DeepTrust: An Automatic Framework to Detect Trustworthy Users in Opinion-based Systems

Edoardo Serra
edoardoserra@boisestate.edu
Computer Science Department
Boise State University
Boise, Idaho

Anu Shrestha
anushrestha@u.boisestate.edu
Computer Science Department
Boise State University
Boise, Idaho

Francesca Spezzano
francescaspezzano@boisestate.edu
Computer Science Department
Boise State University
Boise, Idaho

Anna Squicciarini
acs20@psu.edu
Information Sciences and Technology
Pennsylvania State University
University Park, PA

ABSTRACT

Opinion spamming has recently gained attention as more and more online platforms rely on users' opinions to help potential customers make informed decisions on products and services. Yet, while work on opinion spamming abounds, most efforts have focused on detecting an individual reviewer as spammer or fraudulent. We argue that this is no longer sufficient, as reviewers may contribute to an opinion-based system in various ways, and their input could range from highly informative to noisy or even malicious.

In an effort to improve the detection of trustworthy individuals within opinion-based systems, in this paper, we develop a supervised approach to differentiate among different types of reviewers. Particularly, we model the problem of detecting trustworthy reviewers as a multi-class classification problem, wherein users may be fraudulent, unreliable or uninformative, or trustworthy. We note that expanding from the classic binary classification of trustworthy/untrustworthy (or malicious) reviewers is an interesting and challenging problem. Some untrustworthy reviewers may behave similarly to reliable reviewers, and yet be rooted by dark motives. On the contrary, other untrustworthy reviewers may not be malicious but rather lazy or unable to contribute to the common knowledge of the reviewed item.

Our proposed method, DeepTrust, relies on a deep recurrent neural network that provides embeddings aggregating temporal information: we consider users' behavior over time, as they review multiple products. We model the interactions of reviewers and the products they review using a temporal bipartite graph and consider the context of each rating by including other reviewers' ratings of the same items. We carry out extensive experiments on a real-world dataset of Amazon reviewers, with known ground truth about spammers and fraudulent reviews. Our results show that DeepTrust can detect trustworthy, uninformative, and fraudulent users with an F1-measure of 0.93. Also, we drastically improve on detecting fraudulent reviewers (AUROC of 0.97 and average precision of 0.99 when combining DeepTrust with the F&G algorithm) as compared to REV2 state-of-the-art methods (AUROC of 0.79 and average precision of 0.48). Further, DeepTrust is robust to cold start users and overperforms all existing baselines.

CCS CONCEPTS

• **Information systems** → **Reputation systems**; • **Security and privacy** → **Trust frameworks**.

KEYWORDS

Trustworthy user detection, Deep learning, Temporal embedding, Fraudulent users, Opinion-spammer detection

ACM Reference Format:
Edoardo Serra, Anu Shrestha, Francesca Spezzano, and Anna Squicciarini. 2020. DeepTrust: An Automatic Framework to Detect Trustworthy Users in Opinion-based Systems. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy (CODASPY '20), March 16–18, 2020, New Orleans, LA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3374664.3375744

1 INTRODUCTION

Opinion-based systems rely on users' collective opinion to rank or rate products, services, or even other users' qualifications or qualities (e.g., editors, programmers, micro-task workers). Such a crowdsourced approach allows for transparency and enables informed choices for other users interested in learning about certain reviewed items (or services). Underpinning such a system is an inherent expectation of trust on the participants' willingness and commitment to compile reliable and unbiased reviews accounting for their own experience with the item being reviewed. Although this expectation is generally met, research has consistently shown how these platforms are polluted by unreliable reviews that are either fraudulent, uninformative or inaccurate [4, 6, 16].

The authors are listed in alphabetic order.

CODASPY '20, March 16–18, 2020, New Orleans, LA, USA

ACM ISBN 978-1-4503-7107-0/20/03…$15.00
https://doi.org/10.1145/3374664.3375744

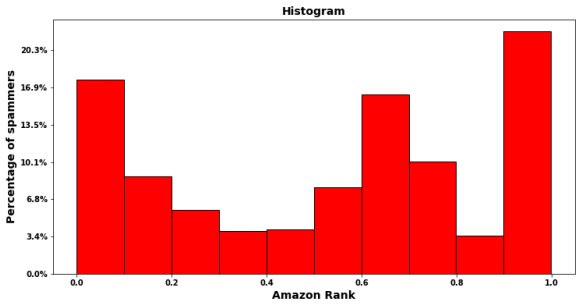

Figure 1: Distribution of the Amazon rank among fraudulent reviewers in the dataset from Section 4.

Malicious actors use underground Internet sites to recruit fake reviewers, who are given strict guidelines on the type of review to write, so as to generate relatively high-quality reviews - that are difficult to ascertain from authentic reviews [16]. This makes the detection of rogue reviews a non-trivial task. Existing user-led endorsement features offered by these platforms (e.g., "Is this review helpful?" in Amazon) are often unable to single out bogus reviews. Figure 1 shows the distribution of the Amazon rank for fraudulent (or opinion-spam) users on a selected dataset of fraudulent reviewers (see Section 4). Amazon ranks each reviewer according to peer-rated helpfulness. The lower the rank, the better the reviewer is. A reviewer's rank is determined by the overall helpfulness of all their reviews, factoring in the number of reviews they have written, and, more recently, a review is written, the greater its impact on rank [1]. As Figure 1 shows, while the majority of the fraudulent users have a high rank, some of them are sophisticated enough to fuel the ranking system and camouflage as top-reviewers.

Opinion-based platforms have strong interests in identifying the best (and worst) contributors of their sites, to block or filter fraudsters, and to provide incentives to reviewers who contribute with honest and accurate information. In an effort to improve the detection of trustworthy individuals within opinion-based systems, in this paper, we focus on classifying reviewers based on their behavioral patterns and feedback they received from other peer reviewers. Particularly, we model the problem of detecting *trustworthy* reviewers as a multi-class classification problem, wherein the possible classes of users include fraudulent, unreliable (or uninformative), and trustworthy. Here, by trustworthy, we refer to individuals who positively contribute to the opinion-based system by means of productive content and, therefore, whose reviews can be trusted as informative and useful. Fraudulent reviewers are instead malicious in nature, in that truly uploaded to affect the ranking of a product or a seller's reputation. For instance, let us consider user u_2 in Figure 2. This user is fraudulent as she/he is trying to demote p_2, which is a generally liked product and promote p_3, which other users consider a bad product. Finally, unreliable reviewers are users whose reviews are "noisy" in that they are not informative or of generally poor quality (e.g., inaccurate or generic). Further, as some reviewers may not have sufficient historical records to ascertain

[1] https://www.amazon.com/gp/customer-reviews/guidelines/top-reviewers.html

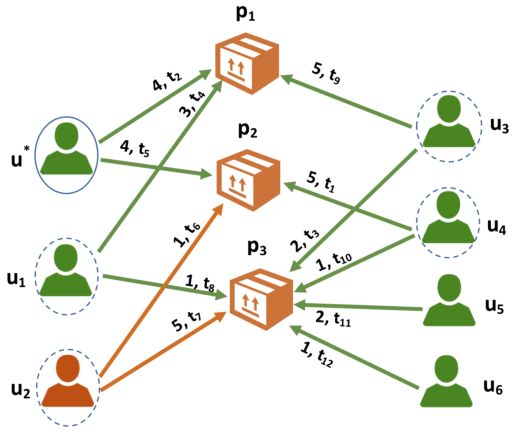

Figure 2: Sample user-item bipartite rating network ($t_i \leq t_j$ if $i \leq j$).

their nature reliably, we also consider a class of "unknown" users, whose true behavioral patterns is not well-supported by data. We classify such users based on their limited history or information.

We note that expanding from the classic binary classification of trustworthy/untrustworthy (or malicious) reviewers to a multi-class setting gives rise to an interesting and challenging problem. Untrustworthy reviewers may be rooted by a variety of motives, and be either perceived as uninformative or unreliable, or be actually malicious. Hence, natural language processing may not be sufficient nor accurate.

Our proposed approach accounts for users' behavior over time, as they review multiple products by means of temporal embeddings. We model the interactions of reviewers and the products they review using a temporal review sequence and consider the context of each interaction by including other reviewers' interactions with the same items.

Precisely, we propose DeepTrust, an unsupervised temporal user embedding model (i.e., it does not require any label about the category of the user to be learned) that is able to extract latent features for each user automatically. Given a certain user u, these features take into account the entire temporal evolution of *all* the posted reviews from all users who reviewed the same products of the user u. In summary, the main strengths of this approach include:

- The entire historical sequence of the reviews can be reconstructed given the embedding features. Thus, we do not suffer from information loss as other existing methods that rely on aggregation of user information;
- Since for each user we also consider the reviews of their peers, the obtained sequences of reviews are usually large enough to allow the neural network embedding to be trained and produce significant embedding features. This allows us to classify users who reviewed a few products and whose history is, therefore, hard to leverage.

We report the results of our approach on a large dataset of Amazon reviews with fraudulent reviewer ground truth [4]. Our results

show a drastic improvement in the detection of fraudulent reviewers as compared to related approaches. In addition, DeepTrust can detect trustworthy, uninformative, and fraudulent users with an F1-measure of 0.93. Also, we drastically improve on detecting fraudulent reviewers (AUROC of 0.97 and average precision of 0.99 when combining DeepTrust with the F&G algorithm) as compared to REV2 state-of-the-art methods (AUROC of 0.79 and average precision of 0.48). Moreover, we show that DeepTrust performances do not decrease in the case of cold start users, and DeepTrust overperforms all the baselines approaches.

2 RELATED WORK

Knowing the trustworthiness or reputation of a node u in opinion-based systems allows other peers to assign the right value to u's judgments. Typically, the trustworthiness of a node is computed as a global trust value taking into account the interactions of a node with other nodes or items in the system [6, 10]. Specifically to opinion-based systems where users provide their opinions of items or products and not of other users as, for instance, in Amazon, several works have been proposed that have the common denominator of computing a trustworthiness score for each user and a goodness score for each item reflecting the rating the item actually deserves. These algorithms assume to work with a bipartite user-item rating network (cf. Figure 2 for an example).

Mishra and Bhattacharya [14] proposed the Bias and Deserve (BAD) algorithm for computing the trustworthiness of a node as a *bias* quantifying the tendency of the node in overestimating or underestimating the rating an item deserves (the higher the bias, the less the trustworthiness of the node). The algorithm also computes a *deserve* score for each item that takes into account the bias of the users that are ranking that item. The bias and deserve scores are computed by a pair of mutually recursive equations.

Similarly, Kumar et al. [12] defined the Fairness and Goodness (F&G) algorithm, which computes a fairness score for each user and a goodness score for each item. Specifically, the *fairness* of a user is a measure of how fair or trustworthy the user is in rating items. Intuitively, a 'fair' or 'trustworthy' rater should give an item the rating that it deserves, while an 'unfair' one would deviate from that value. The latter could be the case of a fraudulent user who is trying to promote (resp. demote) a bad (resp. good) item or a user that is in good faith but unreliable or uninformative. The *goodness* of an item specifies how much users in the system like the item and what its true quality is. Fairness and goodness are mutually recursively computed. Specifically, the goodness of an item is given by the average of its rating where each rating is weighted by the fairness of the rater, while the fairness of a user considers how much the ratings a user gives are far from the goodness of the items. The higher the fairness, the more trustworthy the user is.

Recently, Kumar et al. [11] proposed the REV2 algorithm, which is an extension of the F&G algorithm where they compute a fairness score for each user, a goodness score for each item and a reliability score for each rating as they argue that fraudulent users can also give reliable rating to increase their reputation and fair users can sometimes give unreliable rating, as in case of the class of unreliable or uninformative users we want to detect. Again, fairness is a measure of how trustworthy a user is, and a fair user is one that assigns reliable ratings that are close to the goodness of the items. REV2 computes fairness, goodness, and reliability by using a set of mutually recursive equations where they also combine user behavioral properties computed via the BirdNest algorithm [8].

Trustiness [20] is another algorithm similar to the above-mentioned ones that computes a trustworthiness score for each user, an honesty score for each item, and a reliability score for each rating.

Regarding the detection of fraudulent users (or opinion spammers) in opinion-based systems specifically, existing work can be categorized into network-based methods, behavioral-based methods, and hybrid methods combining both network and behavioral properties. BAD, F&G, and Trustiness can be used to detect fraudulent users as well, and they can be categorized as network-based algorithms. FraudEagle [2] is another network-based algorithm that models the user-item bipartite rating network as a Markov Random Field and computes an anomaly score for each user that is used to identify the opinion spammers. They assume that honest (resp. fraudulent) reviewers are more likely to give positive ratings to good (resp. bad) products and honest (resp. fraudulent) reviewers are more likely to give negative ratings to bad (resp. good) products.

Behavioral-based methods leverage the fact that fraudulent reviewers write many, shorter, positive (4 or 5 stars) and self-similar reviews in short bursts of time [5, 9, 15, 16]. SpamBehavior [13] proposes ranking and supervised methods exploiting the fact that opinion spammers target a specific set of products and their ratings deviate from the ones of benign users. BirdNest [8] detects opinion spammers according to the fact that (i) fake reviews occur in short bursts of time and (ii) fraudulent user accounts have skewed rating distributions.

Among hybrid methods, SpEagle [17] extends FraudEagle by combining both the user-review-product graph and metadata such as text, timestamps, and ratings to detect fraudulent users, reviews, and targeted products. REV2 combines both network and behavioral properties (by incorporating the user BirdNest anomalous score) and is the state-of-the-art algorithm in detecting fraudulent users in opinion-based systems. *In this paper, we extensively compare with REV2 (and other algorithms presented in this section) and show that our proposed DeepTrust user embedding technique significantly outperforms REV2 and other algorithms under different settings. Also, DeepTrust can identify uninformative reviewers, which are not considered in prior work, to avoid they are mistakenly classified as fraudulent users, and addresses the cold start user problem.*

3 DEEPTRUST USER EMBEDDING

In this section, we describe DeepTrust, a deep-learning-based approach to extract user features from their temporal review sequence in an unsupervised way. An "embedding" is a technique to transform an input sequence into a k-dimensional vector. Once the embedding is obtained for each user, its vectorial representation can be used as features in input to machine learning algorithms. In this paper, we use the computed user embedding to determine if a user belongs to one of these categories: *trustworthy*, *unreliable or uninformative*, or *fraudulent*.

Let $U = \{u_1, \ldots, u_m\}$ be the set of users active in the opinion-based system, and $P = \{p_1, \ldots, p_l\}$ be the set of products being reviewed by users in U. We denote by R the set of all reviews

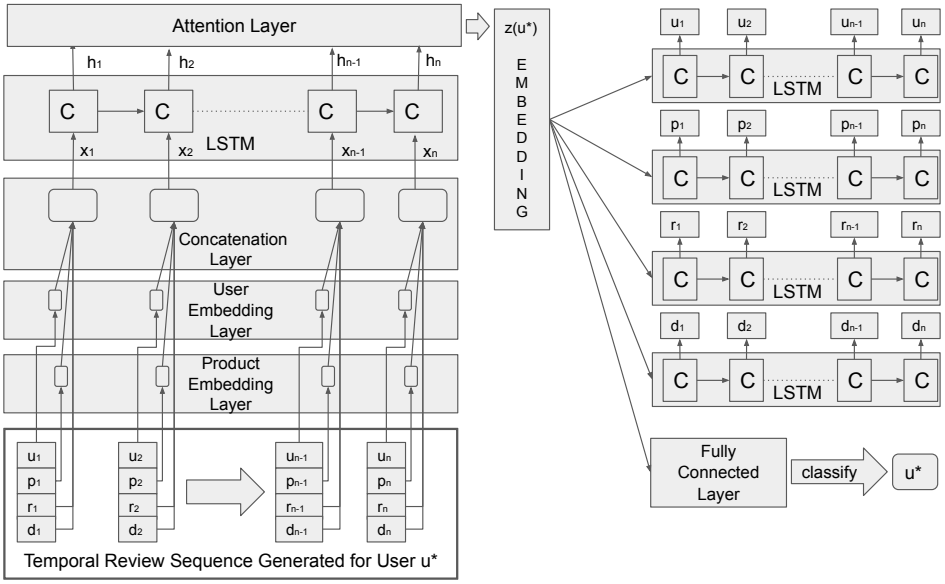

Figure 3: DeepTrust architecture.

generated by users in U for products in P. Each review $re \in R$ is represented by a 4-tuple $re = (u, p, r, t)$ where $u \in U$ is the reviewer, $p \in P$ is the product being reviewed, $r \in \{1, 2, 3, 4, 5\}$ is the five-star scale rating assigned by u to p, and t is the review timestamp. Given a user $u \in U$, we define the set of products reviewed by u as $pr(u) = \{p|(u, p, \_, \_) \in R\}$. Given a product $p \in P$, the set of users who reviewed p is defined as $us(p) = \{u|(u, p, \_, \_) \in R\}$.

In order to define an embedding describing the behavior of a user $u^* \in U$ in the opinion-based system, we consider, in addition to their reviews, also the reviews from other users in the system that can have potentially shaped u^*'s opinions or that can help in identifying an anomalous or fraudulent behavior. We call these reviews the *context* of the user u^*. For instance, users' opinion on a given product may change as they read other users' reviews, or their ratings may change slightly based on other reviewers' ratings of the same or similar products. Also, some reviews written by u^* can affect the opinion of other users. Moreover, we also consider reviews made by other users *after* u^*'s reviews as they can help uncover fraudulent behavior. For instance, opinion spammers may be engaged to promote a new (but not good) product p, which initially is described only by spam reviews with star ratings 4 or 5. Later, benign users start reviewing the product p and giving low ratings. These future reviews are helpful to recognize the opinion spammers.

Accordingly, our embedding will consider in input a sequence of temporally ordered reviews for each user. Given a user u^* the sequence consists of the set of all the reviews given by u^* plus the reviews given by other users to the set of products reviewed by u^* (as they can be potentially related to the behavior of u^*). Then, we define the concept of a temporal review sequence as follows.

Definition 3.1 (Temporal Review Sequence). Given a user $u^* \in U$, let $re(u^*) = \{(u, p, r, t) \mid (u, p, r, t) \in R,\ p \in pr(u^*)\}$ be the subset of reviews that describe the rating behavior of u^* in comparison to the ratings that other users give to the products rated by u^*. [2] The temporal review sequence

$$tres(u^*) = \langle (u_0, p_0, r_0, t_0, d_0), (u_1, p_1, r_1, t_1, d_1), \ldots, (u_n, p_n, r_n, t_n, d_n) \rangle$$

for the user u^* is the set of reviews in $re(u^*)$ ordered by the timestamp such that

- $(u_i, p_i, r_i, t_i) \in re(u^*)$ for each $i \in \{1, \ldots, n\}$
- $t_{i-1} \leq t_i$ for each $i \in \{1, \ldots, n\}$
- $d_i = \begin{cases} 0, & i = 0 \\ t_i - t_{i-1}, & i > 0 \end{cases}$ □

It is worth noting that two reviews $(u_{i-1}, p_{i-1}, r_{i-1}, t_{i-1})$ and (u_i, p_i, r_i, t_i) that occur at the same time $(t_{i-1} = t_i)$ will be recognized by $d_i = 0$. Moreover, we define the *context* of the user u^* as the subsequence of reviews in $re(u^*)$ not written by u^*, i.e., $ctx(u^*) = \{(u, p, r, t) \mid (u, p, r, t) \in R,\ p \in pr(u^*), u \neq u^*\}$.

Example 3.2. Let us consider the sample user-item bipartite rating network shown in Figure 2. We have three products $P = \{p_1, p_2, p_3\}$ reviewed by seven users $U = \{u^*, u_1, u_2, u_3, u_4, u_5, u_6\}$. Each edge is labeled with rating and review timestamp. For user u^*, the temporal review sequence that describes their behavior is

$$tres(u^*) = \{(u_4, p_2, 5, t_1), (u^*, p_1, 4, t_2), (u_1, p_1, 3, t_4), (u^*, p_2, 4, t_5), (u_2, p_2, 1, t_6), (u_3, p_1, 5, t_9)\}$$

since p_1 and p_2 are the products reviewed by user u^* and u_1, u_2, u_3, u_4 are other users reviewing same products. The context for user u^* is given by

$$ctx(u^*) = \{(u_4, p_2, 5, t_1), (u_1, p_1, 3, t_4), (u_2, p_2, 1, t_6), (u_3, p_1, 5, t_9)\}$$

□

[2] Because of data limitation, as discussed in Section 4, behavioral analysis is limited to users' rating activities. However, our proposed technique can be easily extended in case other behavioral data is available.

3.1 DeepTrust Architecture

We compute a set of latent features describing the user behavior that can be used as input to machine learning algorithms to classify users as trustworthy, unreliable, or fraudulent. We learn the features from the input user temporal review sequence in an unsupervised way as we aim to be generic and not constrained on the particular task we are going to use the features for.

Figure 3 illustrates DeepTrust, our proposed deep-learning architecture to compute the user embeddings. Given as input a temporal review sequence of variable length $tres(u^*)$, DeepTrust maps the sequence into a fixed-size vectorial representation $z(u^*) \in \mathbb{R}^k$. For each element $(u_i, p_i, r_i, t_i, d_i)$ in the sequence $tres(u^*)$, DeepTrust associates the IDs of the user u_i and product p_i with their latent representations $e(u_i)$ and $e(p_i)$ (embeddings). These operations are performed by the "Product Embedding Layer" and the "User Embedding Layer" of the neural network.

Note that the embeddings associated with each product and user will be determined during the training phase, i.e., they will be trained with the entire network. Once the IDs are converted into embeddings, the "Concatenation Layer" will concatenate, for each element $(u_i, p_i, r_i, t_i, d_i)$ in the sequence, the embedding of the user $e(u_i)$, the embedding of the product $e(p_i)$, the rating r_i, and the time elapsed since the previous review (delta) d_i. We denote by x_i the above concatenation. Moreover, to improve the training convergence time of our network, we scaled with a standard scaler all the rating r_i and all the deltas d_i. Once the concatenated representation x_i is obtained for each element at time i in the temporal review sequence, the sequence $\langle x_1, \ldots, x_n \rangle$ is passed through a Long Short Term Memory (LSTM) recurrent neural network [7]. Figure 4, adapted from [1], describes the architecture of a single LSTM cell that outputs the next state h_t by taking in input the previous state h_{t-1} and the next symbol x_t. The operations done by the single LSTM cell C are described by the following equations:

$$
\begin{aligned}
a_t &= \rho(W_a \cdot [h_{t-1}, x_t]) \\
b_t &= \rho(W_b \cdot [h_{t-1}, x_t]) \\
y_t &= \tanh(W_y \cdot [h_{t-1}, x_t]) \\
g_t &= \rho(W_g \cdot [h_{t-1}, x_t]) \\
c_t &= c_{t-1} \cdot a_t + b_t \cdot y_t \\
h_t &= \tanh(c_t) \cdot g_t
\end{aligned}
$$

where the W_a, W_b, W_y and W_g are the weights representing the LSTM cell C and the entire LSTM neural network.

The LSTM outputs, for each element of the sequence, a vectorial representation h_i representing the sub-sequence till the element i. We then merge the vectorial representation sequence $\langle h_1, \ldots, h_n \rangle$ via a soft attention layer that produces a unique fixed-size vectorial representation for the temporal review sequence. The attention [3] is a mechanism to discover parts of the sequence that are more relevant for describing user behavior and weight them more when computing the user embedding. The attention layer takes as input the vectorial representation sequence $\langle h_1, \ldots, h_n \rangle$ from the LSTM and returns the output $z(u^*) = tanh(W_c[co; h_n])$ where

- $W_c \in \mathcal{R}^{2|h_n| \times |h_n|}$ is a set of weights to learn, and
- co is the attention context vector obtained by the weighed mean of all h_i vectors, i.e., $co = \sum_{i=1}^{n} c_i^s \cdot h_i$.

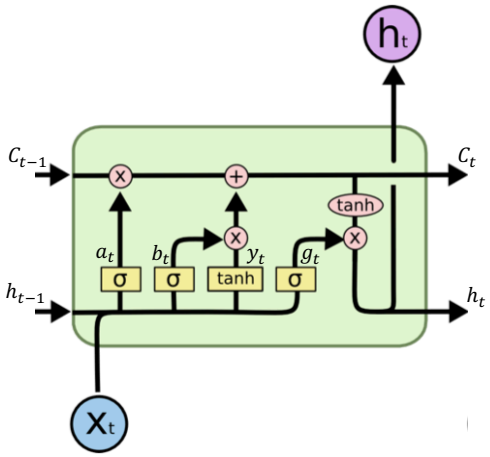

Figure 4: Description of an LSTM cell C. Figure adapted from [1].

The weights $\{c_1^s, \ldots, c_n^s\}$ to compute the attention context vector are obtained by using (1) a unique fully connected layer that, applied singularly to each h_i, produces a single value q_i, and (2) a softmax layer taking in input all $\{q_1, \ldots, q_n\}$ and finally outputting the $\{c_1^s, \ldots, c_n^s\}$.

The output $z(u^*)$ represents the embedding of the user u^* that summarizes their entire temporal review sequence.

Given the embedding $z(u^*)$, the next part of our neural network works on the reconstruction of the input temporal review sequence $tres(u^*)$. The embedding $z(u^*)$ is passed through four different LSTMs that reconstruct the original sequences of all the sub-part of each review: the sequence of user IDs, the sequence of product IDs, the sequence of product ratings, and the sequence of deltas.

We use a softmax layer to reconstruct product and user sequences, and add to the global loss function the cross-entropy loss for each element of the sequence. The reconstruction for the rating and the deltas is done by adding to global loss function the mean square loss for each element of the sequence. This ensures that the embedding $z(u^*)$ stores all the information contained in $tres(u^*)$. In addition, we pass $z(u^*)$ through a fully connected layer that identifies the specific user u^* who is generating the temporal review sequence. By training this entire neural network (i.e., minimizing the global loss function), the output is an embedding $z(u)$ for each user $u \in U$.

4 DATASET

We carry out our experimental evaluation on an Amazon dataset from Fayazi et al. [4]. The dataset includes a large candidate set of potential deceptive reviews, reviewers, and targeted products. Deceptive reviews are retrieved by identifying products that were targeted for manipulation in underground crowdsourcing platforms (e.g., RapidWorkers.com, ShortTask.com, and Microworkers.com). These platforms pay workers to post a review on a target site (e.g., Amazon, Yelp). The dataset also includes samples of reviews of other products performed by suspected deceptive reviewers, along with their profile information (i.e., reviewers whose reviews appear

Table 1: Number of users for each class in the Amazon dataset.

Class	Num. of users
Fraudulent	846
Trustworthy	5,322
Unreliable	91
Unknown	7,872

in the targeted items). A review is labeled as deceptive if (i) the review was associated with a product which was targeted by a crowdsourced malicious task; and if (ii) the review had a high rating and was posted within a few days after the task was posted. Otherwise, it is labeled as a legitimate review.

For each reviewer, attributes such as helpfulness ratio (or reviewer helpfulness), number of helpful/unhelpful votes, and Amazon rank are included. Amazon users can provide feedback on other users' reviews by voting if a given review is helpful or not. Thus, given a reviewer u, their *helpful ratio* is given by the number of helpful votes divided by the total number of helpful/unhelpful votes given to all the reviews of this user. The Amazon rank, as explained earlier in the Introduction, is assigned by taking into account the helpfulness of the reviewer and the recency of the votes. Thus, when reviewers receive enough helpful/unhelpful votes (we consider a threshold of 20 votes in this paper), their helpfulness score can be seen as a collective measure of user trustworthiness (when it is high) or user unreliability (when it is low).

We filtered out reviewers who did only one review and products that have only one review. The resulting dataset includes 94.8K reviews, 14.1K reviewers, and 22K products. We further split reviewers into four classes:

- **Fraudulent users**: users who are marked as opinion spammers (fraudulent) in the dataset.
- **Trustworthy users**: users who are not fraudulent and who have at least 20 helpful votes and a helpfulness ratio ≥ 0.75.
- **Unreliable or uninformative users**: users who are not fraudulent and who have at least 20 helpful votes and a helpful ratio ≤ 0.25.
- **Unknown users**: users who are not fraudulent and have less than 20 helpful votes or the helpfulness ratio is between 0.25 and 0.75. We classify them as unknown as there is not enough evidence in the helpfulness data to reliably assign them a classification label.

Table 1 shows the breakdown of labels in each of the above classes. As we discuss in the next section, the obvious class imbalance shown here is taken into account by adding appropriate weights to our learning models.

5 EXPERIMENTS

In this section, we report an extensive experimental evaluation of our proposed DeepTrust user embedding model and compare its performance against several state-of-the-art algorithms.

Table 2: Precision, recall, and F1-measure results of detecting trustworthy, fraudulent, and unreliable users with DeepTrust and comparison with related work.

Algorithm	Precision	Recall	F1-measure
DeepTrust	**0.93**	**0.93**	**0.93**
DeepTrust w/o context	0.81	0.54	0.58
BAD [14]	0.81	0.40	0.43
F&G [12]	0.90	0.89	0.89
REV2 [11]	0.82	0.64	0.69
Trustiness [20]	0.72	0.26	0.32

5.1 Experimental Settings

We consider three main settings in our experiments: (1) a multi-class problem where we classify users as trustworthy, fraudulent, or unreliable/uninformative; (2) a binary classification problem where we detect *fraudulent* users (vs. trustworthy and unreliable/uninformative), and (3) a binary classification problem where we classify trustworthy vs. untrustworthy (fraudulent and unreliable/uninformative) users. In all the experiments, unknown users are included in the user-item rating network used for computing the temporal review sequence we use to learn the DeepTrust features and for computing the baselines, but unknown users are not used as instances in the classification tasks. However, in Section 5.5, we tackle the problem of classifying the "unknown" users in the dataset into one of the remaining three possible categories and correlate the computed labels with the Amazon rank for additional insights.

We report results for classification with a Random Forest model. We also tested other classification algorithms, including Logistic Regression and Support Vector Machine, but Random Forest resulted in overall best performance.

We used class weighting to deal with class imbalance. Class weighting is a way to learn from an unbalanced dataset where the classification imposes, during training, a penalty proportionally inverse to the class distribution on the model for making classification mistakes. We performed 10-fold cross-validation for all reported experiments.

In regards to evaluation measures, we report weighted precision, recall, and F1-score in the case of multi-class classification. For binary classification, in addition to reporting the above measures to allow comparison, we also calculate the Area Under the ROC curve (AUROC) and Average Precision (AvgP). The best results are highlighted in bold in the tables.

5.2 Detecting Trustworthy, Unreliable, and Fraudulent Users

We tested our DeepTrust user embedding on the problem of classifying users as trustworthy, unreliable, and fraudulent. Table 2 reports the classification results according to precision, recall, and F1-measure for DeepTrust (with and without context) and several state-of-the-art approaches. Our baselines include methods to compute trustworthiness scores for users in opinion-based systems. Specifically, we compare with Bias and Deserve (BAD) [14], Fairness and Goodness (F&G) [12], REV2 [11], and Trustiness [20].

Table 3: Precision, recall, and F1-measure for DeepTrust combined with related work for detecting trustworthy, fraudulent, and unreliable users.

Algorithm	Precision	Recall	F1-measure
DeepTrust	0.93	0.93	0.93
DeepTrust + BAD	0.93	0.93	0.93
DeepTrust + F&G	**0.95**	**0.95**	**0.94**
DeepTrust + REV2	0.93	0.93	0.92
DeepTrust + Trustiness	0.93	0.93	0.93

Table 4: Precision, recall, and F1-measure results of detecting trustworthy, fraudulent, and unreliable *cold start* users with DeepTrust and comparison with related work.

Algorithm	Precision	Recall	F1-measure
DeepTrust	**0.92**	**0.92**	**0.91**
DeepTrust w/o context	0.84	0.34	0.40
BAD [14]	0.02	0.12	0.04
F&G [12]	0.89	0.87	0.88
REV2 [11]	0.82	0.51	0.58
Trustiness [20]	0.001	0.03	0.001

As we can see, our DeepTrust proposed embedding technique consistently outperforms all the other approaches. Among the competitors, F&G achieves the best performance. DeepTrust improves over F&G by 3% in precision and 4% in recall and F1-measure. Moreover, as we can see from the table, the DeepTrust performance significantly drops when removing the context from our user sequences (i.e., not considering the reviews from other users on the set of products reviewed by the given user). This further motivates our choice of considering the user context when computing our embedding.

Next, we combine DeepTrust with any of the existing methods to see if we can further improve our method's performance. To combine DeepTrust with another method, we consider our embedding features and the predictive user features of the other method together in input to the Random Forest classifier. For instance, to combine DeepTrust with REV2, we added to our features the user fairness scores computed by REV2. Table 3 reports comparative results. We see that DeepTrust+F&G yields the best combination, which further improves DeepTrust achieving both precision and recall of 0.95 and F1-measure of 0.94.

Addressing Cold Start Users. We also tested DeepTrust on the problem of classifying users with short or no history (cold start users). In our dataset, we define these "cold-start users" as the users who completed less than four reviews. To perform this experiment, we tested only on cold start users in each test of the 10-fold cross-validation. Results are reported in Table 4 for DeepTrust and baselines, and in Table 5 for the combination.

We see from the results that DeepTrust performance is pretty stable, seemingly due to the user context in the formulation that also allows addressing the cold start user problem. When we compare

Table 5: Precision, recall, and F1-measure for DeepTrust combined with related work for detecting trustworthy, fraudulent, and unreliable *cold start* users.

Algorithm	Precision	Recall	F1-measure
DeepTrust	0.92	0.92	0.91
DeepTrust + BAD	0.92	0.92	0.92
DeepTrust + F&G	**0.94**	**0.94**	**0.93**
DeepTrust + REV2	0.91	0.92	0.91
DeepTrust + Trustiness	0.92	0.92	0.91

results from Tables 2 and 4, we note that that DeepTrust achieves precision, recall, and F1-measure always above 0.91 for *both* general users and cold start ones. Further, we note that not knowing the context results in lower recall performance for the cold start users than for all the users (0.34 vs. 0.54). Specifically, by looking at the individual class recall values, we observe that the recall drastically drops from 0.48 to 0.27 for the class of helpful users. This is because cold start users have just a limited number of reviews, similarly to many fraudulent users. Consequently, benign cold start users can be misclassified as fraudulent. Context information, on the other hand, helps our model with additional information on the ratings of other users to overcome the problem of having a few reviews. This improves the recall.

Baseline methods perform worse than DeepTrust (as expected) and, similarly to what observed before, combining DeepTrust with F&G further improves overall performance by 2% in precision, recall, and F1-measure (see Table 5).

5.3 Detecting Fraudulent Users

For most online platforms, the most damaging category of users is that of fraudulent users who spoil the community posts with fake content. Accordingly, we test our embedding on its ability to detect *fraudulent* users specifically. As this is a binary classification problem, we report Average Precision and AUROC scores in addition to precision, recall, and F1-Measure. Moreover, we compare our proposed DeepTrust with five state-of-the-art algorithms specifically defined for fraudulent user detection in opinion-based systems, namely FraudEagle [2], Bias and Deserve (BAD) [14], SpamBehavior [13], ICWSM'13 [16], and REV2 [11]. We chose these algorithms as they are the top-five best-performing algorithms according to the experiments done for supervised classification in [11] on a similar Amazon dataset. Moreover, we also included the Fairness and Goodness (F&G) algorithm in the comparison as it was not included in the experimental evaluation performed in [11]. Results are shown in Table 6. As we can see, DeepTrust significantly outperforms all the competitors according to all the measures considered on the task of detecting fraudulent users, especially in terms of average precision for fraudulent user detection (+17%). Also in this setting, dropping the context information from the computation of our user embedding decreases the performance. Among the competitors, F&G is the best method according to all performance measures. As we can see from the last row of Table 6, when we combine DeepTrust+F&G we further improve: F1-measure of 0.96, AUROC of 0.97, and average precision of 0.99 (an improvement of 5% in F1-measure,

Table 6: Classification result for fraudulent user detection: precision, recall, F1-measure, AUROC, and average precision (AvgP).

Algorithm	Precision	Recall	F1	AUROC	AvgP
DeepTrust	**0.94**	**0.95**	**0.94**	**0.94**	**0.88**
DeepTrust w/o context	0.84	0.59	0.64	0.72	0.31
REV2 [11]	0.84	0.67	0.71	0.79	0.48
BAD [14]	0.85	0.51	0.56	0.69	0.25
FraudEagle [2]	0.85	0.64	0.69	0.77	0.35
SpamBehavior [13]	0.88	0.87	0.88	0.84	0.57
ICWSM'13 [16]	0.89	0.88	0.88	0.86	0.59
F&G [12]	0.92	0.91	0.91	0.91	0.71
DeepTrust+F&G (Best combination)	**0.96**	**0.96**	**0.96**	**0.97**	**0.99**

6% in AUROC and 28% in average precision as compared to F&G performances). All the other combinations of DeepTrust with baselines achieve worse results than DeepTrust+F&G as reported in Table 8 in the Appendix.

5.4 Classifying Trustworthy vs. Untrustworthy Users

We now analyze the ability of DeepTrust in detecting trustworthy vs. untrustworthy users (fraudulent and unreliable/uninformative) via a binary classification problem. We aim to identify trustworthy users in opinion-based systems so that other users in the platform can rely on their reviews when buying products. In comparing DeepTrust with other works, we consider all the algorithms for trustworthiness and fraudulent user detection used in Sections 5.2 and 5.3. Results are reported in Table 7. Also in this setting, DeepTrust with the user context outperforms all the competitors achieving an F1-measure of 0.93, an AUROC of 0.92, and an average precision of 0.97.

As we can see from the last row of Table 7, when we combine DeepTrust with F&G (which also in this case is the best performing baseline), we further improve the classification: F1-measure of 0.95, AUROC of 0.94, and average precision of 0.98. All the other combinations of DeepTrust with baselines achieve worse results than DeepTrust+F&G as reported in Table 9 in the Appendix.

5.5 Classifying Unknown Users

Finally, we carried an additional experiment attempting to classify users' whose status is "unknown" (see Section 4). We trained our model using trustworthy, unreliable, and fraudulent users and use as test-set the unknown users.

We used our best feature set, i.e., DeepTrust+F&G [3], for training a Random Forest classifier. In order to interpret the quality of our labels, since no other ground truth is available, we relied on Amazon Ranking, and specifically the rank assigned to users. Figure 5 shows the unknown users ordered by the Amazon rank (on the x-axes) along with our prediction: trustworthy users are shown in green, unreliable in yellow, and fraudulent in red. Since the lower the rank, the better the reviewer is, we expect to see in Figure 5 that a higher frequency of trustworthy users are on the left (low rank), unreliable users are mostly in the middle, and fraudulent users are on the right side of the figure (high rank).

[3]We considered the user fairness feature from F&G.

Table 7: Classification result for trustworthy vs. untrustworthy user detection: precision, recall, F1-measure, AUROC, and average precision (AvgP).

Algorithm	Precision	Recall	F1	AUROC	AvgP
DeepTrust	**0.93**	**0.95**	**0.93**	**0.92**	**0.97**
DeepTrust w/o context	0.82	0.58	0.63	0.71	0.89
REV2 [11]	0.82	0.68	0.70	0.77	0.92
BAD [14]	0.83	0.52	0.56	0.68	0.88
FraudEagle [2]	0.82	0.64	0.66	0.73	0.91
SpamBehavior [13]	0.87	0.85	0.86	0.83	0.94
ICWSM'13 [16]	0.87	0.86	0.86	0.84	0.94
F&G [12]	0.90	0.91	0.90	0.88	0.96
Trustiness [20]	0.74	0.61	0.65	0.60	0.85
DeepTrust+F&G (Best combination))	**0.95**	**0.95**	**0.95**	**0.94**	**0.98**

As the figure shows, our prediction follows this pattern very closely. In fact, the top-ranked users are correctly predicted as trustworthy, and the majority of untrustworthy users (fraudulent and unreliable users) appear on the right side. Specifically, 74% of the predicted fraudulent users and 80% of the users predicted as unreliable have a rank higher than 4,000.

Observe that, within our predicted fraudulent and unreliable/uninformative users, some of them (9 fraudulent and 15 unreliable users) rank relatively high, between 1,500 and 2,500. This is suggestive of a possible bias in either our results or in the ranking system itself. While it is not possible to point to a specific error on either side, we note that while Amazon ranks are accepted as strong indicators of the quality of reviewers (and this is consistent with our findings), Amazon ranking method is known to be vulnerable to biases, and fraudulent users may reach high ranks (see Figure 1). We speculate that some of the anomalies in our findings are examples of such users' ability to climb the ranking system.

6 DISCUSSION

This work contributes to the state-of-the-art on trustworthiness detection in opinion-based systems. Our approach is able to detect both trustworthy and malicious users and leverages review traces of users across products. Despite its strengths, our approach is not privy of limitations. We summarize some open issues in what follows.

- *Trustworthiness:* In the context of opinion-based systems, a trustworthy reviewer is a user with a record of well-perceived reviews by readers or testers/users of the items reviewer commented on. Hence, trust in recommender systems (or opinion-based systems) is sometimes defined as "competence" or "confidence," to distinguish it from conventional trust notions wherein trust is based on principals' identities

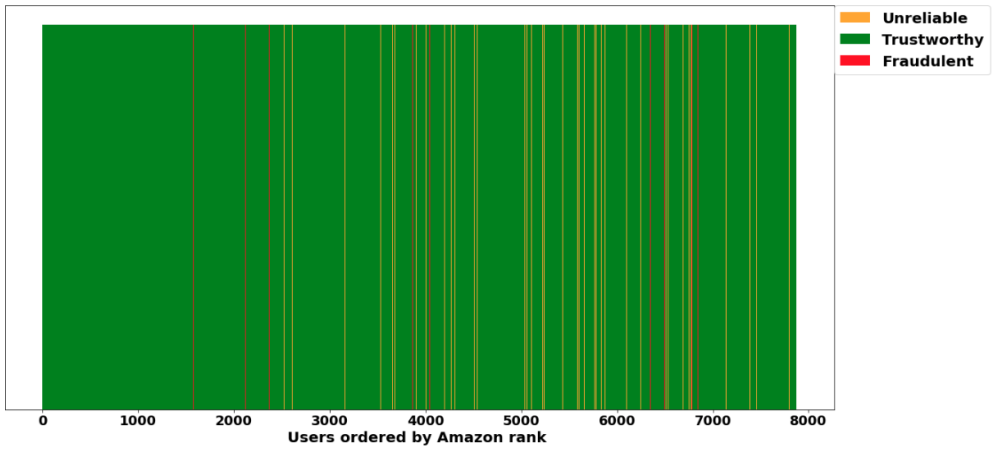

Figure 5: Classification of unknown users as trustworthy (green), unreliable (yellow), and fraudulent (red) and correlation with the Amazon rank.

and credentials [18, 19, 21]. As the definition is often not robust, fraudulent users or users whose behavior changes over time may mistakenly be labeled as trustworthy.

- *Limitations of method:* We have assumed opinion-based systems where the rater and the item being rated are two different entities, e.g., in the case of Amazon, we have users rating products. However, there are other types of opinion-based systems where users rate other users. As an example, in Bitcoin trade networks, users rate the level of trust they have in other users [12]. The definition of temporal review sequence we have given in this work does not fit the case of user-user opinion-based systems as a user u should be considered fraudulent according to how badly she/he is judged by other benign users, rather than how u judges other users. Thus, we should consider u's incoming edges from the user-user rating network to build the temporal review sequence of user u rather than the outgoing edges, as in the case of the user-item rating network. Further, the notion of context should be adapted to the case of user-user opinion-based systems. We plan to investigate this case as future work.
- *Evolution of users:* We have assumed that users do not change status, i.e., they are either malicious or not, with no possible state change. That is to say, the temporal sequence does not reveal an evolving pattern and users are labeled as trustworthy (or otherwise) regardless of the incidence of fake reviews (e.g., if a user posts one fake review she/he is labeled untrustworthy or fraudulent, even if other reviews were actually authentic). This may create some noise in the temporal sequences, in addition to being unrealistic. A soft label approach may be needed to better account for users' changing of behavior.

7 CONCLUSIONS

In this paper, we proposed a supervised approach to identify trustworthy reviewers in an opinion-based system. We presented the problem of detecting trustworthy reviewers as a multi-class classification problem, wherein users may be fraudulent, unreliable or uninformative, or trustworthy. We address the problem by means of a temporal user embedding based on a deep recurrent neural network. We automatically learn relevant features from the input user temporal review sequence in an unsupervised way and use these features for classifying users into trustworthy, unreliable, or fraudulent. Our proposed approach outperforms existing methods under different settings and is able to effectively learn minority classes of users whose behavior is unknown or cannot be learned from the existing traces.

ACKNOWLEDGEMENTS

Serra and Spezzano were partially supported by the Army Research Office under grant W911NF-19-1-0438. Squicciarini was partially supported by the National Science Foundation under grant 1453080.

REFERENCES

[1] [n. d.]. LSTM description. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.
[2] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion Fraud Detection in Online Reviews by Network Effects. In *Proceedings of the Seventh International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, July 8-11, 2013.*
[3] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in neural information processing systems.* 577–585.
[4] Amir Fayazi, Kyumin Lee, James Caverlee, and Anna Cinzia Squicciarini. 2015. Uncovering Crowdsourced Manipulation of Online Reviews. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015.* 233–242.
[5] Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malú Castellanos, and Riddhiman Ghosh. 2013. Exploiting Burstiness in Reviews for Review Spammer Detection. In *Proceedings of the Seventh International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, July 8-11, 2013.*
[6] Jennifer Golbeck, James Hendler, et al. 2006. Filmtrust: Movie recommendations using trust in web-based social networks. In *Proceedings of the IEEE Consumer communications and networking conference*, Vol. 96. 282–286.
[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[8] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. 2016. BIRDNEST: Bayesian Inference for Ratings-Fraud Detection. In *Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, Florida, USA, May 5-7, 2016.* 495–503.
[9] Parisa Kaghazgaran, James Caverlee, and Anna Cinzia Squicciarini. 2018. Combating Crowdsourced Review Manipulators: A Neighborhood-Based Approach.

Table 8: Precision, recall, F1-measure, AUROC, and average precision (AvgP) for DeepTrust combined with related work for detecting fraudulent users.

Algorithm	Precision	Recall	F1	AUROC	AvgP
DeepTrust	0.94	0.95	0.94	0.94	0.88
DeepTrust + BAD	0.95	0.95	0.94	0.95	0.89
DeepTrust + F&G	**0.96**	**0.96**	**0.96**	**0.97**	**0.99**
DeepTrust + REV2	0.94	0.94	0.94	0.95	0.88
DeepTrust + FraudEagle	0.93	0.93	0.93	0.93	0.86
DeepTrust + SpamBehavior	0.93	0.94	0.93	0.92	0.85
DeepTrust + ICWSM'13	0.94	0.94	0.93	0.93	0.87
DeepTrust + Trustiness	0.95	0.95	0.94	0.96	0.89

Table 9: Precision, recall, F1-measure, AUROC and average precision (AvgP) of combining DeepTrust with related work for detecting trustworthy users.

Algorithm	Precision	Recall	F1	AUROC	AvgP
DeepTrust	0.93	0.95	0.93	0.92	0.97
DeepTrust + BAD	0.93	0.93	0.93	0.93	**0.98**
DeepTrust + F&G	**0.95**	**0.95**	**0.95**	**0.94**	**0.98**
DeepTrust + REV2	0.93	0.93	0.93	0.92	0.97
DeepTrust + FraudEagle	0.93	0.93	0.93	0.92	0.97
DeepTrust + SpamBehavior	0.93	0.94	0.93	0.92	0.97
DeepTrust + ICWSM'13	0.94	0.94	0.93	0.93	**0.98**
DeepTrust + Trustiness	0.93	0.93	0.93	0.93	**0.98**

In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*. 306–314.

[10] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. 2003. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*. 640–651.

[11] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and V. S. Subrahmanian. 2018. REV2: Fraudulent User Prediction in Rating Platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*. 333–341.

[12] Srijan Kumar, Francesca Spezzano, V. S. Subrahmanian, and Christos Faloutsos. 2016. Edge Weight Prediction in Weighted Signed Networks. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*. 221–230.

[13] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. 2010. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*. 939–948.

[14] Abhinav Mishra and Arnab Bhattacharya. 2011. Finding the bias and prestige of nodes in networks based on trust scores. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*. 567–576.

[15] Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malú Castellanos, and Riddhiman Ghosh. 2013. Spotting opinion spammers using behavioral footprints. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. 632–640.

[16] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie S. Glance. 2013. What Yelp Fake Review Filter Might Be Doing?. In *Proceedings of the Seventh International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, July 8-11, 2013*.

[17] Shebuti Rayana and Leman Akoglu. 2015. Collective Opinion Spam Detection: Bridging Review Networks and Metadata. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. 985–994.

[18] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. 2003. Trust management for the semantic web. In *International semantic Web conference*. Springer, 351–368.

[19] Sini Ruohomaa and Lea Kutvonen. 2005. Trust management survey. In *International Conference on Trust Management*. Springer, 77–92.

[20] Guan Wang, Sihong Xie, Bing Liu, and Philip S. Yu. 2012. Identify Online Store Review Spammers via Social Review Graph. *ACM TIST* 3, 4 (2012), 61:1–61:21.

[21] Giorgos Zacharia and Pattie Maes. 2000. Trust management through reputation mechanisms. *Applied Artificial Intelligence* 14, 9 (2000), 881–907.

APPENDIX

This appendix reports all classification results when we combine DeepTrust with baselines for the problems of detecting fraudulent users (Table 8) and trustworthy vs. untrustworthy users (Table 9).