervised detection.

In §2.1, we first introduce FraudEagle for completeness and to lay out the main framework that forms the foundation for our proposed method SpEagle, described in §2.2.

2.1 FraudEagle Framework

The network representation used by FraudEagle [1] is the user–product bipartite network with signed edges. The user-product network $G = (V, E^{\pm})$ contains N user nodes $U = \{u_1, \ldots, u_N\}$ and M product nodes $P = \{p_1, \ldots, p_M\}$, $V = U \cup P$, connected through signed edges $(u_i, p_j, s) \in E^{\pm}$. The edges capture 'review' relations, where each edge sign $s \in \{+, -\}$ depicts whether a review is positive or negative.

The goal of classification on a network is to assign a label to each node. In their formulation, the domain of class labels is $\mathcal{L}_U = \{benign, spammer\}$ for users and $\mathcal{L}_P = \{good\text{-}quality, bad\text{-}quality\}$ for products. To formally define the classification problem, the network is represented as a pairwise Markov Random Field (MRF) [11]. An MRF model consists of an undirected graph where each node i is associated with a random variable Y_i that can be in one of a finite number of states, that is, classes. In pairwise MRFs, the label of a node is assumed to be dependent only on its neighbors and independent of all the other nodes in the graph. The joint probability of labels is then written as a product of individual and pairwise factors, parameterized over the nodes and the edges, respectively:

$$P(\mathbf{y}) = \frac{1}{Z} \prod_{Y_i \in V} \phi_i(y_i) \prod_{(Y_i, Y_j, s) \in E^{\pm}} \psi_{ij}^{s}(y_i, y_j) \qquad (1)$$

where $\mathbf{y}$ denotes an assignment of labels to all nodes, y_i refers to node i's assigned label, and Z is the normalization constant. The individual factors $\phi : \mathcal{L} \to \mathbb{R}^+$ are called *prior* (or node) potentials, and represent initial class probabilities for each node, often initialized based on prior knowledge. The pairwise factors $\psi^s : \mathcal{L}_U \times \mathcal{L}_P \to \mathbb{R}^+$ are called *compatibility* (or edge) potentials, and capture the likelihood of a node with label y_i to be connected to a node with label y_j through an edge with sign s. To reduce the number of model parameters, the edge potentials are often shared (in this case, across all edges with the same sign). In case no external information or domain knowledge is available, one can set all the priors as unbiased/uninformative (i.e., equal probability for each class). Otherwise, one can estimate them for every node separately using external sources.

In FraudEagle, one of the design goals is to avoid the requirement for additional sources to estimate priors. This also facilitates generality, as external sources may differ across domains. Therefore, the priors are all set as unbiased to $\phi_i = \{0.5, 0.5\}$, $\forall i \in V$. On the other hand, the compatibility potentials are ideally estimated from labeled data. In a problem setting such as opinion spam detection however, obtaining reasonable amount of labeled data is extremely difficult due to the challenges that human annotators face. Therefore, these parameters are set based on several intuitions reflecting the modus-operandi of spammers and otherwise normal behavior of regular users: (*i*) benign users often write positive reviews to good-quality products and negative reviews to bad-quality products, in contrast (*ii*) spammers more likely write positive reviews to bad-quality products (to hype) and/or negative reviews to good-quality products (to defame, e.g., under competition), (*iii*) spammers can also write genuine-looking reviews to camouflage their misactivities (i.e., positive reviews to good-quality and negative reviews to bad-quality products), although (*iv*) the same (e.g., writing negative reviews to good-quality products) is less likely for benign users (but can happen depending on individual experience). Under these assumptions, the fol-

lowing parameter settings are used, for a small ϵ value.[1]

$\psi^{s=+}$	Product		$\psi^{s=-}$	Product	
User	*good*	*bad*	User	*good*	*bad*
benign	$1-\epsilon$	ϵ	*benign*	ϵ	$1-\epsilon$
spammer	2ϵ	$1-2\epsilon$	*spammer*	$1-2\epsilon$	2ϵ

Given the model parameters (ϕ_i, $\forall i \in V$ and ψ^s for $s \in \{+,-\}$), the task is to infer the maximum likelihood assignment of states (class labels) to the random variables associated with the nodes, in other words, to find the $\mathbf{y}$ that maximizes the joint probability of the network as given in Eqn. (1). This is the inference problem which is combinatorially hard. The enumeration of all possible assignments is exponential to the network size and thus intractable for large graphs. Exact inference is known to be NP-hard for general MRFs, where instead iterative approximate inference algorithms such as Loopy Belief Propagation (LBP) [31] are used. We describe the details of the inference procedure in the context of our proposed method below.

2.2 Proposed Method SpEagle

Besides the relational information between users and products, there exist a variety of metadata in review datasets. Those include the text content of reviews, timestamps, and star ratings. Earlier work have used metadata to design features that are indicative of spam [9, 13, 19, 22, 28]. FraudEagle, on the other hand, completely excludes meta-information and solely relies on the relational structure of the data. The main motivation of this work is to bridge the relational data and the metadata to improve detection performance. In particular, we aim to leverage the metadata to estimate initial class probabilities for users, products, and reviews, which we incorporate as prior potentials of the nodes under a new MRF model.

Our motivation requires two main changes to be made in the FraudEagle framework. First, we represent the reviews as nodes inside the network. As such, we model the relational data as a user–review–product network, where each review node is connected to its corresponding user and product nodes (See Figure 1). The reason is that we can use metadata to estimate a "suspicion score" not only for users and products, but also for reviews. Representing reviews explicitly as nodes enables us to readily integrate this information to the formulation as prior potentials of reviews.

The second change is related to the semantics of class labels for products. The domain of class labels for products in FraudEagle is $\mathcal{L}_P = \{\textit{good-quality}, \textit{bad-quality}\}$. However, the metadata does not lend itself to estimating meaningful priors for these classes. Two possible ways of inferring product quality, average rating and sentiment analysis of reviews, could be misleading—both average ratings and review sentiment of products associated with fake reviewing are tempered with, and thus are not reliable. We could, on the other hand, use the metadata to perform behavioral analysis on products to characterize the likelihood that they are under manipulation (See §2.2.1). In other words, the "suspicion score" of a product estimated from metadata would not translate to its quality but to its likelihood of being a target of opinion spam. Therefore, in our formulation the products are labeled as $\mathcal{L}_P = \{\textit{non-target}, \textit{target}\}$. As the semantics of the network representation has changed, we also discard the sentiment, i.e. the signs on the edges and use an unsigned network $G = (V, E)$. This is because under the new setting, a targeted product can be associated with negative as well as positive fake reviews, when manipulated with an intent to defame or to hype, respectively.

The joint probability of our formulation can be written similar to Eqn. (1), where the node set $V = U \cup P \cup R$ now consists of three types of nodes, including the Q review nodes $R = \{r_1, \ldots, r_Q\}$, with labels from domain $\mathcal{L}_R = \{\textit{genuine}, \textit{fake}\}$. Moreover, the compatibility potentials are typed as ψ^t_{ij} reflecting the two types of relations in the network; the user-review edges $(u_i, r_k, t = \text{`}\textit{write}\text{'}) \in E$ and the review-product edges $(r_k, p_j, t = \text{`}\textit{belong}\text{'}) \in E$. In terms of setting the model parameters, we estimate the prior potentials ϕ_i from metadata for all three types of nodes, $\forall i \in V$, which we describe in §2.2.1. On the other hand, we initialize the compatibility potentials ψ^t_{ij} so as to enforce homophily [17]. In particular, we assume that all the reviews written by spammers (benign users) are fake (genuine), and that with high probability fake (genuine) reviews belong to targeted (non-targeted) products; although with some probability fake reviews may also belong to non-targeted products as part of camouflage, and similarly genuine reviews may co-exist along with fake reviews for targeted products. Nevertheless, we assume that the majority of the reviews for targeted (non-targeted) products are fake (genuine), which possibly needs to hold true for a spam campaign to be able to manipulate the average rating of a targeted product successfully. Overall, SpEagle uses the following settings.

Table 1: Compatibility potentials ψ^t used by SpEagle.

	User ($\psi^{t=\text{`}write\text{'}}$)		($\psi^{t=\text{`}belong\text{'}}$) **Product**	
Review	*benign*	*spammer*	*non-target*	*target*
genuine	1	0	$1-\epsilon$	ϵ
fake	0	1	ϵ	$1-\epsilon$

Next, we describe how we estimate the prior potentials from metadata for all the user, review, and product nodes. Then, we introduce the *semi-supervised* version of SpEagle and show how labels can be used if available. We provide an outline of our algorithm and present the inference steps for computing the class assignments. Finally, we introduce a *light* version of SpEagle for computational speed-up.

2.2.1 From metadata to features to priors

To estimate the prior potentials, we first extract indicative features of spam from available metadata (ratings, timestamps, review text) and then convert them to prior class probabilities. The priors are estimated for all three types of nodes. As such, we compute features for users, products, and reviews separately. Most of our features have been used several times in previous work on opinion spam detection including [5, 9, 13, 15, 18, 20], while several are introduced in this work. Table 2 includes brief descriptions for the features. Most of them are self-explanatory, and hence we omit detailed explanation for brevity. Instead, we provide references to prior work where they have also been used.

In particular, we show the user and product features in Table 2 (top). All but one of these features can be defined for both, where we either consider all reviews of a user or all reviews of a product. One feature (BST) applies only to users, which captures the burstiness related to the age of a user, defined as the number of days between their first and the last review. Intuitively, spammers are short-

[1] Sensitivity analysis in [1] found that $\epsilon \in [0.01, 0.15]$ yields desirable and comparable results. In this work we use $\epsilon = 0.1$.

term members of a review site rather than veterans, which is characterized by BST. We also show the review features in Table 2 (bottom). Note that all the features are categorized into two—behavioral versus text-based. Text-based ones are solely derived from review content. Behavioral features are based on time stamps, ratings, distributions, ranks, etc. In the experiments, we evaluate the effectiveness of individual features as well as the feature categories.

Given a set of values $\{x_{1i}, \ldots, x_{Fi}\}$ for the F features of a node i, the next step is to combine them into a spam score $S_i \in [0, 1]$, such that the class priors can be initialized as $\{1-S_i, S_i\}$. The features, however, may have different scales and varying distributions. To unify them into a comparable scale and interpretation, we leverage the cumulative distribution function (CDF). In particular, when we design the features, we have an understanding of whether a *high* (H) or a *low* (L) value is more suspicious for each feature. For example, high average rating deviation (avgRD) and low entropy of rating distribution (ERD) are suspicious. To quantify the extremity of a feature value x, we then use the empirical CDF to estimate the probability that the data contains a value as low or as high as x. More formally, for each feature l, $1 \leq l \leq F$, and its corresponding value x_{li}, we compute

$$f(x_{li}) = \begin{cases} 1 - P(X_l \leq x_{li}), & \textit{if high is suspicious (H)} \\ P(X_l \leq x_{li}), & \textit{otherwise (L)} \end{cases}$$

where X_l denotes a real-valued random variable associated with feature l with probability distribution P. To compute $f(\cdot)$, we use the *empirical* probability distribution of each feature over all the nodes of the given type. Overall, the features with suspiciously low or high values all receive low f values. Finally we combine these f values to compute the spam score of a node i as follows.

$$S_i = 1 - \sqrt{\frac{\sum_{l=1}^{F} f(x_{li})^2}{F}} \tag{2}$$

Figure 2 shows the CDF of example features for filtered (considered as spam) versus recommended (considered as non-spam) reviews and associated reviewers in one of our Yelp datasets. Notice that spam review(er)s obtain higher values for certain features (e.g., those in top row) and lower values for some others (bottom row)—hence the (H) and (L) distinction in the $f(\cdot)$ function above. Also notice that the individual features provide only weak evidence on their own, as the CDF curves of the classes are somewhat close to each other. We aim to obtain a stronger signal by combining the multiple evidence from all the features using Eqn. (2).

2.2.2 *Semi-supervised* SpEagle

One of the key advantages of our formulation is that it enables seamless integration of labeled data when available. We describe two possible ways to incorporate label information. The first scenario does not involve any learning on the labeled data. Specifically, given the labels for a set of nodes (reviews, users, and/or products), we simply initiate the priors as $\{\epsilon, 1-\epsilon\}$ for those that are associated with spam (i.e., *fake*, *spammer*, or *target*), and $\{1-\epsilon, \epsilon\}$ otherwise. The priors of unlabeled nodes are estimated from metadata as given in Eqn. (2). The inference procedure remains the same. As this integration does not require model training or any other changes, it is extremely efficient. It is particularly suitable when the size of the labeled data is too small or unbalanced to learn from.

Table 2: Features for users, products, and reviews derived from metadata; categorized as behavior and text-based. H/L depicts if a High/Low value of the feature is more likely to be associated with spam.

			User & Product Features
Behavior	MNR	H	Max. number of reviews written in a day [18, 20]
	PR	H	Ratio of positive reviews (4-5 star) [20]
	NR	H	Ratio of negative reviews (1-2 star) [20]
	avgRD	H	Avg. rating deviation $avg(\lvert d_{i*}\rvert)$ of user (product) i's reviews [5, 15, 20], where $\lvert d_{ij}\rvert$ is absolute rating deviation of i's rating from j's average rating: $avg_{e_{ij} \in E_{i*}} \lvert d_{ij}\rvert$, for $d_{ij} = r_{ij} - avg_{e \in E_{*j}} r(e)$
	WRD	H	Weighted rating deviation [15], where reviews are weighed by recency: $\frac{\sum_{e_{ij} \in E_{i*}} \lvert d_{ij}\rvert w_{ij}}{\sum_{e_{ij} \in E_{i*}} w_{ij}}$, for $w_{ij} = \frac{1}{(t_{ij})^{\alpha}}$ (t_{ij} is rank order of review e_{ij} among reviews of j, $\alpha = 1.5$ is decay rate)
	BST	H	Burstiness [5, 20]—spammers are often short-term members of the site. $x_{BST}(i) = \begin{cases} 0, & \textit{if } L(i) - F(i) > \tau \\ 1 - \frac{L(i)-F(i)}{\tau}, & \textit{otherwise} \end{cases}$ where $L(i) - F(i)$ is number of days between last and first review of i, $\tau = 28$ days.
	ERD	L	Entropy of rating distribution of user's (product's) reviews [new]
	ETG	L	Entropy of temporal gapsΔ_t's. Given the temporal line-up of a user's (product's) reviews, each Δ_t denotes the temporal gap in days between consecutive pairs [new]
Text	RL	L	Avg. review length in number of words [20]
	ACS	H	Avg. content similarity—pairwise cosine similarity among user's (product's) reviews, where a review is represented as a bag-of-bigrams [5, 15]
	MCS	H	Max. content similarity—maximum cosine similarity among all review pairs [18, 20]
			Review Features
Behavior	Rank	L	Rank order among all the reviews of product [9]
	RD	H	Absolute rating deviation from product's average rating [13]
	EXT	H	Extremity of rating [18]: $x_{EXT} = 1$ for ratings $\{4, 5\}$, 0 otherwise (for $\{1, 2, 3\}$)
	DEV	H	Thresholded rating deviation of review e_{ij} [18]: $x_{DEV}(i) = \begin{cases} 1, & \textit{if } \frac{\lvert r_{ij} - avg_{e \in E_{*j}} r(e)\rvert}{4} > \beta_1 \\ 0, & \textit{otherwise} \end{cases}$ where β_1 is learned by recursive minimal entropy partitioning
	ETF	H	Early time frame [18]—spammers often review early to increase impact. $x_{ETF}(f(e_{ij})) = 1$ if $f(e_{ij}) > \beta_2$, and 0 otherwise, where, $f(e_{ij}) = \begin{cases} 0, & \textit{if } T(i,j) - F(j) > \delta \\ \frac{T(i,j)-F(j)}{\delta}, & \textit{otherwise} \end{cases}$ where $T(i,j) - F(j)$ is the difference between the time of review e_{ij} and first review j, for $\delta = 7$ months, and β_2 is estimated by recursive minimal entropy partitioning
	ISR	H	Is singleton? If review is user's sole review, then $x_{ISR} = 1$, otherwise 0 [new]
Text	PCW	H	Percentage of ALL-capitals words [9, 13]
	PC	H	Percentage of capital letters [13]
	L	L	Review length in words [13]
	PP1	L	Ratio of 1st person pronouns ('I', 'my', etc.) [13]
	RES	H	Ratio of exclamation sentences containing '!' [13]
	SW	H	Ratio of subjective words (by sentiWordNet) [13]
	OW	L	Ratio of objective words (by sentiWordNet) [13]
	F	H	Frequency of review (approximated using locality sensitive hashing) [new]
	DL_u	L	Description length (information-theoretic) based on unigrams (i.e., words) [new]
	DL_b	L	Description length based on bigrams [new]

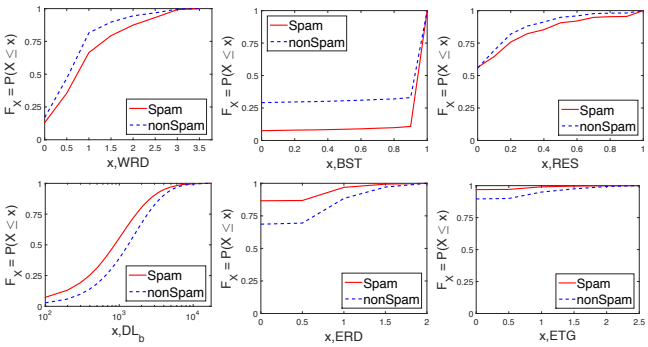


Figure 2: CDF distribution of example features for spam vs. non-spam review(er)s. Spam review(er)s often have (H)igher values for features in top row, and (L)ower values for those in bottom row.

An alternative approach is to use the labeled data to train classifier models (one for each type of node), and initialize the prior potentials for unlabeled nodes with the class probabilities provided by the classifiers. This approach is particularly possible in our setting, where we extract features indicative of spam for each node in our network. When no labels are available, we transform these features into priors in an unsupervised fashion. In the semi-supervised scenario, one can instead use the features and the supplied labels to train models to obtain "supervised priors". Note that the class distribution of labeled data would likely be unbalanced. Most learning methods are sensitive to skewed label distribution. Therefore, one needs to either use cost-sensitive methods [4, 25] or deploy under/over-sampling techniques to balance the training data before learning [2]. Here again, the modifications are only in initializing and estimating the prior potentials of labeled and unlabeled nodes, respectively, and the inference procedure remains intact. We discuss the details of our proposed algorithm next.

2.2.3 *The algorithm*

We provide the steps of our SpEagle in Algorithm 1. As input, we take the user–review–product graph G, compatibility potential parameters ψ^t as given in Table 1, available metadata for feature extraction, and a set of node labels L (if any). Note that L can contain labels for a mixture of user, review, and product nodes, or can be empty. We output the corresponding class probabilities for all the unlabeled nodes.

First, we compute or initialize the prior class probabilities for all the nodes (Lines 3-10). Specifically, the priors are set to $\phi \leftarrow \{\epsilon, 1-\epsilon\}$ for those labeled nodes that belong to the spam category (i.e., *fake*, *spammer*, or *target*), and to $\phi \leftarrow \{1-\epsilon, \epsilon\}$ for nodes labeled as non-spam.[2] For unlabeled nodes, we compute their corresponding features in Table 2 (depending on their type), combine them into a spam score S using Eqn. (2), and set the priors as $\phi \leftarrow \{1-S, S\}$.

In the remainder, we follow the main steps of the Loopy Belief Propagation (LBP) algorithm [31]. LBP is based on iterative message passing between the connected nodes. We first initialize all the messages to 1 (Lines 11-13). At every iteration, a *message* $m_{i\to j}$ is then sent from each node i to each neighboring node j, where $T_i, T_j \in \{U, R, P\}$ denote

[2]Our experiments showed that the alternative approach discussed in §2.2.2 produces similar results. We list the first approach in Algorithm 1 due to its efficiency, as it requires no training.

Algorithm 1: SpEagle

1 **Input:** User–Review–Product graph $G = (V, E)$, compatibility potentials ψ^t (Table 1), review metada (ratings, timestamps, text), labeled node set L
2 **Output:** Class probabilities for each node $i \in V \backslash L$
3 **foreach** $i \in V$ **do** // *compute/initialize priors*
4 **if** $i \in L$ **then**
5 **if** *i is positive (spam) class* **then** $\phi_i \leftarrow \{\epsilon, 1-\epsilon\}$
6 **else** $\phi_i \leftarrow \{1-\epsilon, \epsilon\}$
7 **else**
8 Extract corresponding features in Table 2
9 Compute spam score S_i using Eqn. (2)
10 $\phi_i \leftarrow \{1-S_i, S_i\}$
11 **foreach** $(Y_i^{T_i}, Y_j^{T_j}, t) \in E$ **do** // *initialize all msg.s*
12 **foreach** $y_j \in \mathcal{L}_{T_j}$ **do**
13 $m_{i\to j}(y_j) \leftarrow 1$
14 **repeat**// *iterative message passing*
15 **foreach** $(Y_i^{T_i}, Y_j^{T_j}, t) \in E$ **do**
16 **foreach** $y_j \in \mathcal{L}_{T_j}$ **do**
17 $$m_{i\to j}(y_j) = \alpha \sum_{y_i \in \mathcal{L}_{T_i}} \left(\phi_i(y_i)\ \psi_{ij}^t(y_i, y_j) \prod_{Y_k \in \mathcal{Y}_{\mathcal{N}_i} \backslash Y_j} m_{k\to i}(y_i) \right)$$
18 **until** messages stop changing within a δ threshold
19 **foreach** $Y_i^{T_i} \in \mathcal{Y}_{V\backslash L}$ **do** // *compute final beliefs*
20 **foreach** $y_i \in \mathcal{L}_{T_i}$ **do**
21 $$b_i(y_i) = \beta\ \phi_i(y_i) \prod_{Y_j \in \mathcal{Y}_{\mathcal{N}_i}} m_{j\to i}(y_i)$$

the type of node i and j, respectively. The message represents the belief of i about j, i.e., what i "thinks" j's label distribution is. More formally, $m_{i\to j}$ captures the probability distribution over the class labels of j, and is computed based on the class priors of i, the compatibility potentials of the *type* of edge that connects i and j, and the messages that i receive from its neighbors excluding j. The exact expression is given in Line 17, where $\mathcal{N}_i$ denotes the set of i's neighbors and α is a normalization constant to ensure that class probabilities sum to 1. These messages are exchanged iteratively over the edges until a "consensus" is reached, i.e., the messages stop changing between consecutive iterations within a small threshold such as $\delta = 10^{-3}$ (Lines 14-18).

When the messages stabilize, we compute the marginal probability, called the *belief* $b_i(y_i)$, of assigning each Y_i associated with a node of type $T_i \in \{U, R, P\}$ with the label y_i in label domain $\mathcal{L}_{T_i}$ (Lines 19-21). The exact expression is given in Line 21, where β is the normalization constant so that the marginal probabilities sum to 1. For classification, one can assign labels based on $\max_{y_i} b_i(y_i)$. For ranking, we sort by the probability values $b_i(y_i)$, where $y_i = spammer$ and $y_i = fake$ respectively for users and reviews.

2.2.4 *Light-weight* SpEagle

The original SpEagle computes all the features for every (unlabeled) node of each type in the graph (Line 8, Alg. 1). In the experiments we investigate the effectiveness of the features and identify a small subset of review features that

produces comparable performance to using all of them. As such, we propose a light-weight version of our method, called SpLite (for SpEagle-Light), where we initialize the priors for unlabeled reviews based on the spam score computed only on those features, and use unbiased priors $\{0.5, 0.5\}$ for (unlabeled) users and products. This significantly reduces the feature extraction overhead for reviews, and completely avoids it for users and products, enabling speed-up with only slight compromise in performance. We compare the performance and running time of SpLite and SpEagle in §3.4.

3. EVALUATION

We evaluate our approach *quantitative*ly on real-world datasets with near-ground-truth. In the following we first describe our datasets, evaluation metrics, and compared methods and then present the performance results.

Data description.

In this study, we use three datasets collected from Yelp.com, summary statistics of which are given in Table 3. The first dataset, named `YelpChi`, has been collected and used by [20] and contains reviews for a set of restaurants and hotels in the Chicago area. We collected two more datasets from Yelp for this work, named as `YelpNYC` and `YelpZip`. `YelpNYC` contains reviews for restaurants located in NYC. `YelpZip` is even larger, where we start with a zipcode in NY state, collect reviews for restaurants in that zipcode[3], increase the zipcode number incrementally, and repeat. The zipcodes are organized by geography, as such this process gives us reviews for restaurants in a continuous region of the U.S. map, including NJ, VT, CT, and PA.

Yelp has a filtering algorithm in place that identifies fake/suspicious reviews and separates them into a filtered list. The filtered reviews are also made public; the Yelp page of a business shows the *recommended* reviews, while it is also possible to view the filtered/unrecommended reviews through a link at the bottom of the page. While the Yelp anti-fraud filter is not perfect (hence the "near" ground truth), it has been found to produce accurate results [27]. Our Yelp datasets contain both recommended and filtered reviews. We consider them as *genuine* and $fake$, respectively. We also separate the users into two classes; $spammers$: authors of $fake$ (filtered) reviews, and *benign*: authors with no filtered reviews. We evaluate SpEagle on both tasks: spammer detection and fake review detection.

Evaluation metrics.

We use four well-known ranking based metrics for our performance evaluation. We obtain the (i) precision-recall (PR) as well as (ii) ROC (true positive rate vs. FPR) curves, where the points on a curve are obtained by varying the classification threshold. We compute the area under the curve, respectively denoted as AP (average precision) and AUC. For problems such as outlier/spam/fraud detection, often the quality at the top of the ranking results are the most important. Therefore, we also inspect (iii) *precision*@k and (iv) $NDCG@k$, for $k = \{100, 200, \ldots, 1000\}$. *Precision*@$k$ captures the ratio of spam(mers) in top k positions. $NDCG@k$ provides a weighted score, which favors rankings where spam(mers) are ranked closer to the top. In particular, $NDCG@k = \frac{DCG@k}{IDCG@k}$ for $DCG@k = \sum_{i=1}^{k} \frac{2^{l_i}-1}{\log_2(i+1)}$, where l_i captures the true label of item at rank i (1: spam, 0: non-spam) and $IDCG@k$ is the DCG for ideal ranking where all l_i's are 1.

[3]Yelp allows to search for restaurants by zipcode, e.g., `http://www.yelp.com/search?cflt=restaurants&find_loc=11794`

Table 3: Review datasets used in this work.

Dataset	#Reviews (filtered %)	#Users (spammer %)	#Products (rest.&hotel)
`YelpChi`	67,395 (13.23%)	38,063 (20.33%)	201
`YelpNYC`	359,052 (10.27%)	160,225 (17.79%)	923
`YelpZip`	608,598 (13.22%)	260,277 (23.91%)	5,044

Compared Methods.

We compare the performance of SpEagle to FraudEagle [1] as well as to the graph-based approach by [26] denoted as Wang et al., thanks to their publicly available implementations. We also consider Prior, where we use the spam scores (for users and reviews) computed solely based on metadata as discussed in §2.2.1. Prior does not use the network information, and hence corresponds to SpEagle without LBP. We also compare to the semi-supervised SpEagle$^+$ with varying amount of labeled data, as well as to the computationally light-weight version SpLite.

3.1 Detection results

We start by comparing the detection performance of the methods. Table 4 provides the AP and AUC over all three datasets for both user and review ranking. Notice that SpEagle outperforms all others, Wang et al. and Prior being the second best method for user and review ranking, respectively. The superiority of SpEagle's ranking becomes more evident when the top of the ranking results are considered through *precision*@k in Table 5, as well as the $NDCG@k$ in Figure 3.

3.2 Semi-supervised detection

Next we investigate how much the detection performance can be improved by semi-supervision; i.e., providing SpEagle with a subset of the *review* labels. We analyze performance for varying amount of labeled data. Our datasets contain different number of reviews (See Table 3), where `YelpChi` is the smallest dataset and `YelpZip` is the largest. To keep the number of provided labels at realistic size, we label smaller percentages of the data for larger datasets; in particular $\{1\%, 5\%, 10\%\}$ for `YelpChi`, $\{0.5\%, 1\%, 2\%\}$ for `YelpNYC`, and $\{0.25\%, 0.5\%, 1\%\}$ for `YelpZip`.

Figure 3 shows the $NDCG@k$ performance of semi-supervised SpEagle, denoted as SpEagle$^+$, on all three datasets for both user and review ranking, where varying amount (%) of the review labels are provided (values are averaged over 10 independent runs). Table 6 shows the corresponding *precision*@k values for review ranking (user ranking performance is similar, omitted for brevity). We notice that the performance is improved considerably even with very small amount of supervision, where the semi-supervised results are significantly better than all the competing methods. For example, labeling only 1% of the reviews (around 670 labels for `YelpChi`, 3600 for `YelpNYC`, and 6000 for `YelpZip`), *precision*@k for review ranking is increased by 4-22% on `YelpChi`, 44-56% on `YelpNYC`, and 39-47% on `YelpZip` in absolute terms, for $k \in [100, 1000]$.[4] Table 4 also includes the AP and AUC performance of SpEagle$^+$ across all datasets for 1% labeled data.

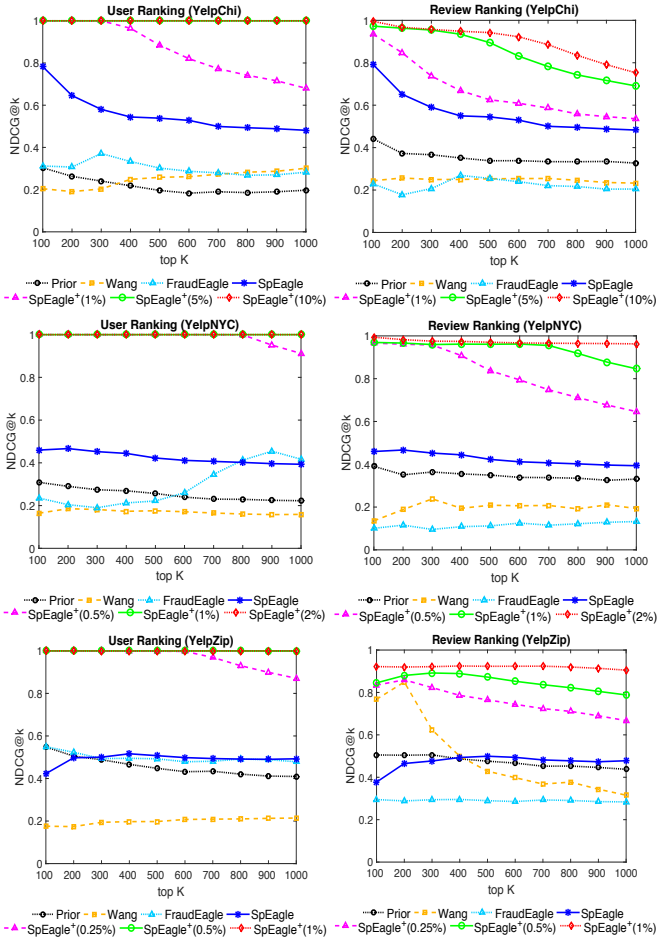
[4]The corresponding absolute improvements for user ranking are 18-47% on `YelpChi`, 54-61% on `YelpNYC`, and 46-56% on `YelpZip`.

Table 4: AP and AUC performance of compared methods on all three datasets.

	User Ranking						Review Ranking					
	AP			AUC			AP			AUC		
	Y'Chi	Y'NYC	Y'Zip	Y'Chi	Y'NYC	Y'Zip	Y'Chi	Y'NYC	Y'Zip	Y'Chi	Y'NYC	Y'Zip
RANDOM	0.2024	0.1782	0.2392	0.5000	0.5000	0.5000	0.1327	0.1028	0.1321	0.5000	0.5000	0.5000
FRAUDEAGLE	0.2537	0.2233	0.3091	0.6124	0.6062	0.6175	0.1067	0.1122	0.1524	0.3735	0.5063	0.5326
WANG ET AL.	0.2659	0.2381	0.3306	0.6167	0.6207	0.6554	0.1518	0.1255	0.1803	0.5062	0.5415	0.5982
PRIOR	0.2157	0.1826	0.2550	0.5294	0.5081	0.5269	0.2241	0.1789	0.2352	0.6707	0.6705	0.6838
SPEAGLE	**0.3393**	**0.2680**	**0.3616**	**0.6905**	**0.6575**	**0.6710**	**0.3236**	**0.2460**	**0.3319**	**0.7887**	**0.7695**	**0.7942**
SPEAGLE$^+$(1%)	0.3967	0.3480	0.4245	0.7078	0.6828	0.6907	0.3352	0.2757	0.3545	0.7951	0.7829	0.8040
SPLITE$^+$ (1%)	0.3777	0.3331	0.4218	0.6744	0.6542	0.6784	0.3124	0.2550	0.3448	0.7693	0.7631	0.7923

Table 5: *Precision@k* of compared methods on (from top to bottom) YelpChi, YelpNYC, and YelpZip.

	User Ranking				Review Ranking			
k	PRIOR	FRAUDEAGLE	WANG ET AL.	SPEAGLE	PRIOR	FRAUDEAGLE	WANG ET AL.	SPEAGLE
100	0.32	0.30	0.21	**0.73**	0.38	0.25	0.24	**0.74**
200	0.26	0.30	0.19	**0.59**	0.33	0.18	0.26	**0.59**
300	0.23	0.38	0.21	**0.52**	0.33	0.21	0.25	**0.53**
400	0.21	0.33	0.26	**0.49**	0.32	0.29	0.25	**0.50**
500	0.18	0.29	0.27	**0.50**	0.31	0.27	0.25	**0.50**
600	0.17	0.28	0.27	**0.49**	0.32	0.25	0.26	**0.49**
700	0.18	0.27	0.29	**0.46**	0.31	0.22	0.26	**0.46**
800	0.18	0.26	0.30	**0.46**	0.32	0.22	0.25	**0.46**
900	0.18	0.26	0.30	**0.46**	0.32	0.20	0.23	**0.45**
1000	0.19	0.28	0.32	**0.45**	0.31	0.20	0.23	**0.45**
100	0.34	0.21	0.15	**0.44**	0.34	0.10	0.17	**0.44**
200	0.30	0.19	0.19	**0.46**	0.32	0.12	0.22	**0.46**
300	0.28	0.17	0.18	**0.44**	0.34	0.09	0.27	**0.44**
400	0.27	0.21	0.17	**0.44**	0.34	0.11	0.21	**0.44**
500	0.25	0.22	0.17	**0.41**	0.33	0.11	0.22	**0.41**
600	0.23	0.27	0.17	**0.40**	0.32	0.13	0.22	**0.40**
700	0.22	0.37	0.16	**0.39**	0.32	0.12	0.22	**0.39**
800	0.22	**0.45**	0.16	0.39	0.32	0.13	0.20	**0.39**
900	0.22	**0.50**	0.15	0.38	0.31	0.13	0.22	**0.38**
1000	0.22	**0.45**	0.16	0.38	0.32	0.14	0.20	**0.38**
100	0.51	**0.55**	0.18	0.44	0.51	0.29	**0.86**	0.43
200	0.48	0.52	0.18	**0.53**	0.51	0.29	**0.92**	0.52
300	0.46	0.48	0.20	**0.52**	0.51	0.29	**0.61**	0.51
400	0.44	0.49	0.20	**0.54**	0.48	0.30	0.46	**0.53**
500	0.42	0.48	0.20	**0.52**	0.47	0.29	0.38	**0.53**
600	0.41	0.47	0.21	**0.51**	0.46	0.28	0.35	**0.52**
700	0.41	0.47	0.21	**0.50**	0.44	0.29	0.32	**0.50**
800	0.40	0.49	0.22	**0.50**	0.45	0.29	0.34	**0.49**
900	0.39	0.48	0.22	**0.49**	0.44	0.28	0.30	**0.48**
1000	0.39	0.47	0.22	**0.50**	0.43	0.28	0.27	**0.49**



(a) User Ranking (b) Review Ranking

Figure 3: *NDCG@k* of compared methods on (from top to bottom) YelpChi, YelpNYC, and YelpZip for both user and review ranking. Also shown are results for SPEAGLE$^+$ with varying % of labeled data.

3.3 Analyzing priors

Next we investigate the informativeness of feature categories and individual features in estimating effective priors.

User vs. Review vs. Product priors.

We start by analyzing the user, review, and product priors. To study the effectiveness of a certain group of priors (e.g., user, or user+review), we only initialize the priors for the nodes in that group in the graph (as estimated from metadata) and set the remaining node priors to unbiased, i.e. $\{0.5, 0.5\}$. We then compare the performance of SPEAGLE with priors of various groups.

Table 7 shows the AP and AUC performance of SPEAGLE across datasets with various prior groups. We find that the review priors produce the most effective results, followed by user priors, and product priors. The difference in performance is especially pronounced on our largest dataset YelpZip. To our surprise, we find that the product priors alone yield performance that is lower than that by random ranking. As a result, SPEAGLE with only user and review priors performs almost as well as using all the priors.

Text- vs. Behavior-based priors.

Recall from Table 2 that our features are derived from review text as well as behavioral clues. Here we investigate the performance of SPEAGLE when priors are estimated from

Table 6: *Precision@k* **of SpEagle$^+$ for review ranking on all three datasets with varying % of labeled data.**

	YelpChi				YelpNYC				YelpZip			
k	**0%**	**1%**	**5%**	**10%**	**0%**	**0.5%**	**1%**	**2%**	**0%**	**0.25%**	**0.5%**	**1%**
100	0.7400	0.9300	0.9650	0.9950	0.4400	0.9650	0.9630	0.9930	0.4300	0.8740	0.8540	0.9090
200	0.5900	0.8195	0.9565	0.9600	0.4600	0.9595	0.9625	0.9790	0.5150	0.8850	0.8935	0.9130
300	0.5333	0.6910	0.9477	0.9500	0.4433	0.9557	0.9553	0.9713	0.5133	0.8303	0.9037	0.9173
400	0.4975	0.6162	0.9245	0.9408	0.4350	0.8935	0.9587	0.9710	0.5250	0.7823	0.8972	0.9225
500	0.5020	0.5736	0.8772	0.9344	0.4100	0.8076	0.9586	0.9664	0.5260	0.7574	0.8750	0.9212
600	0.4900	0.5617	0.8008	0.9110	0.3983	0.7602	0.9603	0.9633	0.5150	0.7320	0.8500	0.9218
700	0.4600	0.5407	0.7451	0.8671	0.3943	0.7079	0.9521	0.9623	0.4971	0.7090	0.8307	0.9226
800	0.4587	0.5125	0.7015	0.8078	0.3900	0.6685	0.9067	0.9616	0.4900	0.6946	0.8138	0.9178
900	0.4544	0.5018	0.6739	0.7570	0.3844	0.6307	0.8586	0.9610	0.4833	0.6711	0.7938	0.9106
1000	0.4510	0.4944	0.6471	0.7141	0.3820	0.5982	0.8225	0.9597	0.4880	0.6453	0.7744	0.9004

Table 7: AP and AUC performance of SpEagle when priors are initialized (estimated from metadata) for various node types; (U)sers, (R)eviews, (P)roducts (rest set to unbiased). P-priors yield the lowest performance, while R-priors are the most effective.

	User Ranking						Review Ranking					
	AP			AUC			AP			AUC		
	Y'Chi	Y'NYC	Y'Zip	Y'Chi	Y'NYC	Y'Zip	Y'Chi	Y'NYC	Y'Zip	Y'Chi	Y'NYC	Y'Zip
Random	0.2024	0.1782	0.2392	0.5000	0.5000	0.5000	0.1327	0.1028	0.1321	0.5000	0.5000	0.5000
SpEagle (U)	0.3197	0.2624	0.2808	0.6767	0.6483	0.6183	0.3043	0.2400	0.1427	0.7783	0.7629	0.5940
SpEagle (P)	0.1550	0.1357	0.1814	0.3905	0.3930	0.3801	0.0755	0.0640	0.0806	0.1643	0.2536	0.2277
SpEagle (R)	0.3226	0.2575	0.3449	0.6771	0.6477	0.6562	0.3098	0.2378	0.3180	0.7820	0.7656	0.7884
SpEagle (UR)	**0.3398**	**0.2680**	0.3615	**0.6905**	**0.6575**	0.6709	**0.3241**	**0.2460**	**0.3320**	**0.7887**	**0.7695**	**0.7942**
SpEagle (URP)	0.3393	**0.2680**	**0.3616**	**0.6905**	**0.6575**	**0.6710**	0.3236	**0.2460**	0.3319	**0.7887**	**0.7695**	**0.7942**

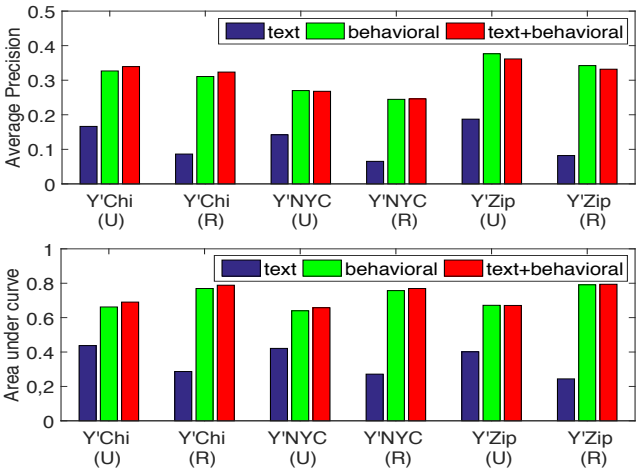


Figure 4: (top) AP and (bottom) AUC performance of SpEagle when various feature types are used to estimate priors; text, behavior, all (See Table 2) on all datasets for both (U)ser and (R)eview ranking.

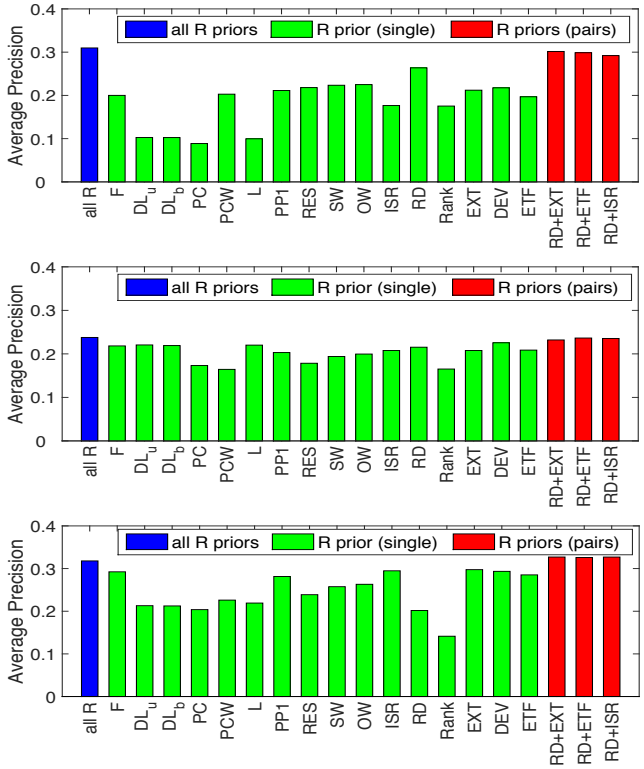


Figure 5: AP performance of SpEagle when all (green), individual (blue), and behavioral pairs (red, only 3 shown) of (R)eview features are used to estimate review priors (rest set to unbiased), on (from top to bottom) YelpChi, YelpNYC, and YelpZip.

only text-based versus only behavioral features (for all user, review, and product nodes) as compared to using all the possible features. Figure 4 shows the AP and AUC performance across all datasets for both user and review ranking.

We observe that using text-based features alone yields inferior performance compared to behavioral features. Moreover, behavioral features alone produce comparable results to using all the features, where the differences across datasets and (U)ser vs. (R)eview ranking tasks are insignificant. These findings are in agreement with those in [20], which found that their behavioral features performed very well, whereas the linguistic features were not as effective.

3.4 SpLite performance

In light of our analysis results, we aim to design a "light" version of SpEagle that is computationally more efficient. Our analyses suggest that (1) review priors alone are the most effective, and achieve comparable performance to using priors for all user, review, and product nodes, and that (2) behavioral features are superior to text-based features.

Table 8: $NDCG@k$ performance comparison of SpEagle vs. SpLite (with 1% supervision on all datasets).

	User Ranking						Review Ranking					
	YelpChi		YelpNYC		YelpZip		YelpChi		YelpNYC		YelpZip	
k	Sp'le	SpLite	Sp'le	SpLite	Sp'le	SpLite	Sp'le	SpLite	Sp'le	SpLite	Sp'le	SpLite
100	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9354	0.9334	0.9694	0.9651	0.9219	0.9377
200	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.8469	0.8007	0.9665	0.9595	0.9200	0.9379
300	1.0000	0.9995	1.0000	1.0000	0.9997	1.0000	0.7373	0.6986	0.9597	0.9584	0.9216	0.9377
400	0.9645	0.9589	1.0000	1.0000	0.9998	1.0000	0.6682	0.6397	0.9615	0.9571	0.9248	0.9360
500	0.8841	0.8677	1.0000	1.0000	0.9998	1.0000	0.6255	0.6103	0.9610	0.9529	0.9234	0.9276
600	0.8205	0.8107	1.0000	1.0000	0.9998	1.0000	0.6089	0.5740	0.9620	0.9432	0.9236	0.9121
700	0.7731	0.7650	1.0000	1.0000	0.9999	1.0000	0.5864	0.5556	0.9552	0.8925	0.9240	0.9021
800	0.7416	0.7279	1.0000	1.0000	0.9999	1.0000	0.5587	0.5317	0.9179	0.8351	0.9199	0.8977
900	0.7157	0.6980	1.0000	1.0000	0.9999	1.0000	0.5458	0.5279	0.8775	0.7923	0.9138	0.8899
1000	0.6803	0.6670	1.0000	1.0000	0.9999	1.0000	0.5361	0.5218	0.8463	0.7577	0.9052	0.8810

The computationally most demanding component of SpEagle is feature extraction (network inference is only linear-time in number of edges in the graph [31]). Armed with the above conclusions, our goal is then to identify a few *behavioral* features for only the *review* nodes to be used in estimating priors fast. The rest of the priors, i.e., those for user and product nodes, are to be set to unbiased.

Figure 5 shows the AP performance of SpEagle (R) (as discussed in Table 7) for the review ranking task on all three datasets, when all the review features (blue bar), individual review features (green bars), as well as pairs of behavioral features (red bars, only 3 shown) are used. We find that while there exists no single feature that produces high performance across all datasets, using only two behavioral features often yields similar performance to using all.

We design SpLite to utilize only the RD and EXT features for estimating priors for only the review nodes. The rest are set to unbiased. Table 8 compares SpEagle and SpLite under 1% labeled data across all datasets for both ranking tasks, and Figure 6 illustrates the running times. Notice that SpLite reduces the feature extraction and hence prior estimation overhead significantly, while yielding quite comparable performance to SpEagle.

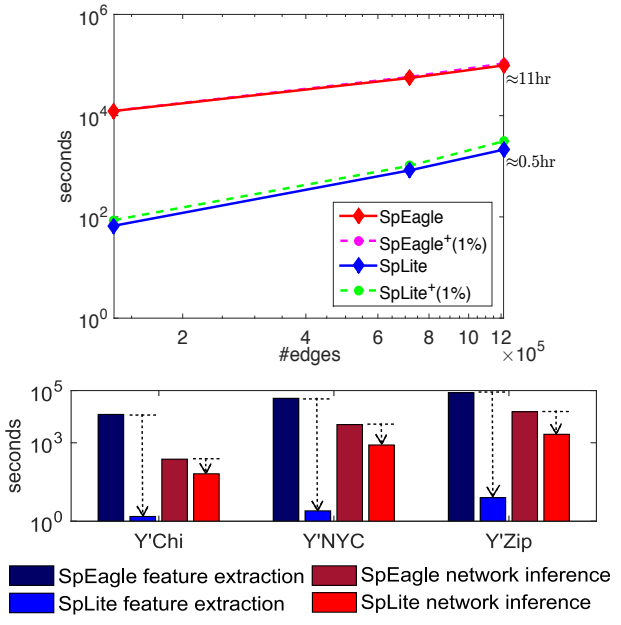


Figure 6: (top) Total running time of SpEagle vs. SpLite, (bottom) Break-down of runtime: feature extraction and network inference, for all datasets.

4. RELATED WORK

Opinion spam is one of the new forms of Web-based spam, and has been the focus of academic research in the last 7-8 years. We organize the various approaches to this problem into three groups: behavior-, language-, and graph-based.

Behavior-based approaches. The approaches in this category often leverage indicative features of spam extracted from the metadata associated with user behavior (e.g., rating distribution), review content (e.g., number of capital letters), and product profile (e.g., brand and price). The seminal work by Jindal and Liu [9] use supervised learning based on 36 such features on a (pseudo) ground truth dataset, constructed by labeling the duplicate reviews in an Amazon dataset as fake reviews. Li *et al.* [13] train semi-supervised models, and use the two views from reviews and users under a co-training framework to spot fake reviews. Jindal *et al.* [10] propose rule-based discovery of unusual patterns in review data associated with the rating and brand distribution of a user's reviews. Other work that study rating based behavior of users include [7] and [15]. More recently, Mukherjee *et al.* [18] utilize reviewing behaviors of users in an unsupervised Bayesian inference framework to detect opinion spammers. Xie *et al.* [28] monitor temporal behavior of products by tracking their average rating, review count, and ratio of singleton reviewers, to spot suspicious single-time reviewers. Those spammers are particularly challenging to detect, as they provide only a single review. Besides detecting individual spammers, there has also been work on identifying spammer groups through group-level behavioral indicators of spam [19, 29].

Language-based approaches. Methods in this category focus on the characteristics of language that the opinion spammers use and how it differs from the language used in genuine reviews. This line of work is also related to studies in deception [21]. Ott *et al.* [22] learn supervised models to detect deceptive reviews based on linguistic features of reviews as well as features borrowed from studies in psychology. Amazon Mechanical Turk has been employed to crowdsource fake reviews by paying anonymous online users to write fake hotel reviews. Feng *et al.* [6] investigate syntactic stylometry for deception detection, and show that features derived from context-free-grammar parse trees improve performance over shallow lexico-syntactic features.

An investigation by Mukherjee *et al.* [20] analyzed the effectiveness of linguistic and behavioral clues on a Yelp dataset with filtered and recommended reviews, and found that linguistic features are not as effective and that Yelp's filter might be using a behavioral based approach.

Graph-based approaches. Wang *et al.* [26] consider the user-review-product network and define scores for trustiness of users, honesty of reviews, and reliability of products. These scores are formulated in terms of one another, and are computed by an iterative algorithm similar to HITS [12]. Akoglu *et al.* [1] proposed a detection framework based on Markov Random Field (MRF) models on the signed bipartite network of users and products, which are connected through positive or negative review relations (signed edges). MRFs have also been utilized in [5] to model user-user relations that capture burst-membership, and in [29] to model user-user collusion relations as well as user-attribute ownership relations. Li *et al.* [14] construct a user-IP-review graph to relate reviews that are written by the same users and from the same IPs. All of these approaches model the fake review(er) detection problem as a collective classification task on these networks, and employ algorithms such as Loopy Belief Propagation (LBP) [31] or Iterative Classification Algorithm (ICA) [23] for inference. Finally, Jiang *et al.* [8] and most recently Ye and Akoglu [30] proposed graph-based methods to identify group spammers solely based on their abnormal network footprints.

5. CONCLUSION

In this work, we proposed a new holistic framework called SpEagle that exploits both relational data (user–review–product graph) and metadata (behavioral and text data) *collectively* to detect suspicious users and reviews, as well as products targeted by spam. Our main contributions are:

- SpEagle employs a review-network-based classification task which accepts prior knowledge on the class distribution of the nodes, estimated from metadata.
- SpEagle works in an unsupervised fashion, but can easily leverage labels (if available). As such, we introduce a *semi-supervised* version called SpEagle$^+$, which improves performance significantly.
- We further design a *light* version of SpEagle called SpLite which uses a very small set of review features as prior information, providing significant speed-up.

We evaluated our method on three real-world datasets with labeled reviews (filtered vs. recommended), as collected from Yelp.com. As such, we provide the largest scale quantitative evaluation results on opinion spam detection to date. Our results show that SpEagle is superior to several baselines and state-of-the-art techniques. Our source code and datasets with ground truth can be found at `http://shebuti.com/collective-opinion-spam-detection/`.

Acknowledgments

The authors thank the anonymous reviewers for their useful comments. This material is based upon work supported by the ARO Young Investigator Program Contract No. W911NF-14-1-0029, NSF CAREER 1452425 and IIS 1408287, a Facebook Faculty Gift, an R&D grant from Northrop Grumman Aerospace Systems, and Stony Brook University Office of Vice President for Research. Any conclusions expressed in this material are of the authors' and do not necessarily reflect the views of the funding parties.

6. REFERENCES

[1] L. Akoglu, R. Chandy, and C. Faloutsos. Opinion fraud detection in online reviews by network effects. In *ICWSM*, 2013.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 2002.

[3] J. D'onfro. *A Whopping 20% Of Yelp Reviews Are Fake*, 2013. `http://read.bi/1MO3jxl`.

[4] C. Elkan. The foundations of cost-sensitive learning. In *IJCAI*, pages 973–978. Morgan Kaufmann, 2001.

[5] G. Fei, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh. Exploiting burstiness in reviews for review spammer detection. In *ICWSM*, 2013.

[6] S. Feng, R. Banerjee, and Y. Choi. Syntactic stylometry for deception detection. In *ACL*, 2012.

[7] S. Feng, L. Xing, A. Gogar, and Y. Choi. Distributional footprints of deceptive product reviews. In *ICWSM*, 2012.

[8] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang. Catchsync: catching synchronized behavior in large directed graphs. In *KDD*, pages 941–950, 2014.

[9] N. Jindal and B. Liu. Opinion spam and analysis. In *WSDM*, pages 219–230, 2008.

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

[11] R. Kindermann and J. L. Snell. *Markov Random Fields and Their Applications*. 1980.

[12] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[13] F. Li, M. Huang, Y. Yang, and X. Zhu. Learning to identify review spam. In *IJCAI*, 2011.

[14] H. Li, Z. Chen, B. Liu, X. Wei, and J. Shao. Spotting fake reviews via collective PU learning. In *ICDM*, 2014.

[15] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In *CIKM*, pages 939–948, 2010.

[16] M. Luca and G. Zervas. Fake it till you make it: Reputation, competition, and Yelp review fraud. Working Papers 14-006, Harvard Business School, 2013.

[17] P. V. Marsden. Homogeneity in confiding relations. *Social Networks*, 10(1):57–76, Mar. 1988.

[18] A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, and R. Ghosh. Spotting opinion spammers using behavioral footprints. In *KDD*, 2013.

[19] A. Mukherjee, B. Liu, and N. S. Glance. Spotting fake reviewer groups in consumer reviews. In *WWW*, 2012.

[20] A. Mukherjee, V. Venkataraman, B. Liu, and N. S. Glance. What Yelp fake review filter might be doing? In *ICWSM*, 2013.

[21] M. Ott, C. Cardie, and J. T. Hancock. Estimating the prevalence of deception in online review communities. In *WWW*, pages 201–210, 2012.

[22] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *ACL*, pages 309–319, 2011.

[23] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[24] D. Streitfeld. *Best Book Reviews Money Can Buy*, 2012. `http://nyti.ms/1cvg5b1`.

[25] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme. Cost-sensitive learning methods for imbalanced data. In *IJCNN*, pages 1–8. IEEE, 2010.

[26] G. Wang, S. Xie, B. Liu, and P. S. Yu. Review graph based online store review spammer detection. In *ICDM*, 2011.

[27] K. Weise. *A Lie Detector Test for Online Reviewers*, 2011. `http://bloom.bg/1KAxzhK`.

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

[29] C. Xu, J. Zhang, K. Chang, and C. Long. Uncovering collusive spammers in Chinese review websites. In *CIKM*, pages 979–988, 2013.

[30] J. Ye and L. Akoglu. Discovering opinion spammer groups by network footprints. In *ECML/PKDD*, 2015.

[31] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding BP and its generalizations. In *Exploring AI in the new millennium*. 2003.