INFORMATION SYSTEMS RESEARCH
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 1047-7047 | EISSN 1526-5536 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000

Supplemental Document for "Are Neighbors Alike? A Semi-supervised Probabilistic Collaborative Learning Model for Online Review Spammers Detection"

Zhiang Wu
School of Computer Science, Nanjing Audit University, Nanjing, China, zawu@nau.edu.cn

Guannan Liu*
School of Economics and Management, Beihang University, Beijing, China
MIIT Key Laboratory of Data Intelligence and Management, Beijing, China. liugn@buaa.edu.cn

Junjie Wu
School of Economics and Management, Beihang University, Beijing, China
MIIT Key Laboratory of Data Intelligence and Management, Beijing, China. wujj@buaa.edu.cn

Yong Tan
Foster School of Business, University of Washington, Seattle, WA, USA, ytan@uw.edu

Appendix A: Computational Details of Individual Behavioral Features

Our model has utilized ten individual behavioral features as summarized in Table 1. As discussed in Section 4.1, some features such as *MNR, ANR, NDR and BS* can be obtained from simple statistics such as the number of ratings, while the other features are more intricate that require more computations given the observed ratings of the users. Mathematically, let I_j be the product j, r_{ij} be the rating score of product j given by user i and $\overline{r}_j$ be average rating of product j. We further denote the set of products rated by user i as $R_{i\cdot}$ and the set of users who rated product j as $R_{\cdot j}$. The computational details of the features *ENR, MeanVar, TMF, PopRank, DisAvg, CatEnt* are presented as follows.

- *ENR*: Entropy of the distribution of reviews posted in different days. Let P_d be the ratio of reviews written in the dth day to all reviews, the *ENR* of the user i is,

$$\mathrm{ENR}_i = -\sum_{d=1}^{\mathrm{NDR}} P_d \log_2 P_d.$$

* Corresponding author

- *MeanVar*: Deviation of a user's ratings from the average ratings of all users, which can be computed as,

$$\text{MeanVar}_i = \frac{1}{|R_{i\cdot}|} \sum_{I_j \in R_{i\cdot}} (r_{ij} - \overline{r}_j)^2.$$

- *TMF*: The concentration of highest ratings on a focal proportion of products. Let $R^m_{i\cdot}$ be the set of products to which user i gives the highest ratings, and $U^m_{\cdot j}$ be the set of users who give the highest rating to product j, the *TMF* of the user i can be calculated as,

$$\text{TMF}_i = \frac{\max_{I_j \in R^m_{i\cdot}} |U^m_{\cdot j}|}{\sum_{p \in U^m_{\cdot j}} |R^m_{p\cdot}|}.$$

- *PopRank*: *PopRank* measures the popularity of products that a user has reviewed, which can be measured as the ratio of the total number of ratings received by the products that user i has rated over the number of products that i has given.

$$\text{PopRank}_i = \frac{\sum_{I_j \in R_{i\cdot}} |R_{\cdot j}|}{|R_{i\cdot}|}.$$

- *DisAvg*: Average distance between the ratings given by the user i and other users, which can be computed by the Pearson Correlation Coefficient (PCC) . Let PCC_{ij} be the PCC between the ratings of user i and j.

$$\text{DisAvg}_i = \frac{1}{N} \sum_{j}^{N} (1 - PCC_{ij}).$$

- *CatEnt*: Entropy of the distribution of rated products in different categories. Let P'_k be the ratio of reviews written for products within the kth category, in which 5 product categories (i.e., book, electronics, beauty, education, other) are contained in the Amazon data.

$$\text{CatEnt}_i = -\sum_k P'_k \log_2 P'_k.$$

Appendix B: Network Topological and Embedding Features

We have constructed 7 topological features on the reviewer network, including *degree*, *PageRank score*, *K-core ID*, *graph coloring ID*, *triangle count*, *component size* and *ID*. These topological features are computed by `Python networkx` package[1]. We brief the explanations for the network topological features as follows.

- *Degree*: Since the reviewer network is undirected, we directly compute the total degree of each node as a feature.

- *PageRank score*: This feature is computed by the Page Rank model to indicate the centrality of each node in the network.

- *K-core ID*: The K-core decomposition recursively removes nodes from the graph with degree less than k and the value of k when a node is removed is its K-core ID.

- *Graph coloring ID*: Graph coloring assigns each vertex in the graph to a group in such a way that no two adjacent nodes share the same group, where the color group ID is used as a feature.

- *Triangle count*: We compute the number of triangles (i.e., a complete subgraph of three nodes) in the reviewer network the node participates in as a feature. This feature is an indication of the connectivity of the graph around the node.

[1] `https://networkx.org`

• *Component size* and *ID*: A connected component is a group of nodes such that there is a path between each pair of nodes in the component. We compute the weakly connected component of the reviewer network and extract both the component identifier (ID) and the size of the component that the node belongs to as two topological features for the nodes.

We also utilized the DeepWalk model to extract latent features of the reviewer network, which is a commonly used network representation method to map the node to a low-dimensional embedding space. To implement DeepWalk, we used the source codes in GitHub[2]. We set the number of walks per node as 1, and set the walk length (i.e., the length of sequence by random sampling) as 10. We then tested the number of latent features from 10 to 60 with 10 as the increment, and found almost all algorithms achieved their best performances at 30.

Appendix C: Detailed Experimental Settings

C.1. Manual labeling process for AmazonCN

For our proprietary dataset *AmazonCN*, the spammer labels are obtained through the cooperation with the experts of Amazon China. Since we cannot completely label all the reviewers one by one for unaffordable huge time consumptions, we turn our priority in the manual labeling strategy to ensuring that the (partially) labeled spammers are correct, which is indeed the typical case for real-life spammer detection tasks. In general, we believe the labels of the *AmazonCN* dataset is unbiased because the sampling is not based on a snowball chain sampling in the user-item bipartite graph, but disperses among different communities of the graph. Specifically, we randomly sampled products with more than 15 reviews according to the ASIN number of the products in the Amazon website[3], which cover a wide range of product categories including books, electronics, beauty, etc. It is notable that the ASIN numbers of the products have no correlations in between, and the products were sampled randomly. Then, we proceeded to obtain the users who had ever reviewed the products and further obtained the entire profile pages of the users for manual labeling. Considering that the number of reviewers for each product can be rather large, we employed some strict principles to guide the manual labeling process. That is, we retained the users who had posted more than 20 reviews as our target users; for the users who frequently gave 5-star ratings to the products of a same manufacturer or brand (over 80%), or whose rating behaviors occured mostly within one day (over 80%), we labeled them as highly suspicious; we finally checked each review text posted by the users manually to confirm their legitimacy. In this manner, we obtained 1,029 suspicious spammers after manual labeling. We then sent the labeled suspicious spammers to the department in charge of filtering fake reviews in Amazon China, and their experts helped us to confirm the spammer labels with their internal data including the IP address, the registering time of the reviewers, their monetary expenses, etc. Finally, 985 users labeled as true spammers were returned by the experts in Amazon China. Obviously, the above manual labeling strategy is mainly based on the behavioral features of the reviewers, either disclosed or internal, rather than the reviewer network information.

[2] `https://github.com/shenweichen/GraphEmbedding`

[3] `http://www.amazon.cn`

C.2. Implementation of baseline methods

We compare our proposed method with seven baselines in the task of online review spammers detection. Here, we describe sources and settings for these baseline methods.

• BadRank, presented by Wu and Davison (2005), is a single-class propagation method where "single-class" means a seed set of nodes belong to one class needs to be given. Therefore, BadRank works in a supervised learning manner, because a set of seed spammer nodes are provided. Meanwhile, BadRank runs on the reviewer network and follows the PageRank intuition by iteratively traversing links while constantly updating each node's score. Specifically, let $E(\mathbf{x}_i)$ denote the initial BadRank score of node i in reviewer network, where $E(\mathbf{x}_i) = 1$ for labeled spammer nodes and $E(\mathbf{x}_i) = 0$ for other nodes. Then, the BadRank score of node i is denoted as $\mathrm{BR}(\mathbf{x}_i)$ and is updated by:

$$\mathrm{BR}(\mathbf{x}_i) = E(\mathbf{x}_i)(1-\beta) + \beta \sum_{j=1}^{N_i} \frac{\mathrm{BR}(\mathbf{x}_j^{(i)})}{N_i}. \tag{1}$$

• AntiTrustRank, presented by Krishnan and Raj (2006), is almost the same as BadRank except that the AntiTrustRank score $\mathrm{AR}(\mathbf{x}_i)$ is updated by:

$$\mathrm{AR}(\mathbf{x}_i) = \frac{E(\mathbf{x}_i)(1-\beta)}{N_i} + \beta \sum_{j=1}^{N_i} \frac{\mathrm{AR}(\mathbf{x}_j^{(i)})}{N_i}. \tag{2}$$

In both BadRank and AntiTrustRank, N_i is the degree of node i and the tunable parameter β is set to 0.85.

• QoC+QoL, presented by Zhang et al. (2006), is a dual-class propagation method where the seed set should include labeled nodes belonging to two classes. Then, two kinds of scores QoC and QoL are updated alternatively and the QoC scores are used for ranking spammers. Due to space limitations, we do not introduce computational details of this method. Abbasi et al. (2012) offer a good summary of a number of propagation-based spam detection methods including BadRank, AntiTrustRank and QoC+QoL.

• SpEAGLE, presented by Rayana and Akoglu (2015), is a pMRF-based method and utilizes features constructed on metadata as well as the reviewer-product bipartite graph. It is implemented in MATLAB and the code is available online[4]. For *YelpNYC* data set, features for users, products and reviews are computed and thus the prior potentials are estimated. But for *Amazon* data set, only features for users are provided, which is identical with other compared methods. Note that the naive SpEAGLE works in an unsupervised learning manner.

• SpEAGLE+, also presented by Rayana and Akoglu (2015), is the supervised version of SpEAGLE. Since it requires labels of every review for training, this information is available on the *YelpNYC* data set but unavailable on the *Amazon* data set. Hence, SpEAGLE+ is included for comparison only on *YelpNYC*.

• ColluEAGLE, presented by Wang et al. (2020), is another pMRF-based and unsupervised method which uses the reviewer network and user features as its input. The reviewer network is constructed by their own way (see Section 3.1 of Wang et al. (2020)). At implementation level, if co-reviews are posted within 90 days and the difference of their ratings is smaller than 3, a co-review similarity is computed as shown in Eq. (1) of (Wang et al. 2020). Then, an edge is added if the co-review similarity is greater than 0.6. We use the publicly available Python code provided by authors[5].

[4] `http://shebuti.com/collective-opinion-spam-detection/`

[5] `https://github.com/zhuowangsylu/ColluEagle`

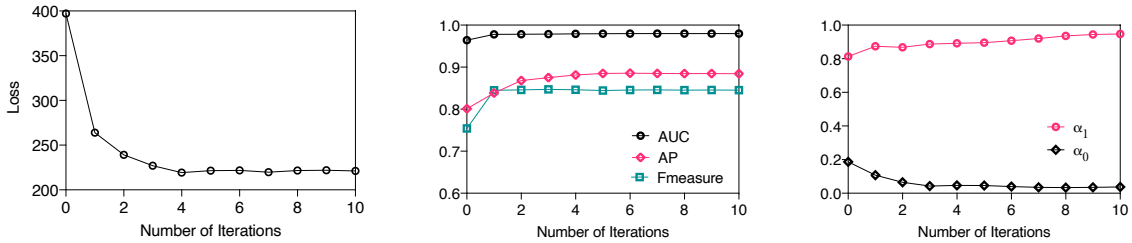


Figure 1 **Model convergence of the proposed method in AmazonCN. (a) Loss with iterations. (b) Detection performances with iterations. (c) Compatibility probability α with iterations.**

• GCNwithMRF, presented by Wu et al. (2020), is designed based on Graph Convolutional Network (GCN) with MRF. The input of GCNwithMRF is same as both ColluEAGLE and our method, i.e., the reviewer network and user features. We feed the same reviewer networks of our method on two data sets to GCNwithMRF. We have carefully tuned parameters on experimental data sets to obtain the best performance. As a result, we set the number of hidden units as 16, the dropout ratio as 0.5, and the number of epochs in each iteration as 500 for *YelpNYC* data set. The GCNwithMRF is implemented in Python of which the source code is provided by the authors[6].

C.3. Reviewer network preprocessing

For *YelpNYC* data set, since the number of users is much greater than the number of restaurants, i.e., 160,225 users but only 923 restaurants, and the reviewing time spans a longer time period. In this case, we consider both the constraints on difference of ratings and posting times in order to avoid the reviewer network being two dense. More specifically, we set a sliding window with a window size of 3 days and it slides from the first day to the last day that users post reviews to this product. For each sliding window, we construct two binary features denoted as w_1 and w_2 where $w_{i1} = 1$ means user i gives positive rating (4-5 star) in this time window and $w_{i1} = 0$ means user i gives negative rating (1-3 star). Then, we are able to represent every user's reviews as a one-hot vector and thus the Jaccard index (J) can be used to measure the co-reviewing similarity of any pair of users with respect to the specific time and rating constraints. We adopt a threshold ϵ to account for the constraint, and edges with $J < \epsilon$ are trimmed and the nodes without any connections are dropped from the network. We tune the threshold in the training set, and set $\epsilon = 0.1$ when it performs the best in the training set to obtain the spammer detection results in the testing set.

Appendix D: Model Convergence

To demonstrate the efficiency and stationarity of the model convergence, we present the changes of the loss value ($\mathcal{L}$), spammer detection accuracy (AP, AUC and F-measure), and one of the key parameter, i.e., the compatibility probability parameter ($\boldsymbol{\alpha}$) during the iterative process for *AmazonCN* in Fig. 1. As can be seen, $\mathcal{L}$, $\boldsymbol{\alpha}$, and the evaluations metrics , have changed abruptly in the first few iterations, but quickly converge to steady values within less than 10, which demonstrate the fast and reliable convergence of the proposed model.

[6] `https://github.com/dleyan/MDGCN/`

Appendix E: Model Regularizers

We have added experiments to empirically validate the effectiveness of model regularizers. Specifically, we add the regularizers for the set of parameters $\boldsymbol{\theta}$ with both L_1 and L_2 norm with a weight γ to regularize the loss function[7], i.e.,

$$\min_{\boldsymbol{\theta},\boldsymbol{\alpha}} \mathcal{L} = \ell_{node} + d\ell_{edge} + \gamma||\theta|| \tag{3}$$

We report the results by tuning the regularizer weight γ to achieve the best results, and the experiments are performed on *AmazonCN* with 10% labeled samples. The results are shown in Figure 2 as follows. As seen from the left subfigure, adding both L_1 and L_2 regularizers can lead to more rapid decrease in the loss function, but the converged loss values are larger than the default settings without the regularizers. Then, we also check the impact of different regularizers on the performances. As shown in the right subfigure, the regularizers do not impact the model performances significantly, but even slightly weaken the performances. We believe it is because our model capacity is not large, which only has limited number of parameters and thus would not incur overfitting issues easily.

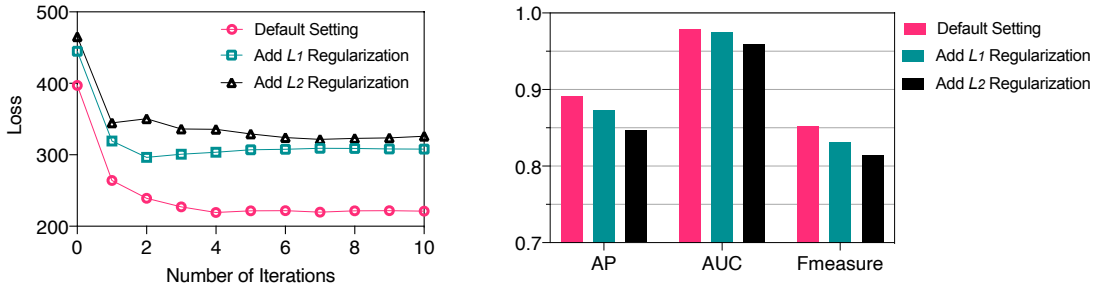
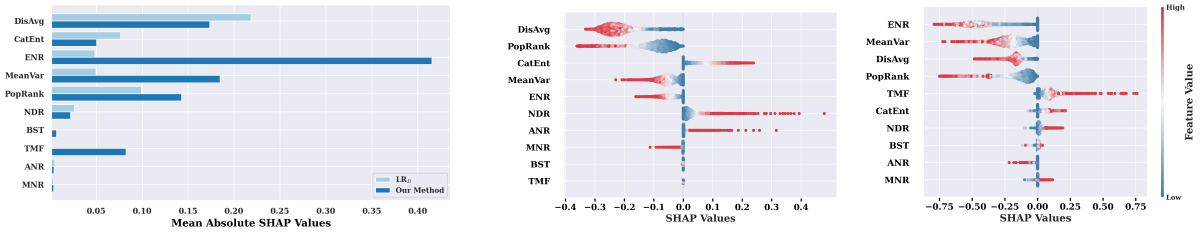


Figure 2 **Influence of regularizers on the model convergence and the performances.**

Appendix F: Feature Importance

We apply one of the most popular machine learning explanation models, Shapley Additive exPlanations (SHAP) (Lundberg and Lee 2017), to gain insights in the behavioral features. In particular, the SHAP value is computed by measuring the marginal contribution of each feature to the predicted value in a data instance level. A feature with more data instances of high absolute SHAP values is deemed more important to the prediction task. To be more specific, to compute the shapley value for each feature, we need to traverse all the feature subsets $S \in 2^F$, where F denotes the number of features of the model. Then, a model should be trained on the feature set S and $S \cup \{l\}$ respectively for the lth feature. Given the model parameters and the prediction function $f(\cdot)$, predictions on the test set can be obtained with and without the feature l, *i.e.*, $f_{S\cup\{l\}}(\mathbf{x}_{S\cup\{l\}}) - f_S(\mathbf{x}_S)$, where $\mathbf{x}_S$ represents the input feature vector on S. Then, shapley values can be computed as a weighted average of all the possible combinations of the feature subsets:

$$\phi_l = \sum_{S\subseteq F\setminus\{l\}} \frac{|S|!(F-|S|-1)!}{F!} \left[f_{S\cup\{l\}}(\mathbf{x}_{S\cup\{l\}}) - f_S(\mathbf{x}_S)\right].$$



(a) (b) (c)

Figure 3 **SHAP values of individual behavioral features. (a) Mean absolution SHAP values. (b) SHAP values using LR_B. (c) SHAP values using our model.**

We conduct comparative analysis of SHAP value under the following two scenarios. The first one is to apply the simple Logistic Regression (LR_B in Table **??**) on the labeled training data, and the second one is to apply our collaborative learning model on both labeled and unlabeled data, and either trained model is then used for computing the SHAP value of each behavioral feature. Fig. 3(b) and Fig. 3(c) show the results, where each dot represents the SHAP value of a data sample in the test set, being either negative (in color blue) or positive (in color red). The mean absolute value can then be obtained by averaging the absolute SHAP values of all the samples, which indicates the aggregate feature importance to spammer detection, as shown in Fig. 3(a).

According to Fig. 3(a), it is obvious that some behavioral features are insignificant in both modelling scenarios, including *NDR*, *ANR*, *MNR* and *BST*. These features are all simple statistics of review posting behaviors within a short time period, e.g., the counts of reviews or days, which can hardly capture the collusive spamming behaviors. Notably, it is interesting to see that features *ENR*, *TMF* and *PopRank*, obtain significantly higher SHAP values when running our model with reviewer network than that in LR. These features characterize the collusive behaviors of spammers; that is, spammers have to grant high ratings to some target products in some specific days to fulfill the orders from their online paymasters, which will result in high *TMF* or low *ENR* and *PopRank* values. In other words, by incorporating the reviewer network in the model, the feature importance of *ENR*, *TMF* and *PopRank* is greatly reinforced due to their very nature in capturing improper profits-driven collusive behaviors.

Finally, two deviation-sensitive features, including *MeanVar* and *DisAvg*, seem also valuable to spammer detection in both modelling scenarios. But their negative SHAP values reveal that the spammers nowadays tend to usurp the role of the public by implementing active collusion ahead of the normal users and the normal users may conform to the prior collusive spammers' ratings (Hong et al. 2016), which in turn makes the spammers' ratings from the other users more alike. In particular, the feature *MeanVar* exerts greater importance in the reviewer network case, which may be due to the fact that the reviewer network can magnify the importance of homophily effects in discriminating the collusive spamming behaviors.

To sum up, while all the individual behavioral features contribute to spammer detection, their contributions are unequal. The features that can disclose spammers' order-driven collusive behaviors seem more important and the reviewer network can reinforce this effect.

[7] Note that $\boldsymbol{\alpha}$ is indeed a probability value and we constrain it between 0 and 1 in the original model, which does not require special regularization.

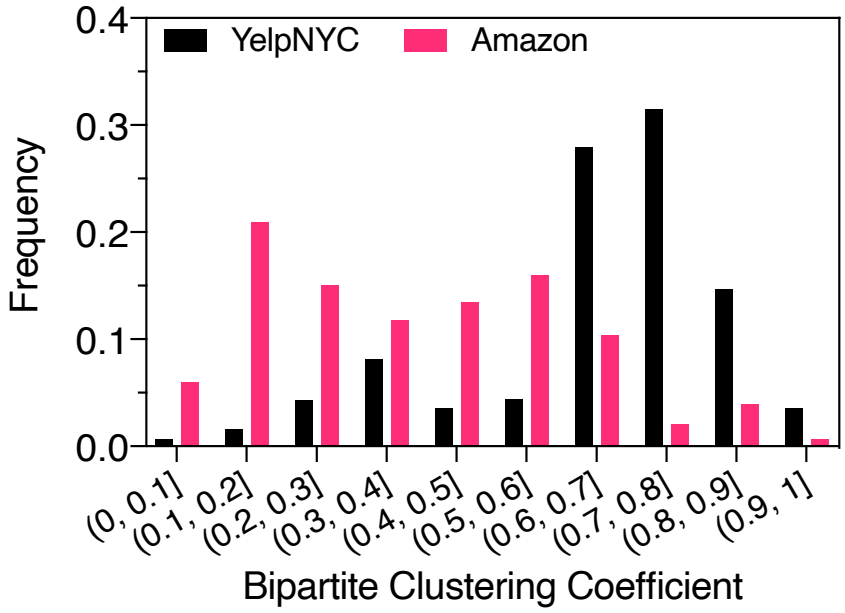


Figure 4 **The distribution of bipartite clustering coefficient for the labeled spammers.**

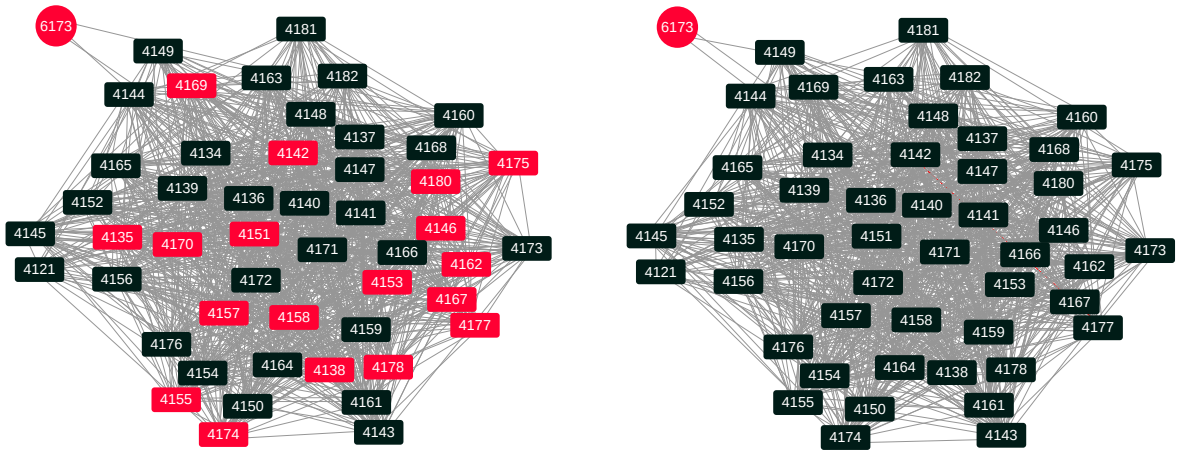
Appendix G: Label distribution in the reviewer network

To validate the difference of label distribution for the two data sets, we exploit the *bipartite clustering coefficient* Latapy et al. (2008) to compare the closeness of the labeled spammers. In particular, the bipartite clustering coefficient of any node denoted as u is defined as:

$$c_u = \frac{\sum_{v \in N(N(u))} c_{uv}}{|N(N(u))|}, \tag{4}$$

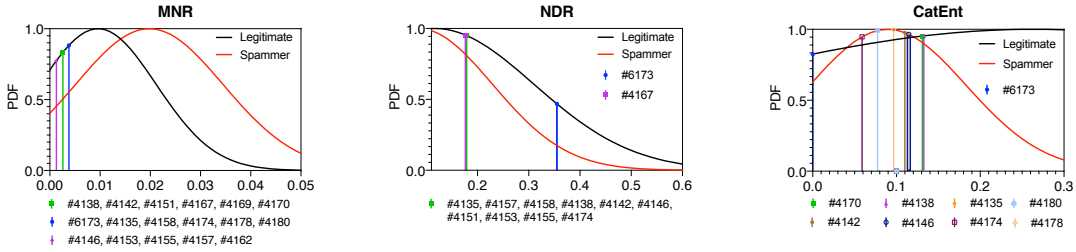
where $N(N(u))$ are the second order neighbors of u in the graph excluding u, and c_{uv} is the pairwise clustering coefficient between nodes u and v which can be computed as, $c_{uv} = \frac{|N(u) \bigcap N(v)|}{|N(u) \bigcup N(v)|}$. In the user-product bipartite graph, since the second order neighbors $N(N(u))$ of any user u is exactly a set of users, we remove legitimate user nodes and their associated edges from the user-product bipartite graph. Hence the bipartite clustering coefficient c_u indicates the average density of connections between a spammer node and its spammer neighbors, where a larger c_u indicates more dense connections between the labeled spammers.

We can then compute the bipartite clustering of each labeled spammer in the two data sets, and show the histogram for the distribution of c_u. As can be seen in Figure 4, the labeled nodes in *YelpNYC* generally have higher bipartite clustering coefficient, and most labeled spammers have a clustering coefficient greater than 0.6, showing that the labeled spammers are densely connected with each other and concentrated within one portion of the network. On the contrary, the clustering coefficients of the labeled spammers in *AmazonCN* distribute more evenly in different intervals, and only a few spammers have high clustering coefficients, which shows that the labels in *AmazonCN* are more evenly distributed and dispersed widely in the network. Obviously, this contrastive network structure between the two data sets indicates the different distribution of labeled spammers in the reviewer networks.



(a) Learning with behavioral features

(b) collaborative Learning with features and reviewer networks



(c) Distribution of representative features

Figure 5 **Community #3 with true and predicted labels in Amazon reviewer network. In (a) and (b), rectangle (circle) node represents true spammer (legitimate), and black (red) node represents accurately (wrongly) detected user. (c) shows distinct feature values of wrongly detected users in (a).**

Appendix H: Visualizing Network Structure and Behavioral Features

In order to better reveal the collective and synergy effects of feature learning along with the reviewer network, we present several "'graph+behavior" examples. One typical example is shown in Fig. 5, where the basic topological structure as shown in Fig. 5(a) and Fig. 5(b) is exactly a community out of the 115 communities in the Amazon reviewer network found by Louvain. As can be seen, there are a great deal of wrongly classified nodes according to predicted labels learned purely on behavioral features (see Fig. 5(a)), but most of them have been corrected by our collaborative learning model based on the reviewer networks (see Fig. 5(b)).

To explain why some users are wrongly classified, we assume the continuous value of each behavioral feature follows a Gaussian distribution and then estimate its probability density function (PDF) based on the ground truth of being spammer or legitimate. We compute the probability density for several representative features and plot their density mass in Fig. 5(c). As can be seen, three features (i.e., CatEnt, NDR and MNR) of most false-negative users (red rectangle nodes in Fig. 5(a)) indicate a higher probability belonging to legitimate reviewers, showing that these spammers try to hide their true intentions and camouflage their spamming behaviors. However, due to the dense connections with the true labeled spammers, their real

identity of being spammers can be discovered through the learning of the reviewer networks, which again advocates the collective learning of both the features and the networks.

References

Abbasi A, Zahedi FM, Kaza S (2012) Detecting fake medical web sites using recursive trust labeling. *ACM Transactions on Information Systems (TOIS)* 30(4):1–36.

Hong Y, Burtch G, Huang N, Li C (2016) Culture, conformity, and emotional suppression in online reviews. *Journal of the Association for Information Systems* 17(11):737.

Krishnan V, Raj R (2006) Web spam detection with anti-trust rank. *AIRWeb*, volume 6, 37–40.

Latapy M, Magnien C, Del Vecchio N (2008) Basic notions for the analysis of large two-mode networks. *Social networks* 30(1):31–48.

Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems 30*, 4765–4774.

Rayana S, Akoglu L (2015) Collective opinion spam detection: Bridging review networks and metadata. *Proceedings of the 21th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining*, 985–994 (ACM).

Wang Z, Hu R, Chen Q, Gao P, Xu X (2020) ColluEagle: collusive review spammer detection using markov random fields. *Data Mining and Knowledge Discovery* 34:1621–1641.

Wu B, Davison BD (2005) Identifying link farm spam pages. *Special interest tracks and posters of the 14th International Conference on World Wide Web*, 820–829.

Wu Y, Lian D, Xu Y, Wu L, Chen E (2020) Graph convolutional networks with markov random field reasoning for social spammer detection. *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 1054–1061.

Zhang L, Zhang Y, Zhang Y, Li X (2006) Exploring both content and link quality for anti-spamming. *The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, 37–37 (IEEE).