Collaborative Shilling Detection Bridging Factorization and User Embedding

Tong Dou[1,2], Junliang Yu[1,2], Qingyu Xiong[1,2(✉)], Min Gao[1,2], Yuqi Song[1,2], and Qianqi Fang[1,2]

[1] Key Laboratory of Dependable Service Computing in Cyber Physical Society, Chongqing University, Ministry of Education, Chongqing 400044, China
{doutong,yu.jl,xiong03,gaomin,songyq,fqq}@cqu.edu.cn
[2] School of Software Engineering, Chongqing University, Chongqing 400044, China

Abstract. The recommender system based on collaborative filtering is vulnerable to shilling attacks due to its open nature. With the wide employment of recommender systems, an increasing number of attackers are disordering the system in order to benefit from the manipulated recommendation results. Therefore, how to effectively detect shilling attacks now becomes more and more crucial. Most existing detection models recognize attackers in statistics-based manners. However, they failed in capturing the fine-grained interactions between users and items, leading to a degradation in detection accuracy. In this paper, inspired by the success of word embedding models, we propose a collaborative shilling detection model, CoDetector, which jointly decomposes the user-item interaction matrix and the user-user co-occurrence matrix with shared user latent factors. Then, the learned user latent factors containing network embedding information are used as features to detect attackers. Experiments conducted on simulated and real-world datasets show that CoDetector has a good performance and generalization capacity and outperforms state-of-the-art methods.

Keywords: Collaborative filtering · Shilling attack · User embedding
Matrix factorization

1 Introduction

Nowadays recommender systems play an important role in dealing with the problem of information overload. In recommender systems, collaborative filtering (CF) is a widely used technique which recommends items according to the assumption that users who have the similar preferences would like to choose similar items. CF model has been proven to be effective but is vulnerable to shilling attacks due to its open nature [1,2]. In shilling attacks, attackers inject user profiles to increase or decrease the recommended frequency of the targeted items, which will reduce the accuracy of recommendation and robustness of a recommender system. As a consequence, they can benefit from the manipulated

I. Romdhani et al. (Eds.): CollaborateCom 2017, LNICST 252, pp. 459–469, 2018.
https://doi.org/10.1007/978-3-030-00916-8_43

results whereas normal users may not trust the system anymore. Therefore, how to detect shilling attacks is a big challenge in the studies of recommender systems.

Generally, shilling attacks can be seen as a binary classification problem, that is, for each user profile, the classified result can only be a normal user or an attacker. Therefore, the key point for this problem is to appropriately design the user features [17]. Most existing detection models recognize attackers in statistics-based manners [5,7]. However, they failed in revealing the fine-grained interactions between users and items, leading to a degradation in detection accuracy when attackers are disguised elaborately.

Matrix factorization (MF) characterizes both items and users by decomposing the user-item rating matrix. Latent factors derived from MF capture the implicit features underlying the interactions between users and items. In this paper, we propose a detection model named **CoDetector** based on MF. However, to further include more information, we bridge basic MF and the word embedding model [4] which can uncover the context structure of users. Inspired by [3], our model jointly factorizes the rating matrix and the user-user co-occurrence matrix with shared user latent factors. For each pair of users, the user-user co-occurrence matrix encodes the number of items they both consumed, which is similar to the word co-occurrence matrix in word-embedding models. The main idea of our model is that attackers tend to promote the target item in group so as to enhance the attack effect [16]. Therefore, factorizing these two matrices can fuse rating preferences and structural information in the user-item bipartite network into the user latent factors, which are the input of the classifier. The experimental results show that CoDetector has a good performance and generalization capacity and significantly outperforms state-of-the-art methods in real scenarios.

The rest of this paper is organized as follows: Sect. 2 reviews the related work of shilling detection. Section 3 focus on the proposed method. In Sect. 4, experimental results of CoDetector on both simulated and real-life dataset are reported. Finally, Sect. 5 concludes our work.

2 Related Work

2.1 Shilling Attack Models

In CF, users who have the similar preferences would like to choose the similar items. Based on this idea and the open nature of recommender systems, attackers can inject biased profiles to manipulate the recommendations [1,2].

In order to behave like a normal user without being detected, malicious users use attack models to generate attacker profiles based on knowledge of recommender systems. The general profile of an attacker can be divided into four segments [1] and is depicted in Fig. 1. I^T denotes target item set which is just what attackers want to promote (push attack) or demote (nuke attack), I^S denotes the selected items based on specific needs of the spam user, I^F denotes filler

item set which is used to disguise attackers, and $I^{\emptyset}$ denotes unrated item set which forms the majority of the profile and are always empty.

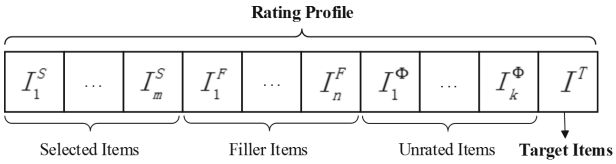

Fig. 1. General framework of attack profile

In accordance with the different chosen items and given ratings in the user profile, attack models are categorized into five types [5]. Table 1 describes these attack strategies.

- **Average attack:** Filler items are assigned the corresponding average ratings of items.
- **Random attack:** Filler items are assigned random values.
- **Bandwagon attack:** Selected items are the frequently rated items and assigned the maximum rating.
- **Segment attack:** Segment attacks choose items similar to target items as selected items.
- **Sampling attack:** The user profile is a copy of that of a normal user.

Table 1. The features of the attack models

Attack model	I^S	I^F	I^T
Random	$\emptyset$	$\beta(i) \sim N(u;\sigma)$, selected randomly	$push : \gamma(i) = r_{max}$ $nuke : \gamma(i) = r_{min}$
Average	$\emptyset$	$\beta(i) \sim N(u;\sigma)$ selected randomly	$push : \gamma(i) = r_{max}$ $nuke : \gamma(i) = r_{min}$
Bandwagon	Widely popular items $\alpha(i) = r_{max}, \forall_i \in I^S$	$\beta(i) \sim N(u;\sigma)$, selected randomly	$push : \gamma(i) = r_{max}$ $nuke : \gamma(i) = r_{min}$
Segment	Similar items to target items $\alpha(i) = r_{max}, \forall_i \in I^S$	$\beta(i) = r_{max}/r_{min}$, selected randomly	$push : \gamma(i) = r_{max}$ $nuke : \gamma(i) = r_{min}$
Sampling	A copy of a exiting users profile		

2.2 Shilling Attacks Detection Methods

According to whether training labels are needed, shilling detection methods can be divided into three categories.

- **Supervised detection model:** This type of detection methods usually recognize attackers by elaborately designing user features. [6] computes rating indicators like DegSim (Degree of Similarity with Top Neighbors) and RDMA (Rating Deviation from Mean Agreement) for all users. [7] extracts popularity patterns based on items analysis.
- **Semi-supervised detection model:** In the real situation, there exist few labeled data and much unlabeled data. Thus, semi-supervised detection models make use of both unlabeled and labeled user profiles for shilling attacks. [10] proposed a model which firstly use naive bayes to train an initial classifier and then improve it with unlabeled data. In [18], a model based on PU-Learning [19] which relies on a few positive labels and much unlabeled data to construct a classifier iteratively was introduced.
- **Unsupervised detaction model:** Compared with supervised detection algorithms, unsupervised ones are more applicable to real scenarios because of less labeled data set. [11] proposed a graph-based detection approach which finds most associated sub-matrices in a user-user similarity matrix for shilling attacks. In [15], the authors exploited the similarity structure in shilling user profiles to separate them from normal user profiles using principal components analysis.

3 Proposed Method

In this section, we propose our model, **CoDetector**, which bridges MF and user embedding to exploit the implicit interactions between users and items.

3.1 Preliminaries

MF and word embedding are pillars of CoDetector. First, we will briefly retrospect these two models.

Matrix Factorization. MF is a basic method in collaborative filtering which uncovers the latent features underlying the interactions between users and items by mapping both users and items into a low-dimensional latent-factor space [12]. The objective function of MF is:

$$L = \sum_{u,i}(y_{ui} - p_u^T q_i)^2 + \lambda(\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2), \tag{1}$$

where user and item latent factors are denoted by $p_u \in \mathbb{R}^d$ and $q_i \in \mathbb{R}^d$ respectively, y_{ui} denotes the observed rating expressed by user u on item i and the algorithmic parameter λ controls the magnitudes of the latent factors.

Word Embedding. Word embedding represents a set of successful models in natural language processing. Using these methods, each word in a sequence of words can be embedded into a continuous vector space. SGNS (the skip-gram neural embedding model) in word2vec [13] is a neural model which trained with

the negative-sampling procedure. [3] proved that SGNS is equivalent to factorizing a word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs. PMI between a word w and a context c is an information-theoretic measure, can be empirically estimated as:

$$PMI(i,j) = \log \frac{\#(i,j) \cdot |D|}{\#(i) \cdot \#(j)} \tag{2}$$

where $\#(i,j)$ is the number of times word j appears in the context of word i, $\#(i) = \sum_j \#(i,j)$ and $\#(j) = \sum_i \#(i,j)$, and $|D|$ is the total number of word-context pairs in the corpora. Then [3] proposed SPPMI (Shifted Positive PMI) based on PMI with different negative samples count k to improve the resulting embedding.

$$SPPMI(i,j) = \max\{PMI(i,j) - \log k, 0\} \tag{3}$$

3.2 CoDetector

Attackers manipulate recommendation results by injecting biased user profiles in a large scale, causing abnormalities not only in the given ratings but also in the local clusters in the user-item bipartite graph. Therefore, to further capture the fine-grained characteristics of attackers, both rating and structural information are supposed to be fused into the user features. In CoDetector, we adopt user embedding to discover the anomalies.

User Embedding. In word2vec [13], given a center word, the sequence of words surrounding it are defined as the context of the center word. Likewise, in the user-item bipartite graph of the recommender system, we can define the context of a user u as other users who consumed or rated same items. For example, both u_1 and u_2 consumed i_1 and i_2, therefore, u_1 and u_2 are contexts of each other. Then, the user-user co-occurrence SPPMI matrix $M \in \mathbb{R}^{m \times m}$ is constructed by computing $\#(i,j)$ that denotes the number of items which both user i and user j consumed. After that, we can obtain user embedding by factorizing M. As attackers tend to promote/demote target items in group, factorizing SPPMI matrix can reveal the implicit interactions among attackers in the user-item bipartite graph and embed structural information into user latent factors. In addition, by tuning the value of negative samples, noises in the bipartite can be neglected whereas available connections are preserved.

Training Procedure. To fuse both rating and structural information into user latent factors, CoDetector jointly decomposes the rating matrix R and the SPPMI matrix M with shared user latent factors. The overall process is shown in Algorithm 1. The objective function of CoDetector is stated as:

$$\begin{aligned} L = &\sum_{u,i} (y_{ui} - p_u^T q_i)^2 + \sum_{u,j} (m_{uj} - p_u^T g_j - w_u - c_j)^2 \\ &+ \lambda (\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2 + \sum_j \|g_j\|^2) \end{aligned} \tag{4}$$

Algorithm 1. The process of CoDetector

Input: User labels U; user−item ratings matrix R, which include attack profiles.
Output: Labels of users to be recognized.

```
Constructing SPPMI matrix M
for user i in U do
    for user j in U do
        Count the number of items both user i and user j consumed.
        Compute the shifted positive point-wise mutual information.
    end for
end for
while notConverged do
    Jointly decompose R and M with shared user latent factors P;
    update latent vectors.
end while
Use P to predict user labels.
```

where p_u is the shared user latent factors which embeds rating and structure information, m_{uj} denotes the shifted positive point-wise mutual information between user u and user j, g_j is the context of user u, and w_u and c_j are the biases of the user and context. The model parameters are updated by using stochastic gradient descent method. The update rules are as follows:

$$\begin{aligned}
&\frac{\partial L}{\partial p_u} = \lambda p_u - (y_{ui} - p_u^T q_i)q_i - (m_{uj} - p_u^T g_j - w_u - c_j)g_j \\
&\frac{\partial L}{\partial q_i} = \lambda q_i - (y_{ui} - p_u^T q_i)p_u \qquad \frac{\partial L}{\partial g_j} = \lambda g_j - (m_{uj} - p_u^T g_j)p_u \\
&\frac{\partial L}{\partial w_u} = m_{uj} - p_u^T g_j - w_u - c_j \qquad \frac{\partial L}{\partial c_j} = m_{uj} - p_u^T g_j - w_u - c_j
\end{aligned} \tag{5}$$

It should be noted that constructing SPPMI matrix is time-consuming due to its $O(n^2)$ complexity. This part should be computed off-line. Fortunately, updates for elements in this matrix is not computationally expensive. For each update, only the cells related to this consumption or click need to be modified.

4 Experimental Results

To evaluate the performance of the proposed algorithm, we conduct experiments on MovieLens and Amazon datesets. Here we show the results of CoDetector in comparison with four state-of-the-art shilling attack detectors and analyze the effect of model parameters.

Datasets including MovieLens and Amazon, are used in our experiments. MovieLens contains 100,000 ratings rated by 943 users on 1,682 movies, and Amazon dataset released by [20] contains 60,000 ratings rated by 4,902 users on 21,394 items. Precision, recall and F1-score were used to measure the performance.

To tune the methods included, we use 80% of the data as the training set, from which we randomly select 10% as the validation set. For the remaining 20% of the data, we consider the users and items that appear in the training and validation sets to obtain the test set. We record the best parameters of these methods according to their performance on the validation set. Afterwards, all the experiments are performed with 5-fold cross validation. The dimensionality d of latent factors in CoDetetor is 10, and negative samples count k is 25 in Sect. 4.1. After 200 iterations, the CoDetector reached a stable result.

Table 2. Detection results on three typical shilling attacks on MovieLens

Attack Size		Filler Size	%3	%5	%7	%10	%15
average	%3	Precision	0.5000	0.7500	0.8000	0.8182	0.8889
		Recall	0.2500	1.0000	1.0000	1.0000	1.0000
		F1	0.3333	0.8571	0.8889	0.9000	0.9412
	%5	Precision	0.6667	0.9091	1.0000	1.0000	0.9600
		Recall	1.0000	1.0000	0.8333	0.7778	0.9917
		F1	0.8000	0.9254	0.9091	0.8750	0.9756
	%7	Precision	0.7857	0.8939	0.8333	0.8889	0.9880
		Recall	1.0000	0.9158	1.0000	1.0000	0.9832
		F1	0.8800	0.9023	0.9091	0.9412	0.9845
random	%3	Precision	0.2500	0.8000	0.9300	0.9057	0.8268
		Recall	1.0000	1.0000	0.9588	0.9023	0.9478
		F1	0.4000	0.8889	0.9442	0.9030	0.8832
	%5	Precision	0.7778	1.0000	0.9550	0.9600	0.9917
		Recall	1.0000	0.8750	1.0000	0.9917	0.9835
		F1	0.8750	0.9333	0.9770	0.9756	0.9876
	%7	Precision	0.9416	0.6667	0.9300	0.7143	0.8182
		Recall	0.9453	1.0000	0.9588	1.0000	1.0000
		F1	0.9421	0.9017	0.9442	0.8333	0.9000
bandwagon	%3	Precision	0.6667	0.8442	0.8000	0.9057	0.8430
		Recall	1.0000	0.9145	1.0000	0.9023	0.9158
		F1	0.8000	0.8779	0.8889	0.9030	0.8779
	%5	Precision	0.8000	1.0000	0.9000	0.8268	0.9565
		Recall	1.0000	0.8571	0.9000	0.9479	0.9483
		F1	0.8889	0.9231	0.9000	0.8832	0.9524
	%7	Precision	0.9426	0.9915	1.0000	0.9416	0.9750
		Recall	0.8928	1.0000	0.8750	0.9453	0.9832
		F1	0.9017	0.9957	0.9333	0.9421	0.9791

4.1 Performance for Detecting Profile Injection Attacks

In this section we firstly evaluate the performance of CoDetector on three different attack models: random attack, average attack, and bandwagon attack, respectively. Then we compare CoDetector with other four methods on the hybrid attack model which contains simulated attackers generated by three attack models and real-world dataset, Amazon, to show the excellent generalization capacity of CoDetector. In MovieLens, we assume that the original users are normal user, and inject simulated attackers manually according to the definition of attack models. In Amazon, normal users and genuine spammers have been labeled by the authors of [20]. In addition. as the principle of push attacks and nuke attacks is almost the same, we only inject attackers to promote the target items.

As can be seen in Table 2, CoDetector is very successful at detecting spam users generated from specific attack models mentioned in Sect. 2. In most cases, with the rise of the filler size and attack size, the values of the measures gradually increase. All of the attackers can be detected. The results confirm the robustness of CoDetector.

Table 3. Comparison of multiple methods

Method		DegreeSAD	FAP	PCASelectUsers	SemiSAD	CoDetector
MovieLens	Precision	**0.9565**	0.9631	0.9062	0.9415	0.9500
	Recall	1.0000	0.9539	0.9667	0.9181	**1.0000**
	F1	**0.9825**	0.9564	0.9355	0.9255	0.9744
Amazon	Precision	0.6880	**0.8943**	0.5465	0.6035	0.8812
	Recall	0.5850	0.7320	0.8852	0.6208	**0.8915**
	F1	0.6324	0.8050	0.6757	0.6120	**0.8863**

Table 3 shows the results of DegreeSAD [7], FAP [14], PCASelectUsers [15], SemiSAD [10] and CoDetector on MovieLens and Amazon. Simulated attacker here are generated with attack size at %15 and filler size at 10%. We can see that CoDetector and other methods have the comparable performance on MovieLens. However, on the real-world dataset, Amazon, CoDetector beats other methods by a fairly large margin. Specifically, the improvements on $F1$ are 40.14%, 10.09%, 31.16%, and 44.82% respectively. It should be noted that DegreeSAD is based on statistical features. By contrast, underlying features fusing rating and structure information in CoDetector can significantly improve the performance.

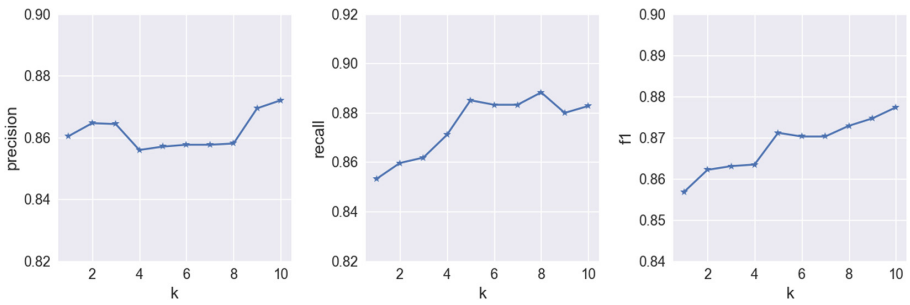

Fig. 2. Impact of the count of negative samples

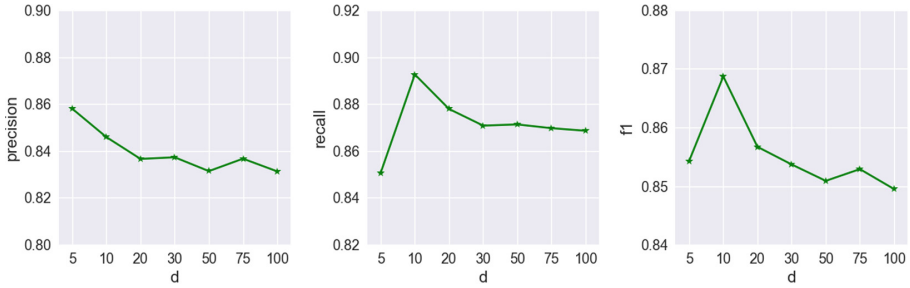

Fig. 3. Impact of parameter d

4.2 Impact of Model Parameters

Two parameters are introduced in CoDetector. One is the dimensionality d of latent factors, the other is the count of negative samples k considered in the construction of SPPMI matrix. Experiments in this part investigate the sensitivity of these two parameters on Amazon. First we fix d at 10 to tune negative samples count ranging from 2^1 to 2^{10}. Figure 2 shows that precision, recall and F1 rise with the increase of negative samples count. We can see in Eq. 3, a larger negative samples count means a more sparse SPPMI matrix. As attackers tend to launch attacks in group, larger co-occurrences between two users means that they may be accomplices. Giving a larger negative count can help to filter noises generated by coincidences.

Then, we fix negative samples count at 25, to see the influence of d ranging from 5 to 100. In Fig. 3, we can see CoDetector obtains best recall and F1 when d=10. However, results on other d values are also acceptable, which validates the stability of our model.

5 Conclusion

In this paper, we present a collaborative shilling detection model called CoDetector bridging factorization and user embedding. It integrates rating and structure information into shared user latent factors to recognize shilling attackers in recommender systems. Experiments conducted on common datasets show that our model significantly outperforms state-of-the-art methods.

Shilling attacks in this paper disorder recommender systems by injecting biased ratings. However, hybrid attacks based on both fake ratings and social connections were proposed [2]. In the future, we will extend our model to detect spammers in the social network.

Acknowledgment. This research is supported by the National Key Basic Research Program of China (973) (2013CB328903), and the Graduate Scientific Research and Innovation Foundation of Chongqing (cys17035).

References

1. Lam, S.K., Riedl, J.: Shilling recommender systems for fun and profit. In: International Conference on World Wide Web, pp. 393–402. ACM (2004)
2. Yu, J., et al.: Hybrid attacks on model-based social recommender systems. Phys. Stat. Mech. Its Appl. **483**, 171–181 (2017)
3. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Advances in Neural Information Processing Systems, vol. 3, pp. 2177–2185 (2014)
4. Mikolov, T., et al. : Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, vol. 26, pp. 3111–3119 (2013)
5. Burke, R., et al.: Classification features for attack detection in collaborative recommender systems. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 542–547. ACM (2006)
6. Chirita, P.-A., Nejdl, W., Zamfir, C.: Preventing shilling attacks in online recommender systems. In: Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management. ACM (2005)
7. Li, W., et al.: Shilling attack detection in recommender systems via selecting patterns analysis. IEICE Trans. Inf. Syst. **99**(10), 2600–2611 (2016)
8. Mobasher, B., Burke, R., Williams, C., Bhaumik, R.: Analysis and detection of segment-focused attacks against collaborative recommendation. In: Nasraoui, O., Zaïane, O., Spiliopoulou, M., Mobasher, B., Masand, B., Yu, P.S. (eds.) WebKDD 2005. LNCS (LNAI), vol. 4198, pp. 96–118. Springer, Heidelberg (2006). https://doi.org/10.1007/11891321_6
9. Burke, R., et al.: Segment-based injection attacks against collaborative filtering recommender systems. In: Fifth IEEE International Conference on Data Mining. IEEE (2005)
10. Cao, J., et al.: Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system. World Wide Web **16**(5–6), 729–748 (2013)
11. Zhang, Z., Kulkarni, S.R.: Graph-based detection of shilling attacks in recommender systems. In: 2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP). IEEE (2013)
12. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
13. Mikolov, T., et al.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
14. Zhang, Y., et al.: Catch the black sheep: unified framework for shilling attack detection based on fraudulent action propagation. In: IJCAI (2015)
15. Mehta, B., Nejdl, W.: Unsupervised strategies for shilling detection and robust collaborative filtering. User Model. User Adapt. Interact. **19**(1), 65–97 (2009)
16. Jiang, M., Cui, P., Faloutsos, C.: Suspicious behavior detection: current trends and future directions. IEEE Intell. Syst. **31**(1), 31–39 (2016)
17. Wu, Z.A., Wang, Y.Q., Cao, J.: A survey on shilling attack models and detection techniques for recommender systems. Chin. Sci. Bull. **59**(7), 551–560 (2014)
18. Wu, Z., et al.: Spammers detection from product reviews: a hybrid model. In: 2015 IEEE International Conference on Data Mining (ICDM). IEEE (2015)

19. Li, X.-L., et al.: Positive unlabeled learning for data stream classification. In: Proceedings of the 2009 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics (2009)
20. Xu, C., et al.: Uncovering collusive spammers in Chinese review websites. In: ACM International Conference on Conference on Information and Knowledge Management, pp. 979–988. ACM (2013)