Opinion Fraud Detection in Online Reviews by Network Effects

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

Rishi Chandy
Carnegie Mellon University
School of Computer Science
rishic@cs.cmu.edu

Christos Faloutsos
Carnegie Mellon University
School of Computer Science
christos@cs.cmu.edu

Abstract

User-generated online reviews can play a significant role in the success of retail products, hotels, restaurants, etc. However, review systems are often targeted by opinion spammers who seek to distort the perceived quality of a product by creating fraudulent reviews. We propose a fast and effective framework, FRAUDEAGLE, for spotting fraudsters and fake reviews in online review datasets. Our method has several advantages: (1) it exploits the *network effect* among reviewers and products, unlike the vast majority of existing methods that focus on review text or behavioral analysis, (2) it consists of two complementary steps; *scoring* users and reviews for fraud detection, and *grouping* for visualization and sensemaking, (3) it operates in a completely *unsupervised* fashion requiring no labeled data, while still incorporating side information if available, and (4) it is *scalable* to large datasets as its run time grows linearly with network size. We demonstrate the effectiveness of our framework on synthetic and real datasets; where FRAUDEAGLE successfully reveals fraud-bots in a large online app review database.

Introduction

The Web has greatly enhanced the way people perform certain activities (e.g. shopping), find information, and interact with others. Today many people read/write reviews on merchant sites, blogs, forums, and social media before/after they purchase products or services. Examples include restaurant reviews on Yelp, product reviews on Amazon, hotel reviews on TripAdvisor, and many others. Such user-generated content contains rich information about user experiences and opinions, which allow future potential customers to make better decisions about spending their money, and also help merchants improve their products, services, and marketing.

Since online reviews can directly influence customer purchase decisions, they are crucial to the success of businesses. While positive reviews with high ratings can yield financial gains, negative reviews can damage reputation and cause monetary loss. This effect is magnified as the information spreads through the Web (Hitlin 2003; Mendoza, Poblete, and Castillo 2010). As a result, online review systems are attractive targets for *opinion fraud*. Opinion fraud involves reviewers (often paid) writing bogus reviews (Kost May 2012; Streitfeld August 2011). These spam reviews come in two flavors: *defaming*-spam which untruthfully vilifies, or *hype*-spam that deceitfully promotes the target product.

The opinion fraud detection problem is to spot the fake reviews in online sites, given all the reviews on the site, and for each review, its text, its author, the product it was written for, timestamp of posting, and its star-rating. Typically no user profile information is available (or is self-declared and cannot be trusted), while more side information for products (e.g. price, brand), and for reviews (e.g. number of (helpful) feedbacks) could be available depending on the site.

Detecting opinion fraud, as defined above, is a non-trivial and challenging problem. Fake reviews are often written by experienced professionals who are paid to write high quality, believable reviews. As a result, it is difficult for an average potential customer to differentiate bogus reviews from truthful ones, just by looking at individual reviews text(Ott et al. 2011). As such, manual labeling of reviews is hard and ground truth information is often unavailable, which makes training supervised models less attractive for this problem.

Summary of previous work. Previous attempts at solving the problem use several heuristics, such as duplicated reviews (Jindal and Liu 2008), or acquire bogus reviews from non-experts (Ott et al. 2011), to generate pseudo-ground truth, or a reference dataset. This data is then used for learning classification models together with carefully engineered features. One downside of such techniques is that they do not generalize: one needs to collect new data and train a new model for review data from a different domain, e.g., hotel vs. restaurant reviews. Moreover feature selection becomes a tedious sub-problem, as datasets from different domains might exhibit different characteristics. Other feature-based proposals include (Lim et al. 2010; Mukherjee, Liu, and Glance 2012).

A large body of work on fraud detection relies on review text information (Jindal and Liu 2008; Ott et al. 2011; Feng, Banerjee, and Choi 2012) or behavioral evidence (Lim et al. 2010; Xie et al. 2012; Feng et al. 2012), and ignore the connectivity structure of review data. On the other hand, the network of reviewers and products contains rich information that implicitly represents correlations among these entities. The review network is also invaluable for detecting teams of fraudsters that operate collaboratively on targeted products.

Our contributions. In this work we propose an unsupervised, general, and network-based framework, FRAUDEAGLE, to tackle the opinion fraud detection problem in online review data. The review network successfully captures the correlations of labels among users and products, e.g. fraudsters are mostly linked to good (bad) products with negative (positive) fake reviews, and vice versa for honest users. As such, the network edges are *signed* by sentiment. We build an iterative, propagation-based algorithm that exploits the network structure and the long-range correlations to infer the class labels of users, products, and reviews. A second step involves analysis and summarization of results. For generality, we do not use review text information, but only the positive or negative sentiment of the reviews. As such, our method can be applied to any type of review data and is complementary to existing approaches.

We summarize our main contributions as follows.

- We formulate the opinion fraud detection problem as a network classification task on signed networks.
- We propose a novel framework that (1) employs a propagation-based algorithm that exploits the *network effect* for classification, and (2) provides a summary and analysis of results.
- The proposed method is (a) *general*; which can be applied to all types of review networks, (b) *unsupervised*; that can work with no prior knowledge, and (c) *scalable*; with its run time linear in network size.
- We evaluate our method compared to alternative methods on synthetic and real online app review data, where we successfully spot fraudulent users and bots that unfairly distort product ratings.

The rest of the paper is organized as follows: survey, proposed framework, competitors, evaluation, and conclusion.

Related Work

Much of the previous work in opinion fraud focuses on review text content, behavioral analysis, and supervised methods. (Jindal and Liu 2008) identified opinion spam by detecting exact text duplicates in an Amazon.com dataset, while (Ott et al. 2011) crowd-sourced deceptive reviews in order to create a highly accurate classifier based on n-grams. Several studies tried to engineer better features to improve classifier performance. (Li et al. 2011) uses sentiment scores, product brand, and reviewer profile attributes to train classifiers. Other work has computed scores based on behavioral heuristics, such as rating deviation by (Lim et al. 2010), and frequent itemset mining to find fraudulent reviewer groups by (Mukherjee, Liu, and Glance 2012).

Unfortunately, these methods are not generalizable: the models need re-training to account for differences between problem domains, such as book reviews versus movie reviews. Moreover, the features might not be consistent even for datasets within the same domain, depending on the dataset source. Consequently, feature extraction becomes a time-consuming yet pivotal sub-problem with attributes varying across domains.

Another group of work mines behavioral patterns in review data. (Jindal, Liu, and Lim 2010) finds unexpected rules to highlight anomalies, and (Xie et al. 2012; Feng et al. 2012; Feng, Banerjee, and Choi 2012) respectively study temporal review behavior, rating distributions, and syntactic stylometry.

On the other hand, methods that account for the *network* of reviewers, reviews, and products can more elegantly encapsulate structural signals that go beyond the review content and simple heuristics, thus generalizing across domains. (Wang et al. 2011) proposed the first (and to the authors' knowledge the only) review graph-based method and a simple yet effective algorithm to compute scores for each node. We propose an even more flexible method that exploits the network effects; being able to incorporate side information, is based on the rigorous theoretical foundation of belief propagation, and is linearly scalable.

Network effects have been exploited in securities fraud (Neville et al. 2005), accounting fraud (McGlohon et al. 2009), and auction fraud (Pandit et al. 2007) detection. However, none of these proposals are applicable to opinion fraud detection, as their problem domains do not involve ratings and sentiment spam. Related to ratings on products, work on recommender systems (Koren 2009; Menon and Elkan 2011), aim for best prediction of future user ratings, but do not address the fraud problem.

Proposed FRAUDEAGLE Framework

In this section we first introduce the opinion fraud problem with a toy example. After we give notation and formal definition, we present our problem formulation and the proposed algorithm, along with results on our didactic example. We conclude this section with a discussion on how to analyze and summarize results.

Problem Description and Toy Example

Simply put, we consider the problem of spotting fraudulent reviewers, and consequently spotting fake reviews in online review datasets.

The online review datasets mainly consist of a set of users (also called customers, reviewers), a set of products (e.g., hotels, restaurants, etc.), and the reviews. Each review is written from a particular user to a particular product, and contains a star-rating, often an integer from 1 to 5. As such, a review dataset can be represented as a bipartite network. In this network, user nodes are connected to product nodes, in which the links represent the "reviewed" relationships and each link is associated with the review rating.

The objects in the review network, i.e. the users, products, and reviews, can be grouped into certain classes. In this paper, we consider two classes for each object type: products are either *good* or *bad* quality, users are either *honest* or *fraud*, and finally reviews are either *real* or *fake*.

Intuitively, a product is *good* (*bad*) if it most of the time receives many positive (negative) reviews from *honest* users. Similarly, a user is *honest* (*fraud*) if s/he mostly writes positive (negative) reviews to *good* products, and negative (positive) reviews to *bad* products. In other words, a user is *fraud* if s/he is trying to promote a set of target *bad* products (hype-spam), and/or damage the reputation of a set of target *good* products (defaming-spam). All the reviews of *honest* users can safely be regarded as *real*. In an ideal setting, all the

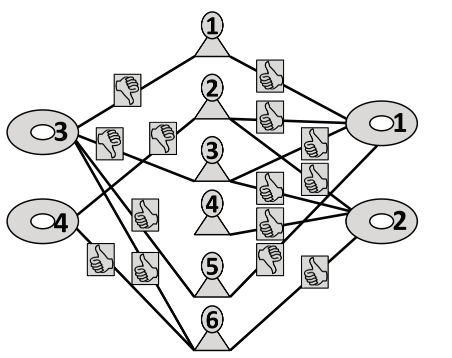


Figure 1: A toy review network of 6 users and 4 products. The sentiment of reviews are observed, depicted as signed edges: thumbs-up for $+$ and thumbs-down for $-$.

reviews of the *fraud* users can also be thought of as *fake*. However, in reality fraudsters could also write *real*istic reviews, trying to hide their otherwise fraudulent activity. All in all, notice that the class labels of the interconnected objects in the review data are strongly correlated as they are described in terms of one another. As a result, it is natural to think of a review dataset as a network in order to attack the opinion fraud problem coherently.

Given a user-product review network as described above, the task is to assign each object with a class label that best describes the observed data, i.e. the network structure and ratings. In this task, we need to infer from review text whether it is of positive or negative sentiment. While there exists prior work on sentiment classification using text information (Pang, Lee, and Vaithyanathan 2002), we take the review rating as a highly correlated indicator for sentiment. More specifically, we consider a *signed* network in which each network link (i.e. review) is marked as positive if its rating is above a threshold, and as negative otherwise.

We show an example in Figure 1. Users 1-4 are honest while 5-6 are fraud, and products 1-2 are good-quality while 3-4 are bad. The network structure and the sentiment (sign) of edges are observed. The fraudsters write fake reviews for the products except the $+$ review from user 6 to product 2; it is posted to camouflage this user's otherwise fraudulent activity. The goal is to develop a method to automatically label the users, products, and reviews in the network.

Notice that in our problem setting we do not use the review text and other side information such as timestamp, product brand, etc. We can, however, integrate such additional information into our formulation as prior information, as we will discuss in the proceeding. As such, our proposed method is highly complementary to existing techniques that use language technologies or behavioral analysis.

Next, we introduce the notation and define the classification problem formally.

Problem Formulation

Notation. We are given a signed review network $G_s = (\mathcal{V}, \mathcal{E})$, in which a set of user nodes $\mathcal{U} = \{u_1, \ldots, u_n\}$ and a set of product nodes $\mathcal{P} = \{p_1, \ldots, p_m\}$ are connected with signed links $e(u_i, p_j, s) \in \mathcal{E}$, $s \in \{+, -\}$, and $\mathcal{U} \cup \mathcal{P} = \mathcal{V}$. A neighborhood function $\mathcal{N}$, $\mathcal{N}_{u_i} \subseteq \mathcal{P}$ and $\mathcal{N}_{p_j} \subseteq \mathcal{U}$, describes the underlying bipartite network structure.

Each node in $\mathcal{V}$ and each edge in $\mathcal{E}$ is a random variable that takes a value from an appropriate label domain; in our case, $\mathcal{L}_{\mathcal{U}} = \{honest, fraud\}$, $\mathcal{L}_{\mathcal{P}} = \{good, bad\}$, and $\mathcal{L}_{\mathcal{E}} = \{real, fake\}$. In this classification task, let $\mathcal{Y}^{\mathcal{V}} = \mathcal{Y}^{\mathcal{U}} \cup \mathcal{Y}^{\mathcal{P}}$ and $\mathcal{Y}^{\mathcal{E}}$ respectively denote the nodes and edges the values of which need to be assigned, and let y_i refer to Y_i's label.

Objective formulation. We next define our objective function we seek to optimize for the above classification task. We propose to use an objective function that utilizes pairwise Markov Random Fields (pMRF) (Kindermann and Snell 1980), which we adapt to our problem setting as follows.

Let $G_s = (\mathcal{V}, \mathcal{E})$ denote a signed network of random variables as before, where $\mathcal{V}$ consists of the unobserved variables $\mathcal{Y}$ which need to be assigned values from label set $\mathcal{L} = \mathcal{L}_{\mathcal{U}} \cup \mathcal{L}_{\mathcal{P}}$. Let Ψ denote a set of clique potentials that consists of two types of functions:

- For each $Y_i \in \mathcal{Y}^{\mathcal{U}}$ and $Y_j \in \mathcal{Y}^{\mathcal{P}}$, $\psi_i \in \Psi$ is a *prior* mapping $\psi_i^{\mathcal{U}} : \mathcal{L}_{\mathcal{U}} \rightarrow \mathbb{R}_{\geq 0}$, and $\psi_j^{\mathcal{P}} : \mathcal{L}_{\mathcal{P}} \rightarrow \mathbb{R}_{\geq 0}$, respectively, where $\mathbb{R}_{\geq 0}$ is non-negative real numbers.
- For each $e(Y_i^{\mathcal{U}}, Y_j^{\mathcal{P}}, s) \in \mathcal{E}$, $\psi_{ij}^s \in \Psi$ is a *compatibility* mapping $\psi_{ij}^s : \mathcal{L}_{\mathcal{U}} \times \mathcal{L}_{\mathcal{P}} \rightarrow \mathbb{R}_{\geq 0}$.

Given an assignment $\mathbf{y}$ to all the unobserved variables $\mathcal{Y}^{\mathcal{V}}$ and $\mathbf{x}$ to observed ones $\mathcal{X}^{\mathcal{V}}$ (variables with known values), our objective is associated with the probability distribution

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{Y_i \in \mathcal{Y}^{\mathcal{V}}} \psi_i(y_i) \prod_{e(Y_i^{\mathcal{U}}, Y_j^{\mathcal{P}}, s) \in \mathcal{E}} \psi_{ij}^s(y_i, y_j) \tag{1}$$

where $Z(\mathbf{x})$ is the normalization function.

Problem definition. Now we can define the opinion fraud detection problem more formally.

Given
- a bipartite network $G_s = (\mathcal{V}, \mathcal{E})$ of users and products connected with *signed* edges,
- *prior* knowledge (probabilities) of network objects belonging to each class, and
- *compatibility* of two objects with a given pair of labels being connected;

Classify the network objects $Y_i \in \mathcal{Y} = \mathcal{Y}^{\mathcal{V}} \cup \mathcal{Y}^{\mathcal{E}}$, into one of two respective classes; $\mathcal{L}_{\mathcal{U}} = \{honest, fraud\}$, $\mathcal{L}_{\mathcal{P}} = \{good, bad\}$, and $\mathcal{L}_{\mathcal{E}} = \{real, fake\}$, where the assignments y_i maximize the objective probability in Equation (1).

We give details on priors and compatibilities in the context of our algorithm, which we discuss in the next section.

Having formulated the objective function for our problem, we are ready to introduce the two major steps of our proposed FRAUDEAGLE framework: (1) scoring for fraud detection, and (2) grouping for analysis and sensemaking. An overview of FRAUDEAGLE is given in Outline 1.

Step 1. FraudEagle Scoring

Finding the best assignments to unobserved variables in our objective function, as given in Equation (1), is the inference problem. In general, exact inference is known to be an NP-hard problem, therefore we use a computationally tractable (in fact linearly scalable with network size) approximate inference algorithm called Loopy Belief Propagation (LBP). LBP is based on iterative message passing, and while it is provably correct only for certain cases, it has been shown to perform extremely well for a wide variety of applications in the real world (Yedidia, Freeman, and Weiss 2003).

In the following we propose a new algorithm that extends LBP in order to handle signed networks. At convergence, we use the maximum likelihood label probabilities for scoring.

signed Inference Algorithm (sIA) The inference algorithm applied on a *signed* bipartite network can be concisely expressed as the following equations:

$$m_{i\to j}(y_j) = \alpha_1 \sum_{y_i \in \mathcal{L}^{\mathcal{U}}} \psi^s_{ij}(y_i, y_j)\ \psi^{\mathcal{U}}_i(y_i) \prod_{Y_k \in \mathcal{N}_i \cap \mathcal{Y}^{\mathcal{P}} \setminus Y_j} m_{k\to i}(y_i),\ \ \forall y_j \in \mathcal{L}^{\mathcal{P}} \quad (2)$$

$$b_i(y_i) = \alpha_2\ \psi^{\mathcal{U}}_i(y_i) \prod_{Y_j \in \mathcal{N}_i \cap \mathcal{Y}^{\mathcal{P}}} m_{j\to i}(y_i), \forall y_i \in \mathcal{L}^{\mathcal{U}} \quad (3)$$

where $m_{i\to j}$ is a *message* sent by user Y_i to product Y_j (a similar equation can be written for messages from products to users), and $b_i(y_i)$ denotes the *belief* of user i having label y_i (again, a similar equation can be written for beliefs of products). α's are the normalization constants, which respectively ensure that each message and each set of marginal probabilities sum to 1.

The algorithm (Outline 1 Line 6-25) proceeds by making each set of $Y_i \in \mathcal{Y}^{\mathcal{U}}$ and $Y_j \in \mathcal{Y}^{\mathcal{P}}$ alternately communicate messages with its neighbors in an iterative fashion until the messages stabilize, i.e. convergence. After they stabilize, we calculate the marginal probabilities; say of assigning Y_i with label y_i by computing the final belief $b_i(y_i)$. Although convergence is not theoretically guaranteed, the LBP has been shown to converge to beliefs within a small threshold fairly quickly with accurate results.

Priors. To completely define the sIA algorithm, we need to instantiate the clique potential functions Ψ. The *prior* beliefs $\psi^{\mathcal{U}}_i$ and $\psi^{\mathcal{P}}_j$, respectively of users and products can be suitably initialized if there is any prior knowledge of the objects (e.g. Angry Birds is a good product). These priors could also be estimated based on available side information such as review text (e.g. using text-feature classifiers (Ott et al. 2011)), timeseries activity and other behavioral features (e.g. using behavioral analysis (Jindal and Liu 2008)), etc. As such, our method is general and complementary to existing feature-based methods. In case there is no prior knowledge available, each node is initialized equally likely to have any of the possible labels.

Compatibility matrices. Finally, we define the *compatibility* potentials. These can be thought of as matrices with entries $\psi^s_{ij}(y_i, y_j)$, which gives the likelihood of a node having label y_i given that it has a neighbor with label y_j. Recall that in our adaptation of belief propagation to the fraud detection problem, these potentials are dependent on and are thus indexed by review sentiment. A sample instantiation of the *compatibility* matrices is shown in Table 1. This instantiation is based on the following intuition: honest users tend to write positive reviews to good products and negative ones to bad products, honest users could also have complaints about good products and, although with much less affinity, might "like" the bad products depending on their preferences, fraudsters tend to write positive reviews to bad products to boost their ratings and negative reviews to good products to underrate them, fraudsters could also behave like normal honest users with reverse activity in order to camouflage themselves. Note that automatically learning the compatibility potentials among classes in the existence of labeled ground truth data is a valuable future direction.

s: +	**Products**	
Users	Good	Bad
Honest	1-ϵ	ϵ
Fraud	2ϵ	1-2ϵ

s: -	**Products**	
Users	Good	Bad
Honest	ϵ	1-ϵ
Fraud	1-2ϵ	2ϵ

Table 1: Instantiation of the sentiment-based compatibility matrices for online review fraud detection. Entry $(y_i, y_j)^s$ denotes the compatibility of a product node having label y_j while having a user node neighbor with label y_i, given the review from i to j has sentiment s, for small ϵ.

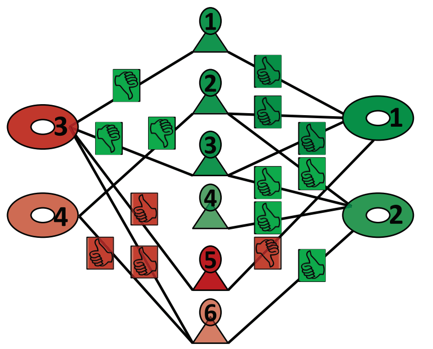


Figure 2: Classification results of sIA on toy network in Figure 1. Dark red (green): fraud (honest) users, also fake (real) for reviews, and bad (good) for products. Figure is best viewed in color.

Assigning Scores After sIA converges, the class labels of users and products can be inferred by their final belief vectors. Each set of marginal beliefs sum to 1, i.e. $\sum_{y_i} b_i(y_i) = 1$, and thus can be treated as class probabilities for users, and similarly for products (Line 27, 28).

Recall that we also want to label the network edges $\mathcal{Y}^{\mathcal{E}}$ (i.e. reviews) as fake or real. To do so, we simply take the converged messages from products to users, i.e. $m_{j\to i}(y_i)$, as the final beliefs on reviews. Each message also sum to 1, i.e. $\sum_{y_i} m_{j\to i}(y_i) = 1$, and thus can be taken as the class probabilities for the reviews (Line 29).

Outline 1: FRAUDEAGLE FRAMEWORK

```
Step 1. Scoring
    signedInferenceAlgorithm()
Step 2. Grouping
    findGroups()
--------------------------------------------------
Step 1. signedInferenceAlgorithm()
Input: Bipartite network like Figure 1 of users, products, and review ratings
Output: Score for every user ($fraud$), product ($bad$), review ($fake$)
foreach $e(Y_i, Y_j, s) \in \mathcal{E}$ s.t. $Y_i, Y_j \in \mathcal{Y}^{\mathcal{V}}$ do // initialize
    foreach $y_i \in \mathcal{L}^{\mathcal{U}}$, $y_j \in \mathcal{L}^{\mathcal{P}}$ do
        $m_{i \to j}(y_j) \leftarrow 1$, $\phi_i^{\mathcal{U}}(y_i) \leftarrow |\mathcal{L}^{\mathcal{U}}|$
        $m_{j \to i}(y_i) \leftarrow 1$, $\phi_j^{\mathcal{P}}(y_j) \leftarrow |\mathcal{L}^{\mathcal{P}}|$
repeat // perform message propagation
    // update messages from users to products
    foreach $e(Y_i, Y_j, s) \in \mathcal{E}$ s.t. $Y_i, Y_j \in \mathcal{Y}^{\mathcal{V}}$ do
        foreach $y_j \in \mathcal{L}^{\mathcal{P}}$ do
            $m_{i \to j}(y_j) \leftarrow \alpha_1 \sum_{y_i \in \mathcal{L}^{\mathcal{U}}} \psi_{ij}^{s}(y_i, y_j) \phi_i^{\mathcal{U}}(y_i) \prod_{Y_k \in \mathcal{N}_i \cap \mathcal{Y}^{\mathcal{P}} \setminus Y_j} m_{k \to i}(y_i)$
    // update messages from products to users
    foreach $e(Y_i, Y_j, s) \in \mathcal{E}$ s.t. $Y_i, Y_j \in \mathcal{Y}^{\mathcal{V}}$ do
        foreach $y_i \in \mathcal{L}^{\mathcal{U}}$ do
            $m_{j \to i}(y_i) \leftarrow \alpha_3 \sum_{y_j \in \mathcal{L}^{\mathcal{P}}} \psi_{ij}^{s}(y_i, y_j) \phi_j^{\mathcal{P}}(y_j) \prod_{Y_k \in \mathcal{N}_j \cap \mathcal{Y}^{\mathcal{U}} \setminus Y_i} m_{k \to j}(y_j)$
until all messages $m(y)$ stop changing
foreach $Y_i, Y_j \in \mathcal{Y}^{\mathcal{V}}$ do // compute beliefs
    foreach $y_i \in \mathcal{L}^{\mathcal{U}}$, $y_j \in \mathcal{L}^{\mathcal{P}}$ do
        $b_i(y_i) = \alpha_2 \phi_i^{\mathcal{U}}(y_i) \prod_{Y_j \in \mathcal{N}_i \cap \mathcal{Y}^{\mathcal{P}}} m_{j \to i}(y_i)$
        $b_j(y_j) = \alpha_4 \phi_j^{\mathcal{P}}(y_j) \prod_{Y_i \in \mathcal{N}_j \cap \mathcal{Y}^{\mathcal{U}}} m_{i \to j}(y_j)$
// assign scores
score$_i(fraud) = b_i(y_i : fraud), \forall i \in \mathcal{U}$
score$_j(bad) = b_j(y_j : bad), \forall j \in \mathcal{P}$
score$_{e(i,j)}(fake) = m_{j \to i}(y_i : fraud), \forall e(i,j) \in \mathcal{E}$
--------------------------------------------------
Step 2. findGroups()
Input: ranked list of users by score from Step 1., number of top users $k$
Output: bot-users and products under attack
$G_{top} \leftarrow$ induced subgraph on top $k$ users and products
cluster $G_{top}$ (we use cross-associations)
return/visualize clusters, bipartite-cores
```

Each such incoming message to user Y_i from each product j that s/he wrote reviews contributes to the probability of Y_i having label y_i and equivalently represents the probability of his/her review for product j having label $y_i^{\mathcal{E}}$. More precisely, $y_i^{\mathcal{U}}$:fraud $\equiv y_i^{\mathcal{E}}$:fake and $y_i^{\mathcal{U}}$:honest $\equiv y_i^{\mathcal{E}}$:real. This helps us to differentiate the fraudulent reviews of a fraudster from his/her realistic reviews that s/he wrote to hide his/her fraudulent activity.

We show the results on our running toy example in Figure 2. The algorithm correctly classifies all the objects, including the (camouflage) positive review from (fraud) user 6 to (good) product 2 which is classified as real.

Step 2. FRAUDEAGLE Grouping

Marginal class probabilities over users, products, and reviews, i.e. scores, enable us to order each set of them in a ranked list. While a rank list of, say, users with respect to being a fraudster is a valuable resource, it does not put the top such users in context with the products that they rated.

In order to help with visualization, summarization, and further sensemaking of fraudsters we project the top users back on the review graph, and obtain the induced subgraph including these users along with the union of products that they rated. The idea is to partition this subgraph into clusters to gain more insight about how they are organized in the network. For partitioning, one can use any graph clustering algorithm. We employ the cross-associations (CA) clustering algorithm (Chakrabarti et al. 2004) on the adjacency matrix of the induced graph. The CA algorithm performs clustering by finding a permutation of the rows (users) and columns (products) of the matrix such that the resulting matrix contains homogeneous blocks (defined by the clustering), where dense blocks correspond to near-bipartite cores (e.g., a team of users attacking on a target set of products).

Given the top-scoring users, the grouping step essentially merges evidence further; by revealing attack and target groups as well as providing summarization by clustering.

Our Adapted Competitors

In this section we describe two alternative methods that we developed as baselines. We present them as competitors and compare all methods in the experiments.

We modify two algorithms for network classification to handle signed networks for our fraud detection setting: weighted-vote relational classifier (wv-RC) (Macskassy and Provost 2003; Hill, Provost, and Volinsky 2007) and hubs-and-authorities (HITS) algorithm (Kleinberg 1998).

weighted-vote Relational Classifier

The wv-RC is a neighbor-based classifier which estimates class-membership probability of each node as the weighted mean of the class-membership probabilities of its neighbors. In our setting the underlying network is bipartite and the edges are signed. Therefore, the above definition translates to the following equations:

$$\mathrm{Pr}_i^{\mathcal{U}}(hon.) = \frac{1}{Z_i}\left(\sum_{j \in \mathcal{N}_i^+} w_{ij}^{+} \mathrm{Pr}_j^{\mathcal{P}}(good) - \sum_{j \in \mathcal{N}_i^-} w_{ij}^{-} \mathrm{Pr}_j^{\mathcal{P}}(bad)\right)$$

$$\mathrm{Pr}_j^{\mathcal{P}}(good) = \frac{1}{Z_j}\left(\sum_{i \in \mathcal{N}_j^+} w_{ij}^{+} \mathrm{Pr}_i^{\mathcal{U}}(hon.) - \sum_{i \in \mathcal{N}_j^-} w_{ij}^{-} \mathrm{Pr}_i^{\mathcal{U}}(fr.)\right)$$

where $\mathcal{N}^+$ denotes the neighbors of a node that are linked to it by positive weights w^+. In our setting, edge weights are w^+=1 and w^-=−1, as we have positive and negative reviews. Z's are normalization constants, i.e. $Z_i = \sum_{j\in\mathcal{N}_i^+} w_{ij}^+ - \sum_{j\in\mathcal{N}_i^-} w_{ij}^- = |\mathcal{N}_i|$. Finally, $\Pr_i^{\mathcal{U}}(fraud) = 1 - \Pr_i^{\mathcal{U}}(honest)$ and $\Pr_j^{\mathcal{P}}(bad) = 1 - \Pr_j^{\mathcal{P}}(good)$; $\Pr_i^{\mathcal{U}}, \Pr_j^{\mathcal{P}} \in [0,1]$.

We use the above equations to iteratively update class probabilities of all nodes. Nodes with unknown labels are initially assigned class priors $\Pr_i^{\mathcal{U}}(honest) = \Pr_j^{\mathcal{P}}(good) = 0.9$ (note that priors should be set to other than 0.5 at least for some nodes for progress). Due to the loopy nature of propagation convergence is not guaranteed, although in our experiments the probabilities converged within a small threshold of change (ϵ=10^{-4}) from one iteration to the next.

Iterative Honesty-and-Goodness

We also adapt the HITS algorithm (Kleinberg 1998) to compute the *honesty* of users and *goodness* of products. Honesty and goodness values are defined in terms of one another in a mutual recursion. The honesty (goodness) value of a user (product) is computed as the scaled sum of the goodness (honesty) values of products (users) linked to it by positive reviews minus the sum of those linked by negative reviews. We give the corresponding equations as below.

$$H_i^{\mathcal{U}} = f\left(\sum_{j\in\mathcal{N}_i^+} w_{ij}^+ G_j^{\mathcal{P}} + \sum_{j\in\mathcal{N}_i^-} w_{ij}^- G_j^{\mathcal{P}}\right)$$

$$G_j^{\mathcal{P}} = f\left(\sum_{i\in\mathcal{N}_j^+} w_{ij}^+ H_i^{\mathcal{U}} + \sum_{i\in\mathcal{N}_j^-} w_{ij}^- H_i^{\mathcal{U}}\right)$$

where $f(.)$ is the normalizing function $f(x) = \frac{2}{1+exp(x)} - 1$; $H_i^{\mathcal{U}}, G_j^{\mathcal{P}} \in [-1,1]$, and $w^{\pm}$ and $\mathcal{N}^{\pm}$ are defined as before.

We use the above equations to iteratively update the honesty and goodness values of users and products, respectively. Nodes with unknown labels are initially assigned values $H_i^{\mathcal{U}} = G_j^{\mathcal{P}} = \epsilon \approx 0$, i.e. unbiased priors. For convergence, we need to set a maximum number of iterations.

Computing Review Scores

Both wv-RC and the HITS-like algorithm we adapted for our setting compute scores for the nodes in the network, i.e. users and products. To compute the corresponding scores for the edges, i.e. reviews, we follow similar ideas as in (Wang et al. 2011).

For each review, we split all the reviews (in fact their owners) for the same product into two sets: agreement set $\mathcal{U}_a$ contains the users who wrote those reviews with the same sentiment as of current review, and disagreement set $\mathcal{U}_d$ contains those who wrote the ones with opposite sentiment.

Then, the *reliability* score of each review for wv-RC is:

$$R_{ij}^{\mathcal{E}} = \frac{1}{|\mathcal{U}_a \cup \mathcal{U}_d|}\left(\sum_{u\in\mathcal{U}_a} \Pr_u^{\mathcal{U}}(honest) + \sum_{u\in\mathcal{U}_d} \Pr_u^{\mathcal{U}}(fraud)\right)$$

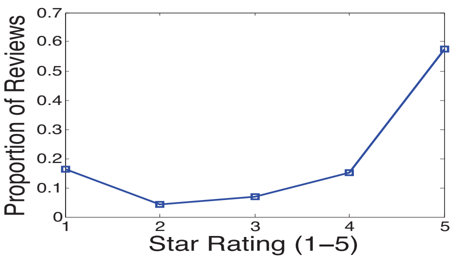


Figure 3: Star rating distribution for the SWM dataset. Notice the upward tilting 'J' shape.

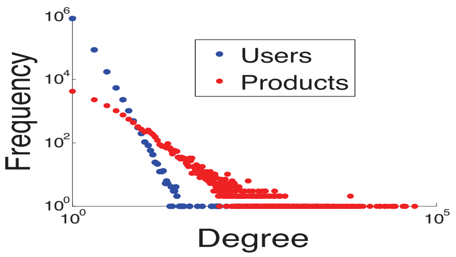


Figure 4: Degree distributions for user and product nodes in the SWM dataset. Notice the power-law like skewness as seen in real networks.

Similarly, each review score for the adapted HITS is:

$$R_{ij}^{\mathcal{E}} = f\left(|G_j^{\mathcal{P}}|\left(\sum_{u\in\mathcal{U}_a} H_u^{\mathcal{U}} - \sum_{u\in\mathcal{U}_d} H_u^{\mathcal{U}}\right)\right)$$

Evaluation and Comparison

In our experiments we used a new dataset, which is a collection of app reviews. We refer to this data as the SoftWare Marketplace (SWM) dataset. In this section, we first describe our data and later present evaluation results on both synthetic and SWM datasets.

SWM Data and Analysis

Data description. The SWM dataset was collected by crawling *all* the software product (app) reviews under the entertainment category from an anonymous online app store database. These products consist of a diverse set of sub-categories (e.g. games, movies, news, sports). The complete collection includes $1,132,373$ reviews from $966,842$ unique users for $15,094$ apps (as of June 2012). As part of a review, a user rates a product from 1 (worst) to 5 (best).

Data analysis. Figure 3 shows the star rating distribution for reviews in the SWM dataset, with 1 being the worst and 5 being the best. As expected, the distribution has a characteristic 'J' shape, which reflects user apathy when deciding whether to review mediocre products. In this dataset, the reviews are skewed towards positive ratings.

Figure 4 shows that there are many product nodes with high degree, i.e. with high number of reviews, and much greater than the user degree. This indicates that a large portion of the reviews come from users who individually have very few reviews.

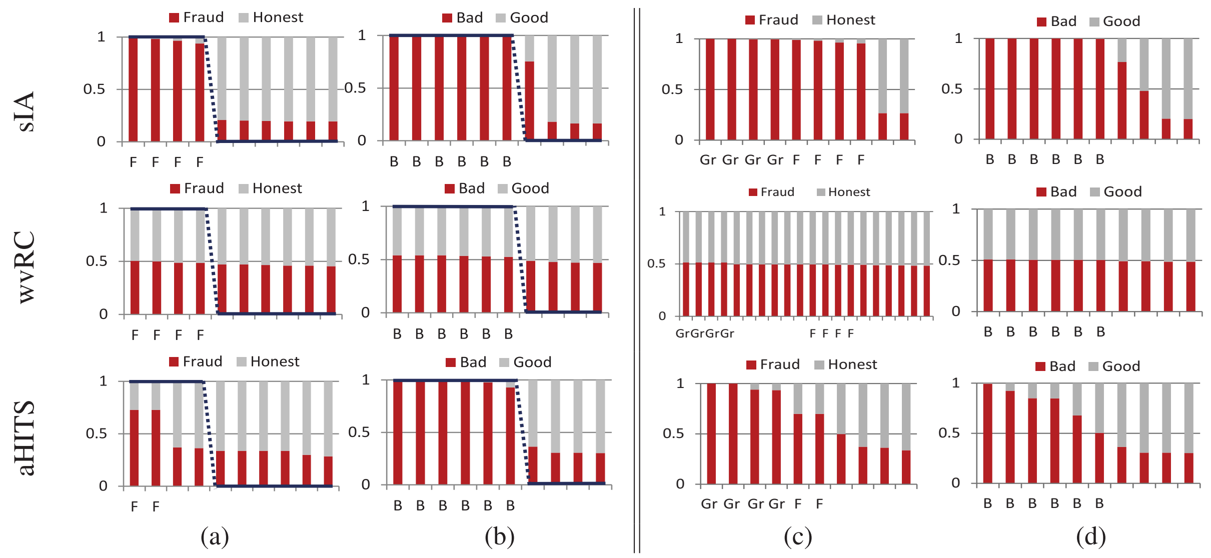


Figure 5: Proposed sIA performs the best; high scores for labeled objects (F: fraud, B: bad). (left) Class-memberships of top-ranked (a) users and (b) products by (top) sIA, (middle) wv-RC, and (bottom) adapted HITS on the synthetic review graph. sIA successfully ranks 4 fraudsters and 6 bad products on top, with very high scores. Solid lines depict ideal scores. (right) Class-memberships of top-ranked (c) users and (d) products when 4 "grumpy" (Gr) users are assigned in the graph. sIA's performance remains consistent.

Sanity Check on Synthetic Dataset

Dataset generation. In order to validate the effectiveness of our algorithm, we first tested it on a syntheticly generated review dataset, and compared it to two competitors we described before. To generate a realistic bipartite graph (with skewed degree distribution, etc.) we used the Random Typing Graph generator (Akoglu and Faloutsos 2009), with parameters W=5K, k=5, β=0.6, and q=0.4. The resulting graph had 12 connected components; we took the largest component with 196 users, 78 products, and 558 reviews (we discarded multiple edges).

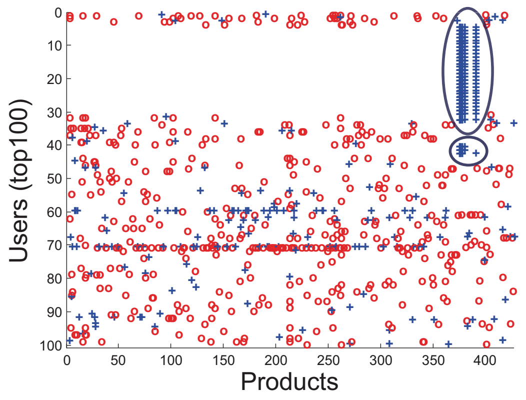
There is no known generator that mimics review networks with realistic ratings and fraudulent behavior, therefore we used the following scheme to create one: we assigned 4 users as fraudsters and the rest as honest, 7 products with the highest degree as famous good, 6 other as bad, and the rest as non-famous good. The sentiment on edges are then assigned as follows. If there is an edge in the synthetic graph, i) honest users always give '-' to bad products, ii) fraudsters always give '+' to bad products, iii) fraudsters always give '+' to the famous good products (to hide their otherwise bad activity), and iv) honest users always give '+' to good products. This way we ended up with 23 fake reviews.

Proposed algorithm result. We show the class-memberships for top-scoring (most malicious) users and products found by our sIA in top row of Figure 5. In (a), the algorithm successfully ranks all the fraudsters on top. In (b), all bad products are also ranked top with very high scores, while another product also shows up with high score (0.75) —this product has degree 1, from a fraudster with a '+' review, which increases its probability of being bad. Results are similar for fake reviews, which we omit for brevity.

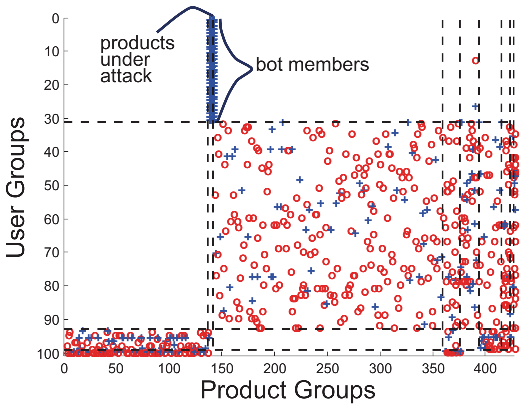
Competitors. The middle row of Figure 5 gives results of wv-RC. It successfully ranks all the fraudulent objects on top. However, the gap between the scores of malicious and benign objects is quite small, in fact 3 fraudsters have fraud scores less than 0.5 although they are ranked correctly. Finally, last row gives results of adapted HITS (recall the scores for HITS are $\in [-1, 1]$, for coherence we add 1 to the scores and divide by 2 to scale them $\in [0, 1]$). While successfully recovering bad products in (b), it misses two fraudsters in (a) (they rank 151th and 175th) —those are the ones that hide their fraudulent activity more than the others by rating more products in a honest fashion.

All-negative users. In order to make the classification task harder, we next assign 4 honest users as "grumpy" —those who always give '-' to all the products they review, regardless of being good or bad. We show the class-memberships for users in (c) and products in (d) in Figure 5 (we omit reviews for brevity). In top row, we notice that sIA ranks all the "grumpy" users on top followed by the assigned fraudsters. In middle row, wv-RC also ranks them on top while fraudsters have lower rank than before (11-14th). Similar results hold for aHITS where two fraudsters are still ranked much lower. We conclude that the algorithms do not differentiate "grumpy" users from fraudsters —they seem like fraudsters who give '-' to good products while also giving '-' to bad ones for camouflage. From the system admin's point of view both fraudsters and "grumpy" users distort the product ratings unfairly, and could both be ignored.

Sensitivity analysis. Our formulation involves choosing a small ϵ for setting compatibility potentials $\psi^s_{ij}(y_i, y_j)$. Our analysis on this and other synthetic graphs created with various parameter settings of our graph generator (an thus varying topologies) showed that a wide range of small ϵ, $\in [0.01, 0.15]$, yields desired results.



(a) induced A for top users



(b) induced A after grouping

Figure 7: (a) Adjacency matrix on top 100 users ranked by signed belief propagation algorithm and the 427 unique products these users rated. The blue '+'s and red 'o's denote the edges and indicate positive and negative reviews, respectively. (b) Adjacency matrix after rows and columns are reordered for grouping. The dashed vertical lines indicate group boundaries. Users and products resp. form 4 and 7 major groups. Notice the grouping reveals the 31-user group-attack on 5 products.

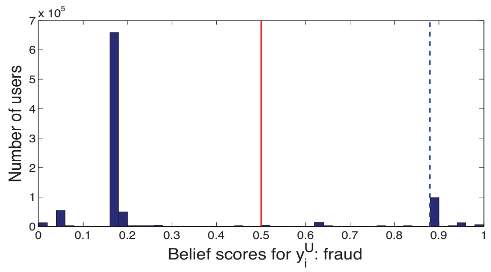


Figure 6: Distribution of users' fraud scores. The vertical red line depicts 0.5, and dashed blue line, the 90th quantile.

Results on the Real SWM Dataset

We build the customer-app network with review relations and *sign* the network edges as positive if its rating is above 3, and as negative if it is below 3.

Since we do not have ground truth label for any of the nodes, we set all prior potentials to 0.5. As we discussed before, several heuristics to obtain better priors can be derived, for example using the side information on timestamps (e.g., whether a user wrote all his/her reviews on a single day). The availability of such information, however, is dataset dependent. We employ our method in a general fashion to work completely unsupervised, while being complementary to feature-based methods in the existence of side information or any prior knowledge.

The sIA converges in 37 iterations on the SWM network. The distribution of belief scores $y_i^{\mathcal{U}}$:*fraud* of all users is shown in Figure 6. Most users have relatively small fraud scores, and are well separated from those with high scores. There are about 107K (11%) users having fraud scores greater than or equal to the 90th quantile of the distribution (0.89), and 156K users with score greater than the 0.5 threshold.

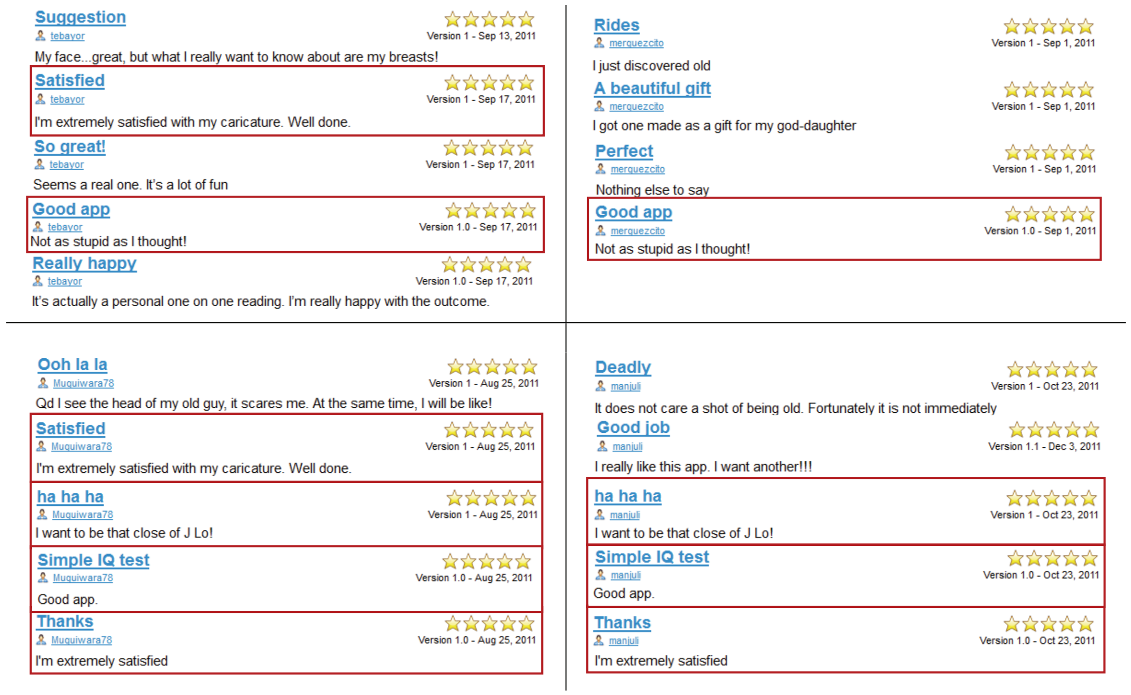
Detecting fraud-bots. Next, we obtain the top-ranked (most malicious) 100 users and extract the induced graph on these users and the (427) products they reviewed. The adjacency matrix of the induced graph is shown in Figure 7 (a). The '+'s and 'o's depict positive and negative reviews, respectively. Looking at the spy-plot, we can immediately realize a group of users who always wrote '+' reviews to a small set of products.

Cross-associations clusters the users and products, automatically revealing the set of fraudsters as shown in Figure 7 (b); 31 users all with 5-star ratings to a set of 5 products (!). Further analyses showed that all 5 products belong to the same developer, which provides further evidence of fraudulent activity. As for more evidence, we analyzed the review text of the fraudsters and found a significant number of replicated reviews among them. For example, we show the reviews of 4 example fraudsters in Figure 8. The reviews that are replicated at least once are highlighted with a (red) box. The graph of these users, where an edge exists between two users if they have at least one same review in common, is in fact connected. This observation suggests multiple user accounts created by the same fraudster.

The fraudsters in the detected bot, all with rating 5, significantly affect the average rating of the 5 products they reviewed. In Figure 9, notice that all their average ratings drop close to 1, once those fraudsters and their reviews are removed from our dataset.

Comparison to other methods. Unlike the synthetic dataset as discussed before, our SWM data does not contain ground truth labels. Therefore, obtaining precision-recall curves of the alternative methods in comparison to sIA is out of the scope of this work. We can, however, use our results in the previous section as a pseudo ground truth. That is, we treat the 31 users which form a bot as true fraudsters.

The bot-fraudsters rank within rank 5 to 43 in the sorted list of users by sIA (Fig. 7(a)). The same users lie within ranks 36 and 131 in the sorted list by adapted HITS (aHITS). Finally, wv-RC puts them between ranks 9860 and 10310 (note that the order of users could be different).


Suggestion
tebayor
Version 1 - Sep 13, 2011
My face...great, but what I really want to know about are my breasts!

Satisfied
tebayor
Version 1 - Sep 17, 2011
I'm extremely satisfied with my caricature. Well done.

So great!
tebayor
Version 1 - Sep 17, 2011
Seems a real one. It's a lot of fun

Good app
tebayor
Version 1.0 - Sep 17, 2011
Not as stupid as I thought!

Really happy
tebayor
Version 1.0 - Sep 17, 2011
It's actually a personal one on one reading. I'm really happy with the outcome.

Rides
merquezcito
Version 1 - Sep 1, 2011
I just discovered old

A beautiful gift
merquezcito
Version 1 - Sep 1, 2011
I got one made as a gift for my god-daughter

Perfect
merquezcito
Version 1 - Sep 1, 2011
Nothing else to say

Good app
merquezcito
Version 1.0 - Sep 1, 2011
Not as stupid as I thought!

Ooh la la
Muguiwara78
Version 1 - Aug 25, 2011
Qd I see the head of my old guy, it scares me. At the same time, I will be like!

Satisfied
Muguiwara78
Version 1 - Aug 25, 2011
I'm extremely satisfied with my caricature. Well done.

ha ha ha
Muguiwara78
Version 1 - Aug 25, 2011
I want to be that close of J Lo!

Simple IQ test
Muguiwara78
Version 1.0 - Aug 25, 2011
Good app.

Thanks
Muguiwara78
Version 1.0 - Aug 25, 2011
I'm extremely satisfied

Deadly
manjuli
Version 1 - Oct 23, 2011
It does not care a shot of being old. Fortunately it is not immediately

Good job
manjuli
Version 1.1 - Dec 3, 2011
I really like this app. I want another!!!

ha ha ha
manjuli
Version 1 - Oct 23, 2011
I want to be that close of J Lo!

Simple IQ test
manjuli
Version 1.0 - Oct 23, 2011
Good app.

Thanks
manjuli
Version 1.0 - Oct 23, 2011
I'm extremely satisfied

Figure 8: Reviews of 4 example bot members detected in SWM data for (from top to bottom) the same 5 products (all from the same developer) (see §). Reviews shared among these users are highlighted with a (red) box. Replication of all 5-star reviews provides evidence for fraudulent activity. Also notice the reviews of each fraudster are written mostly on a single day.

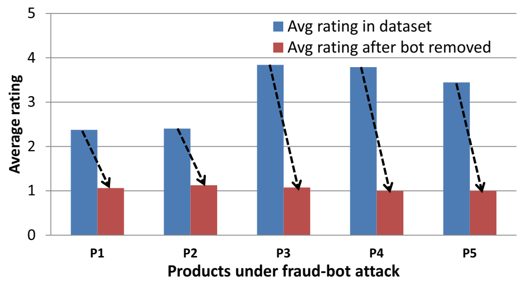


Figure 9: Average rating of 5 products under attack by a bot of fraudsters (see §, Fig. 7) drops significantly to ≈1 (lowest possible rating) after removing their fake reviews.

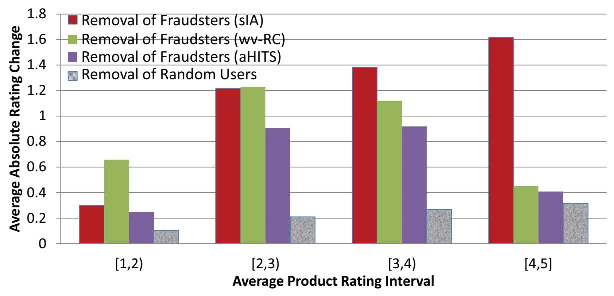


Figure 10: FRAUDEAGLE top-scorers matter. Absolute change in average ratings of all products, after removing the reviews of top fraudsters detected by each method, along with change when as many random users are removed (averaged over 10 runs).

Another way to compare the methods is to study the impact of the fraudsters they detect on the average ratings of products. In particular, we study the average rating of a product when fraudsters are included versus when they are excluded. Figure 10 gives the mean absolute change in average ratings of products after top users are removed by each method —these are the users with fraud scores greater than 0.5 for sIA and wv-RC, and with honesty scores less than 0 for aHITS. The figure shows that the rating changes are more significant for the high-rating products ([3-5]) for the removed fraudsters by sIA, while changes are more significant for low-rating ([1-2]) products for fraudsters removed by wv-RC. In fact when top fraudsters by sIA are removed, the average rating of high-rated products drop by more than 1 point on average. The removed random users, on the other hand, do not have as significant effects on the ratings.

Computational complexity

Lemma 1 *Proposed* FRAUDEAGLE *is scalable to large data, with computational complexity linear in network size.*

Proof 1 In step 1 of FRAUDEAGLE, sIA performs message passing over the edges in a repeated fashion (see Outline 1 Line 12-21), with time complexity $O(|\mathcal{E}|d^2t)$, where $|\mathcal{E}|$ is the number of edges in the network, d is the maximum domain size of a variable (i.e. number of classes, which is often small), and t is the number of iterations until convergence. In our setting, domain sizes of both users and products is $d = |\mathcal{L}^{\mathcal{U}}| = |\mathcal{L}^{\mathcal{P}}| = 2$, and $t \ll |\mathcal{E}|$ is often small ($t = 37$ on SWM data). Therefore, the time complexity is linear in the number of edges, i.e. network size.

In step 2, we cluster the induced graph on top users and their products using the cross-associations algorithm, which is also linear in number of edges (Chakrabarti et al. 2004). The induced graph is also much smaller than the input graph.

Finally, we also empirically show in Figure 11, that the running time grows linearly with increasing size.

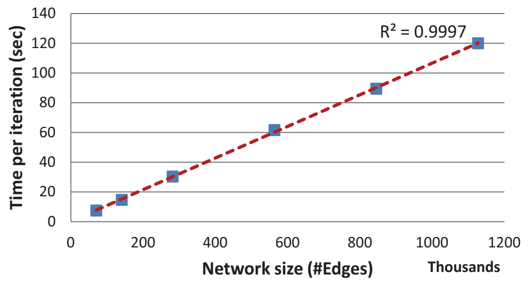


Figure 11: Run time for sIA vs network size. Notice the linear scalability with a good fit.

Conclusions

We propose a novel framework called FRAUDEAGLE that exploits the network effects to automatically detect fraudulent users and fake reviews in online review networks. Our main contributions are:

- *Problem formulation*: We formally define the opinion fraud detection problem as a classification task on *signed* bipartite review networks, and thus we capture the *network effects* for improved classification.
- *Scoring algorithm*: We show how to efficiently solve the inference problem, on *signed* bipartite networks. Our approach uses several compatibility matrices and computes scores for all 3 types of objects: reviews (fake/truthful), users (honest/fraud), and products (good/bad quality).
- *Desirable properties*: FRAUDEAGLE is (a) *general* and applicable to several types of review networks; (b) *unsupervised*, requiring no prior knowledge or side information (although it can use it, if it exists); and (c) *scalable*, with linear run-time complexity.
- *Evaluation*: We compared FRAUDEAGLE against alternatives, on synthetic as well as real networks. FRAUDEAGLE successfully detects fraudulent attack groups, and the users that significantly distort product ratings.

Our method is complementary to previous works that use text and behavioral clues; future work will seed our algorithm with priors inferred from those clues, and study the effects of more informed priors on performance.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback. Research was sponsored by NSF under Grant No. IIS1017415, ARL under Cooperative Agreement No. W911NF-09-2-0053, and ADAMS sponsored by DARPA under Agreements No.s W911NF-12-C-0028, W911NF-11-C-0200, and W911NF-11-C-0088.

References

Akoglu, L., and Faloutsos, C. 2009. RTG: A recursive realistic graph generator using random typing. *DAMI* 19(2):194–209.

Chakrabarti, D.; Papadimitriou, S.; Modha, D. S.; and Faloutsos, C. 2004. Fully automatic cross-associations. In *KDD*, 79–88.

Feng, S.; Banerjee, R.; and Choi, Y. 2012. Syntactic stylometry for deception detection. In *ACL*, 171–175.

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

Hill, S.; Provost, F.; and Volinsky, C. 2007. Learning and inference in massive social networks. In *MLG*, 47–50.

Hitlin, P. 2003. False reporting on the internet and the spread of rumors: Three case studies. *Gnovis J.*

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

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

Kindermann, R., and Snell, J. L. 1980. *Markov Random Fields and Their Applications*.

Kleinberg, J. M. 1998. Authoritative sources in a hyperlinked environment. In *SODA*, 668–677.

Koren, Y. 2009. Collaborative filtering with temporal dynamics. In *KDD*, 447–456.

Kost, A. May 2012. *Woman Paid To Post Five-Star Google Feedback*. http://bit.ly/SnOzRi.

Li, F.; Huang, M.; Yang, Y.; and Zhu, X. 2011. Learning to Identify Review Spam. In *IJCAI*.

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

Macskassy, S., and Provost, F. 2003. A simple relational classifier. In *2nd Workshop on Multi-Relational Data Mining, KDD*, 64–76.

McGlohon, M.; Bay, S.; Anderle, M. G.; Steier, D. M.; and Faloutsos, C. 2009. Snare: a link analytic system for graph labeling and risk detection. In *KDD*.

Mendoza, M.; Poblete, B.; and Castillo, C. 2010. Twitter under crisis: can we trust what we rt? In *SOMA*, 71–79.

Menon, A. K., and Elkan, C. 2011. Link prediction via matrix factorization. In *ECML/PKDD*.

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

Neville, J.; Simsek, O.; Jensen, D.; Komoroske, J.; Palmer, K.; and Goldberg, H. G. 2005. Using relational knowledge discovery to prevent securities fraud. In *KDD*, 449–458.

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

Pandit, S.; Chau, D. H.; Wang, S.; and Faloutsos, C. 2007. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW*.

Pang, B.; Lee, L.; and Vaithyanathan, S. 2002. Thumbs up? sentiment classification using machine learning techniques. *CoRR* cs.CL/0205070.

Streitfeld, D. August 2011. *In a Race to Out-Rave, 5-Star Web Reviews Go for 5 Dollars*. http://nyti.ms/nqiYyX.

Wang, G.; Xie, S.; Liu, B.; and Yu, P. S. 2011. Review graph based online store review spammer detection. In *ICDM*, 1242–1247.

Xie, S.; Wang, G.; Lin, S.; and Yu, P. S. 2012. Review spam detection via temporal pattern discovery. In *KDD*, 823–831.

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