World Wide Web
https://doi.org/10.1007/s11280-019-00727-4

Co-Detection of crowdturfing microblogs and spammers in online social networks

Bo Liu[1] · Xiangguo Sun[1] · Zeyang Ni[1] · Jiuxin Cao[2] · Junzhou Luo[1] · Benyuan Liu[4] · Xinwen Fu[3,4]

Received: 1 March 2019 / Revised: 6 June 2019 / Accepted: 4 September 2019 /
Published online: 23 October 2019

Abstract

The rise of online crowdsourcing services has prompted an evolution from traditional spamming accounts, which spread unwanted advertisements and fraudulent content, into novel spammers that resemble those of normal users. Prior research has mainly focused on machine accounts and spams separately, but characteristics of new types of spammers and spamming make it difficult for traditional methods to perform well. In this paper, we integrate the study of these new types of spammers with the study of crowdturfing microblogs, investigating the mechanism of crowdsourcing and the close relationship between crowdturfing spammers and microblogs in order to detect new types of spammers and spams more precisely. We propose a novel semi-supervised learning framework for co-detecting crowdturfing microblogs and spammers by comprehensively modeling user behavior, message content, and users' following and retweeting networks. In order to meet the challenge of sparsely labeled datasets, we design an elaborate co-detection target optimal function to minimize empirical error and to permit the dissemination of sparse labels to unlabeled samples. The advantage of this framework is threefold. First, through a deep-level mining of new-type spammers, we aggregate a number of new-found features that can help us make significant distinctions between normal users and new-type spammers. Secondly, by modeling both following networks and retweeting networks, we characterize the essence of the crowdsourcing mechanism abused by spammers in crowdturfing microblog diffusion to markedly increase detection performance. Thirdly, through our optimal function based on semi-supervised methods, we overcome the problem of label sparseness, thus obtaining a more reliable capacity to deal with the challenges of big, sparsely labeled data. Extensive experiments on real datasets demonstrate that our method outperforms four baselines in various metrics (Precision-Recall, AUC values, Precision@K and so on). We also develop a robust system, the functions of which include data collection and availability analysis, spam and spammer detection, and visualization. To render our experiments replicable,

This article belongs to the Topical Collection: *Special Issue on Trust, Privacy, and Security in Crowdsourcing Computing*
Guest Editors: An Liu, Guanfeng Liu, Mehmet A. Orgun, and Qing Li

✉ Bo Liu
bliu@seu.edu.cn

Extended author information available on the last page of the article.

we have made our dataset and codes openly available at https://github.com/sunxiangguo/Crowdturfing.

Keywords Crowdsourcing · Spammer detection · Semi-supervised learning · Online social networks

1 Introduction

Crowdturfing is a relatively new phenomenon; the term combines the familiar "crowdsourcing" and "astroturfing". Crowdturfing gathers large numbers of users to artificially support products, companies, organizations, or opinions. **Crowdturfing microblogs** are sales microblogs created by some advertisers and disseminated widely in online social networks (OSNs) with the help of spammers. In most situations, advertisers place their spam messages in a crowdsourcing platform such as Mechanical Turk[1], Freelancer[2], or Sandaha[3] and hire people dubbed the "Water Army" to spread the message by retweeting, liking, and commenting on various OSNs. Figure 1 depicts how crowdturfing microblogs spread in the Sina platform. These astroturfers are paid according to the amount of work they do; over time, related spams suffuse the OSNs, causing normal users considerable trouble. For example, spammers may simply retweet tremendous numbers of spams in a topic group, even occupying the whole screen of users' smartphones when they swipe to browse news; the platform managers too suffer from these crowdturfing microblogs because the spam messages destroy the quality of the content and may lead to the loss of valuable users.

Unlike the traditional machine-driven spam accounts, these novel spammers are real users similar to normal accounts[23]. Recently, attention has turned to this new type of spammers, but traditional methods of detection mainly focus on machine accounts. In order to detect the new type of spammers and crowdturfing microblogs, there is a critical need for deep-level mining of crowdsourcing mechanisms and clearly understanding the close relationship between crowdturfing spammers and microblogs. A further complication is that most existent datasets are sparsely labeled, which means that supervised learning methods are not equal to the challenges of big data with sparse labels.

Accordingly, we propose a co-detection model to detect both crowdturfing microblogs and spammers. Considering that most datasets are sparsely labeled, we here use a semi-supervised learning method so that our model can fit the current situation better and will have more potential to overcome the challenges of big data. To sum up, the contributions of this work are as follows:

- We detect both spammers and spams (crowdturfing microblogs) based on a semi-supervised strategy. Our co-detection target function guarantees the spread of sparse labels to unlabeled samples so that we can more reliably deal with the challenges of big data with sparse labels.
- We significantly improve performance by integrating retweeting and following networks as well as newfound features embedded in users and content. By means of a deep-level analysis, we uncover a number of useful phenomena in crowdturfing microblogs and spammers.

[1] https://www.mturk.com
[2] https://www.freelancer.com
[3] http://www.sandaha.cc

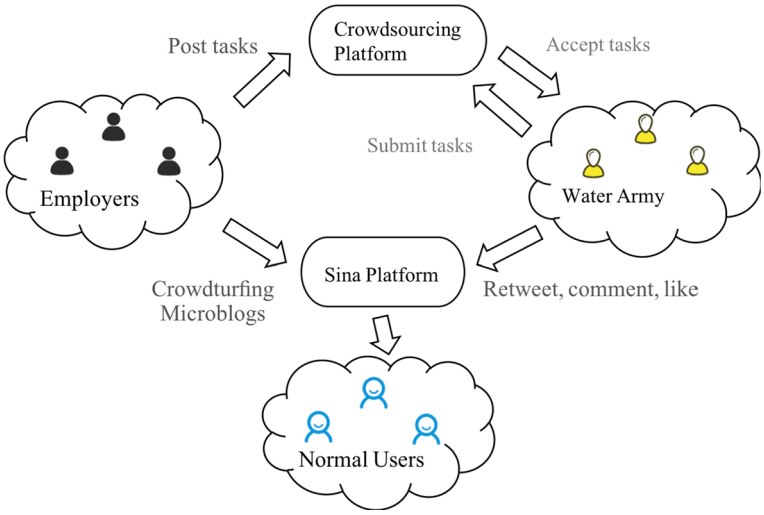


Figure 1 How crowdturfing microblogs spread information. Three kinds of crowdturfing accounts exist: **a** Employers, which originate spam messages and place them into crowdturfing microblogs; **b** the "Water Army," which accepts a task and spreads a message created by employers; and **c** normal users, which as a consequence of spam spread, become the "trash bin" of spams

- Diverging from previous research on traditional spammers, we investigate the new type of spammer to find useful features that will help us significantly distinguish between normal users and spammers.
- Finally, we develop a robust system include multiple social networks data collecting, crowdturfing microblogs and spammer detection, and interface. We also extensively evaluate the performance of our method compared with the most popular baselines. Our results indicate that our model outperforms other baselines in the detection of both spammers and crowdturfing microblogs.

The rest of this paper is organized as follows. Section 2 provides an overview of the related work on spam and spammer detection. Section 3 introduces the processes by which we collected data and conducted the needed analysis. Section 4 formalizes our co-detection problem and explains how we constructed our model. Section 5 presents the CMSC (Crowdturfing Microblogs and Spammers Co-detection) algorithms we used to obtain our solutions. Section 6 illustrates the experimental setup. Our experimental results are extensively reported in Section 7. Then we discuss stability in Section 8 and develop a robust prototype system in Section 9. Section 10 concludes the paper.

2 Related works

Anomaly detection is an emerging and broad area. In online social networks, anomaly detection has two important branches: spammers and spams. Many researches treat spammers as machine accounts[1], volowers (followers providing following services)[15], rumormongers, the water army, and so on. Spams in online social networks often exist in the form of rumors[3, 21], clickbaits[17], and advertisements. In this paper, our target spammers

are advertisement posters in the crowdturfing background, and our target spams are those crowdturfing microblogs.

Early researchers broadly studied spammers, which are mostly machine accounts in online social networks [1, 2, 26]. Machine spammers are usually created and controlled by script programs in batches; hence their published spams are also distinguished by normal contents. With the rise of crowdsourcing services, a new type of spammers, those are real users instead of machine accounts, have gradually emerged [9, 10, 22]. These accounts often have very normal profiles with more hidden features than machine accounts have, making their detection much harder. In order to solve this problem, researchers have tried to detect spams or spammers from three angles: the users' individual attributes, the users' retweeted contents, and following/retweeting links.

Users' attributes are widely employed as the input features for downstream spammer classifiers; such attributes include the number of friend requests [19], logging information, check-ins [11], and sentiment scores [7]. For example, Yuan et al. [27] used a well-designed similarity function to cluster similar accounts into several groups; then they sent users some golden questions and analyzed their responses by comparing them with others in groups. However, this method is insufficient to detect new-type users because they are often organized well and are able to answer questions as normal users would; that is, their individual attributes are often close to those of normal users [6]. Therefore, many related studies have combined users' attributes with content features such as text features, hashtags [14], and the click rate of target tweets. For spam detection, Song et al. [18] considered such user attributes as retweeting time, content features, retweeting users, classifying the messages according to whether they were crowdturfing messages or not. For spammer detection, Yang et al. [25] clustered similar accounts based on their microblog contents and leveraged the topic distribution to isolate suspicious accounts exhibiting characteristics indicative of spammers.

In addition to content features, network link structures are also important clues for spammer detection. Related research [28] has found that spammers often disguise themselves by changing their profiles and their patterns of posting contents. Taking aim at this problem, some studies have tried to integrate a content-based method with link features to make the detection performance more robust. One of the clearest manifestations for following links is malicious following activity [15], meaning abnormal following activities provided by the follower market. In this situation, some users turn to the follower market to buy followers so that they can raise their popularity in a short time. In this vein, Jiang et al. [8] found that there exist some clear distinctions between spammers and normal users in the properties of their following networks, but the structures of following networks in spammers were similar. Therefore, they offered an unsupervised method to model the links features, sending the features to downstream tasks to calculate the degree of suspiciousness. In our earlier works [12, 13], we studied retweeting behaviors in Sina microblogs and constructed a retweeting network such that spammers could be detected based on their retweeting behaviors. In spam detection, researchers also try to integrate users' features, contents features and network features. One of the most useful methods is the graph-based scan approach[20]. For example, Nguyen et al. [20] compute the anomaly scores on entities and relations based on the features of users, contents, and links; then they construct an anomaly graph and translate rumor detection into the task of finding a connected subgraph with maximal anomalousness.

Although a number of meaningful studies have been conducted, there is still considerable potential for improved performance. First, traditional methods have yielded only limited performance in detecting new-type spammers. Second, the mechanism of spammer crowdsourcing still requires much more intensive analysis, especially with relation to

the characteristic topologies of spammers, spams, and normal users. We also notice that most related work on the detection of crowdturfing microblogs and spammers treats them separately, ignoring the close connection between users and their microblogs. In the following sections, we introduce a co-detection model capable of detecting mutual impact between spammers and their spams. Additionally, we consider the attributes of accounts and contents, allowing the method to perform better.

3 Data collection

This section describes our robust system for collecting data from the crowdsourcing and Sina microblogs platforms. We analyze the availability of collected data and implement the work of annotation. The dataset is provided open access at our website for related research.

3.1 Brief description of Sandaha and Sina

As previously mentioned, our "target spammers" are those taking crowdturfing tasks in order to spread advertisements, and the term "spams" refers to the crowdturfing microblogs. In order to collect reasonable data, we investigate the crowdturfing mechanism as it occurs between Sandaha and Sina (see Figure 2). Sandaha is a well-known crowdsourcing website on which many tasks are organized into detailed categories, such as "Sina advertising promotion" and so on. A number of overlapping users exist across these two websites; they can be easily detected because many of them give their corresponding Sina account links in their Sandaha homepages. From one specific category in Sandaha, "advertising promotion for Sina", we collect overlapping users who accept this kind of crowdturfing task. They naturally are spammers we want to research. In contrast to the automated accounts that have been used successfully in the past, these are real users who disguise themselves as normal users to escape detection as spammers. Because spammers are becoming more adept at passing for actual users, a more thorough analysis is urgently needed so that the detection of spammers can be further improved. In the rest of Section 3, we introduce our data collection system and then describe our process of data annotation, followed by our availability analysis.

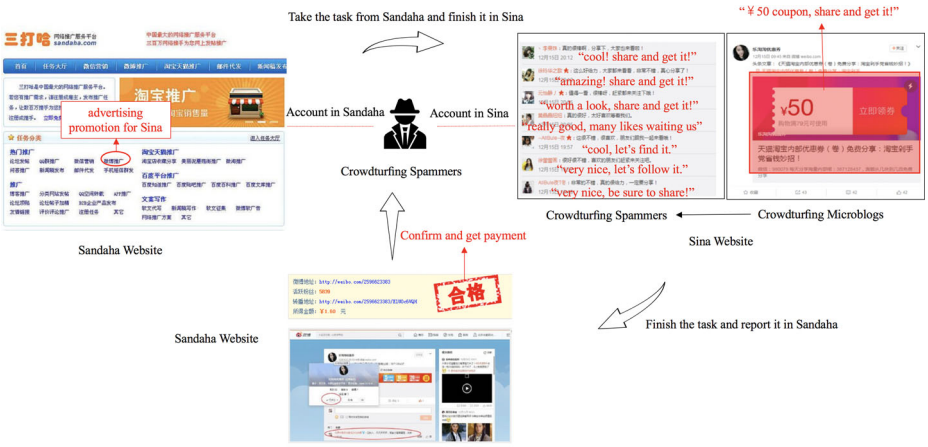


Figure 2 Relations between sandaha and sina in the crowdturfing situation

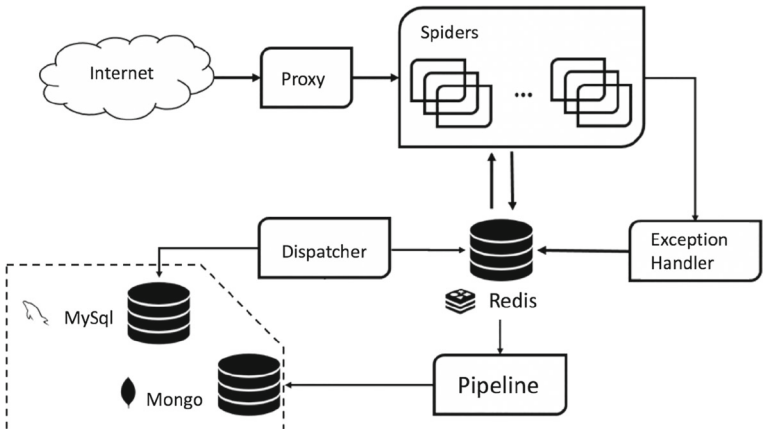


Figure 3 Data collecting system

3.2 Data collection system

Figure 3 presents our data collection system. Basically, multiple tasks were run concurrently in module spiders, allowing the collection of target data from the Internet via proxy service. In case of Internet breakdown, an exception handler module was capable of analyzing the breakpoint, reporting the causes, and regenerating responding tasks. We used Redis to manage the global tasks generated from the collected data, which were stored in Mongodb and MySql. The Module Dispatcher extracted the next task link from the stored data and sent it to Redis.

Using this framework, we first collected 843 spammers and 1,075 crowdturfing microblogs from Sina. We then randomly selected 200 accounts from these spammers and mixed them with another 200 normal accounts to serve as seeds for further data collection. From these seed accounts, we got their friend lists and used the breadth first search method to enlarge our dataset. In the end, we obtained 14,774 user accounts, more than 1,500,000 microblogs published in the preceding three months with 3,680,000 corresponding comments (see Figure 4).

3.3 Data annotations

As shown in Figure 4, in our dataset, we treated all accounts collected from the crowdsourcing platform as spammers. For the remaining unannotated accounts, we randomly selected 3,000 accounts and manually annotated their categories. Basically, if users commented on or retweeted spams several times, and if the contents they commented on or retweeted were very similar to spams, this user was annotated as a spammer. Except for these spammers, we treated other accounts as normal users. In the end, we had 3,883 accounts, including 903 spammers and 2,980 normal users.

The method of spam annotation was similar to that of spammer annotation. Firstly, the microblogs from the crowdsourcing platform were treated as spams. Secondly, we reserved the microblogs with which the above 3,843 accounts interacted, filtering out the messages that had less than 10 retweets or comments. After manual annotation, we had 4,706 microblogs with 479,947 comments. The microblogs included 1,206 crowdturfing microblogs and 3,423 normal microblogs.

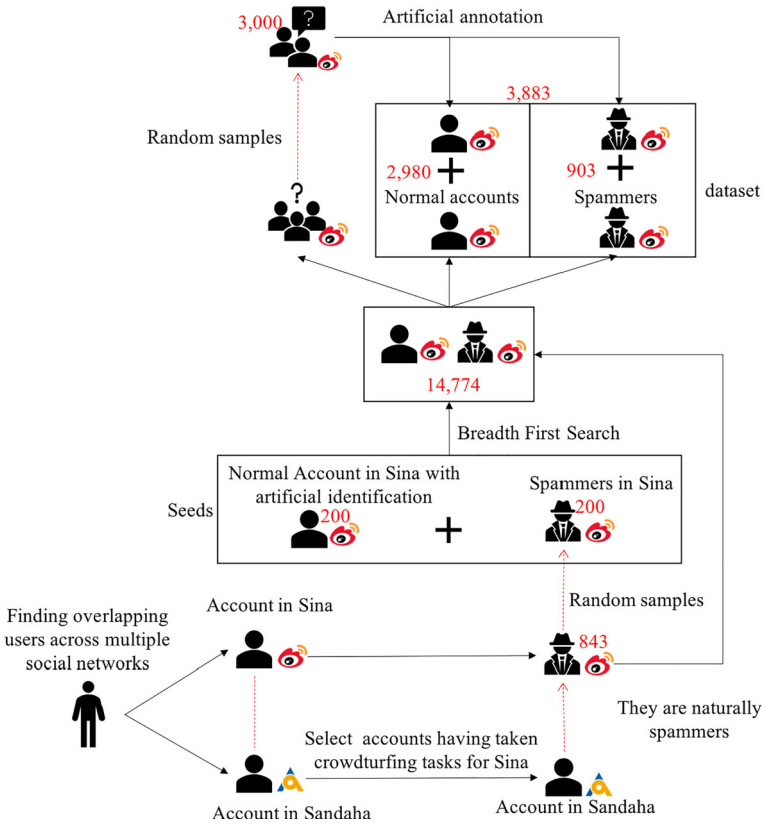


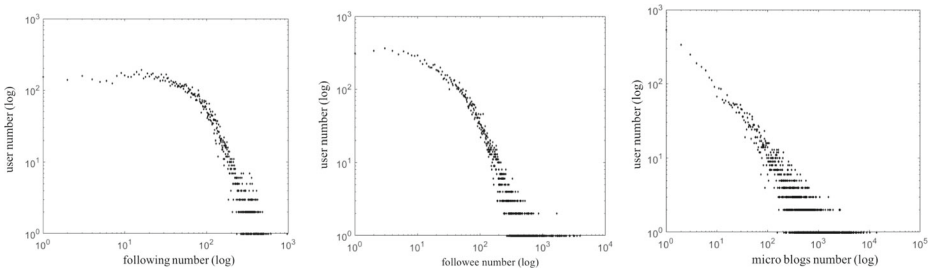
Figure 4 Data collection and annotation with multiple online social. (The process of spams annotation is as same as the spammers, which is not depicted here for simplicity)

3.4 Availability analysis

We investigated the sample distributions with respect to one's following number, followee number, and microblogs' number. In addition, we visualized the fluctuation in the number of microblogs over one week. According to our analysis, our dataset obeys the basic law presented in all social media.

Following behaviors Figure 5a describes the frequency of accounts with regard to their following numbers, from which we can observe that these samples had a long-tailed distribution. It should be noted that this distribution shifts to the right slightly because on the Sina platform, users tend to follow others in order to obtain new messages. A user with almost no following accounts is usually not an active account. A similar pattern is also reflected in Figure 5b, which depicts the frequency of accounts with regard to their follower numbers.

Published microblogs Figure 5c shows the relationship between the frequency of accounts and the numbers of their published microblogs, as well as the power-law distribution. From this relationship, it is clear that most users are not content creators; instead, they tend to



(a) following behavior (b) followee number (c) number of blogs

Figure 5 The following number, followee number and blogs number all follow the law distribution

read or retweet microblogs generated by a small number of users. Figure 6 shows statistical results from the dataset, from which we have three observations: Firstly, there is an obvious periodicity in the number of users' published microblogs from Monday to Sunday. Secondly, there are fewer published microblogs on weekends than there are on weekdays, probably because users have more entertainment options than simply surfing on microblog platforms. Thirdly, there are three minor peaks over the course of a day, at 11 a.m, 2 p.m, and 10 p.m. Evidently, people are typically relaxed during these periods, as shown by their Internet activity.

From the analysis above, we can demonstrate that our dataset is reasonable and is consistent with widely recognized conclusions in the field of complex social networks.

4 Co-Detection model

In this section, we formally define the problem of the co-detection of crowdturfing microblogs and spammers; we then define four primary tasks: abstracting user features,

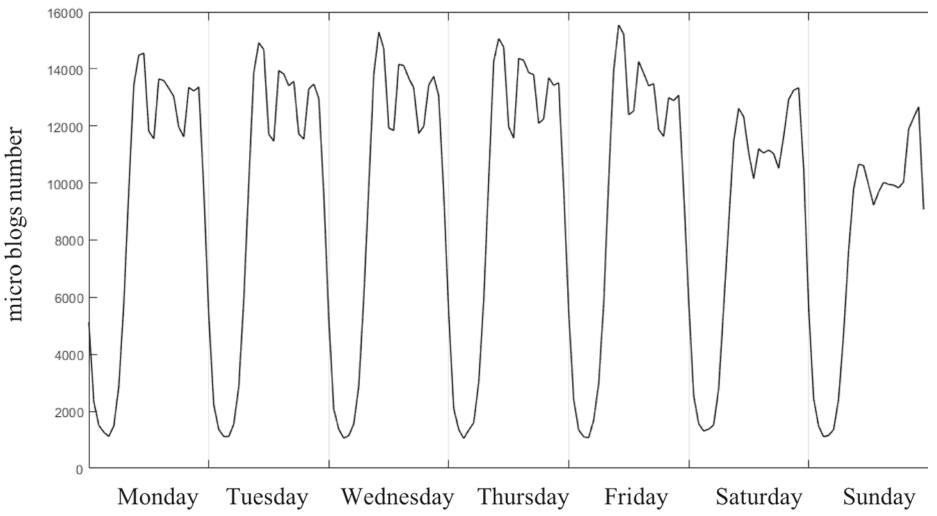


Figure 6 Data distribution of posting messages in a week

abstracting contents features, modeling following networks, and modeling retweeting networks. Finally, we integrate them by optimizing proposed target function.

4.1 Problem definition

For the co-detection of crowdturfing microblogs and spammers, we can construct a heterogeneous network $G = (\mathcal{V}, \mathcal{E}, T_v, T_e, \mathcal{F})$ from the given dataset in OSNs. Here $\mathcal{V}$ denotes the set of vertices and $\mathcal{E}$ denotes the edges in the network. In this scenario, we consider two kinds of vertices: users and messages. Thus the type set of vertex $T_v = \{user, message\}$ and $\mathcal{V} = \mathcal{V}_m \cup \mathcal{V}_u$, $\mathcal{V}_m \cap \mathcal{V}_u = \emptyset$, where $\mathcal{V}_u$ and $\mathcal{V}_m$ denote user set and message set, respectively. The types of edges here include users' following edges, and retweeting edges between users and messages. Thus $T_e = \{following, retweeting\}$ and $\mathcal{E} = \mathcal{E}_f \cup \mathcal{E}_r$, $\mathcal{E}_f \cap \mathcal{E}_r = \emptyset$. We use $\mathcal{E}_f$ and $\mathcal{E}_r$ to denote the following edges and retweeting edges. In addition to the network, each vertex has its own features. Formally, for each $u \in \mathcal{V}_u$, its own features are denoted as a vector $\varphi^u = [\varphi_1^u, \varphi_2^u, \cdots, \varphi_p^u]^T$. Similarly, the individual features of each vertex $m \in \mathcal{V}_m$ are $\varphi^m = [\varphi_1^m, \varphi_2^m, \cdots, \varphi_q^m]^T$. Here, $\mathcal{F} = \{\varphi^u | u \in \mathcal{V}_u\} \cup \{\varphi^m | m \in \mathcal{V}_m\}$.

The problem in this paper can be defined as a classification task where the input is G and the outputs are each vertex's classes. Specifically, for each user vertex $u \in \mathcal{V}_u$, we use x_u to denote its predicted label. $x_u = -1$ means user u is normal account while $x_u = 1$ means this account is a spammer. For each message vertex $m \in \mathcal{V}_m$, we use y_m to denote its predicted label. $y_m = -1$ means that message m has normal content while $y_m = 1$ means that this message is a crowdturfing microblog.

In order to solve the problem defined above, we propose our research framework in Figure 7. Firstly, we quantize some individual features and messages attributes that are highly related to the detection so that we can calculate a priori category for users and messages. Secondly, we consider the following and retweeting networks, and integrate them with users' attributes and microblogs attributes. Thirdly, we put forward an optimal objective that can cause the labels to spread to unlabeled samples. After that, the final category of users and microblogs can be predicted by the downstream classifier.

Because most datasets are sparsely labeled, we split each type of vertex set into two disjointed subsets, that is, $\mathcal{V}_u = \widetilde{\mathcal{U}} \cup \mathcal{U}$, and $\mathcal{V}_m = \widetilde{\mathcal{M}} \cup \mathcal{M}$. Here $\widetilde{\mathcal{U}}$ and $\widetilde{\mathcal{M}}$ denote users and messages with explicit labels, and $\mathcal{U}$ and $\mathcal{M}$ denote users and messages without labels. Finally, the complete network discussed is depicted as Figure 8. In the following

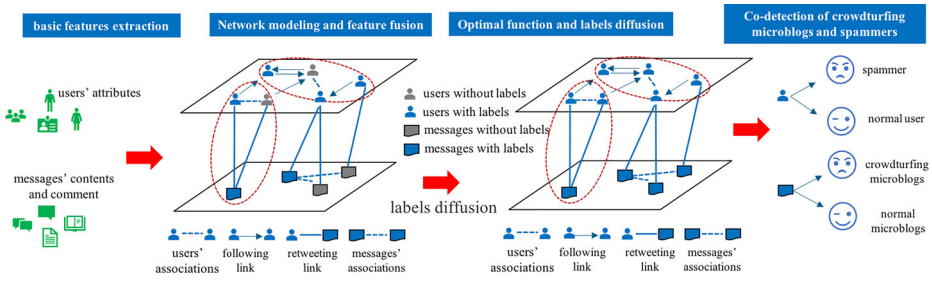


Figure 7 The research framework. In the first stage, some basic features are extracted. Based on these features, we calculate a prior category for users and messages. After that, we fuse the features with the following network and retweeting network, which is the main task in the second stage. In the third stage, an optimal objective is put forward so that the labels can be guided to spread to unlabeled samples. Finally, the system output the final category of users and messages

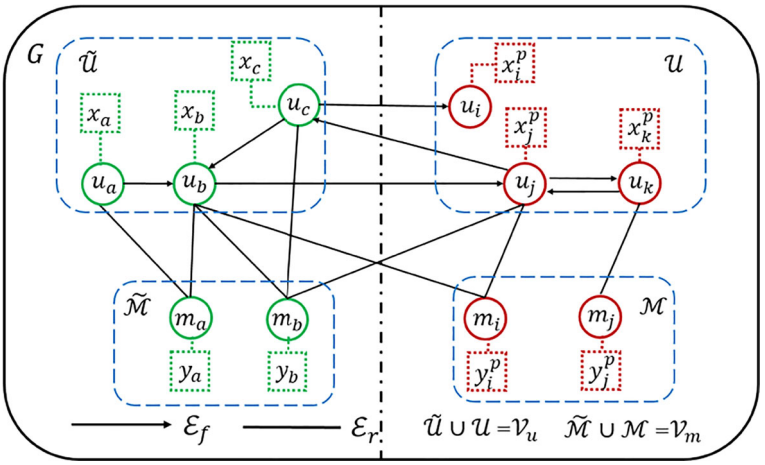


Figure 8 The complete network of users and their messages

subsections, we first calculate a priori category for each vertex by extracting its own features as $\mathcal{F}$. Then for user vertex u_i, its priori category can be denoted as x_i^p and the priori category of message vertex m_i can be denoted as y_i^p.

4.2 Features extraction for users and messages

In order to calculate the priori category for users and messages, we first extract related features from datasets. For users, we consider their account life times, following behaviors, and retweeting behaviors.

Life time In the Sina microblogs platform and many other similar microblogs platforms, a spammer can be reported by normal users, and that account will be closed by the administrator. The registration time of an account can be used to describe the possibility of an account's being a spammer because if an account "survives" for a long time and is still not be closed, the account is less suspected and is more likely to be a normal user. We use φ_l^u to denote the lifetime of user u starting from the time of registration, calculated as:

$$\varphi_l^u = \log(lifetime(u))$$

Following Behaviors In order to obtain greater rewards, spammers often need to have a large number of "friends" in their friend list. Following normal users is easier than being followed by normal users for these spammers. Therefore spammers often follow each other. Taking these into consideration, we use φ_{fe}^u, φ_{fr}^u, and φ_{re}^u to denote the number of followees, followers, and mutual friends for account u, respectively. To be specific, calculated as:

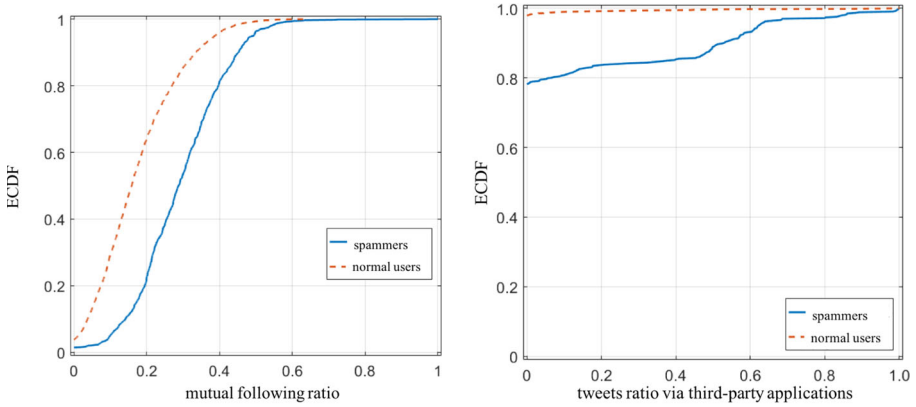
$$\varphi_{fe}^u = \log(followee(u))$$

$$\varphi_{fr}^u = \log(follower(u))$$

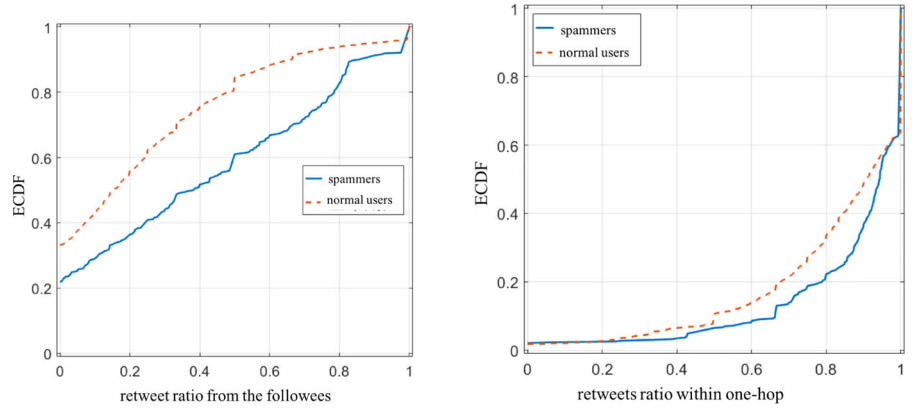
$$\varphi_{re}^u = \frac{e^{\leftrightarrow}(u)}{e(u)}$$

where $e(u)$ is the number of links between u and other users, and $e^{\leftrightarrow}(u)$ is the number of mutual following links between u and other users. We calculate φ_{re}^{u} for spammers and normal users separately in the Sina dataset. The empirical cumulative distribution is drawn in Figure 9a, from this distribution, we can observe that about 20% of spammers have fewer than 20% mutual friends, while about 60% of normal users have fewer than 20% mutual friends. This phenomenon suggests that spammers have a higher percentage of mutual friends than normal users do.

Retweeting Behaviors Compared with normal users, spammers have some distinctive retweeting behaviors. These differences are evident in the statistical results from the real social dataset in Figure 9b, c, and d. Figure 9b shows that more spammers intend to publish spams by third-party applications, while Figure 9c shows that spammers are more likely to retweet messages from their followees. As Figure 9d reports, compared with normal



(a) empirical cumulative distribution for φ_{re}^{u} (b) empirical cumulative distribution for φ_{thi}^{u}



(c) empirical cumulative distribution for φ_{r}^{u} (d) empirical cumulative distribution for φ_{oh}^{u}

Figure 9 Empirical cumulative distribution concerning retweeting behaviors and following behaviors

contents, most spams are retweeted mostly in one hop. Taking these differences into consideration, we define three indexes: φ^u_{thi}, φ^u_r, and φ^u_{oh}, to denote the features above. All these three indexes are elaborated in the following.

The first feature, φ^u_{thi} is the ratio of tweets that are retweeted by user u via third-party applications. Let $wblog(u)$ be the number of microblogs published by user u, and let $wblogThi(u)$ be the number of tweets which are retweeted via third-party applications by user u. Then φ^u_{thi} can be calculated as follows:

$$\varphi^u_{thi} = \frac{wblogThi(u)}{wblog(u)}$$

The second feature, φ^u_r, is the percentage of tweets that are retweeted from user u's followees in his or her total retweets. Let $re(u)$ be the number of tweets retweeted by u, and let $refo(u)$ be the number of tweets retweeted from u's followees. Then we have

$$\varphi^u_r = \frac{refo(u)}{re(u)}$$

The third feature, φ^u_{oh}, describes the percentage of tweets retweeted in one hop. That is,

$$\varphi^u_{oh} = \frac{onehop(u)}{re(u)}$$

where $re(u)$ is the number of tweets retweeted by user u, and $onehop(u)$ refers to the number of tweets retweeted in one hop.

For microblogs, we consider their comments in terms of similarity of content, sentiment polarity, and interaction among different comments.

Content Similarity For the contents of microblog comments, we use φ^m_{sim} to denote the average similarity of comment texts under the same microblog.

$$\varphi^m_{sim} = \frac{2}{|C|(|C|-1)} \sum_{c_k, c_l \in C} cosine(c_k, c_l)$$

where C is the set of comments in the given microblog, and $cosine(c_k, c_l)$ is the cosine similarity between text c_k and text c_l in comments set C.

Sentiment Polarity As comments posted by spammers are very similar, their sentiment polarities are also very stable. Therefore, we use φ^m_{sen} to denote the standard deviation of all comment sentiment polarities, which can be calculated by

$$\varphi^m_{sen} = STD(S_{c_1}, S_{c_2}, \cdots, S_{c_n})$$

where S_{c_k} refers to the sentiment polarity in comment c_k, and $STD(\cdot)$ means the standard deviation of the given samples.

Comments Interaction Since most users that retweet crowdturfing microblogs are spammers, interactions among these microblogs comments are very rare, which is demonstrated particularly by two characteristics: Firstly, the communications between comments (usually meaning responses to comments) are scarce; secondly, the number of likes in these comments is also very low. As Figure 10 depicts, the percentage of normal tweets with comment communication or likes is much higher than it is in crowdturfing microblogs. Based on this observation, we use $reply(m)$ to denote the number of responses to the comments C in a

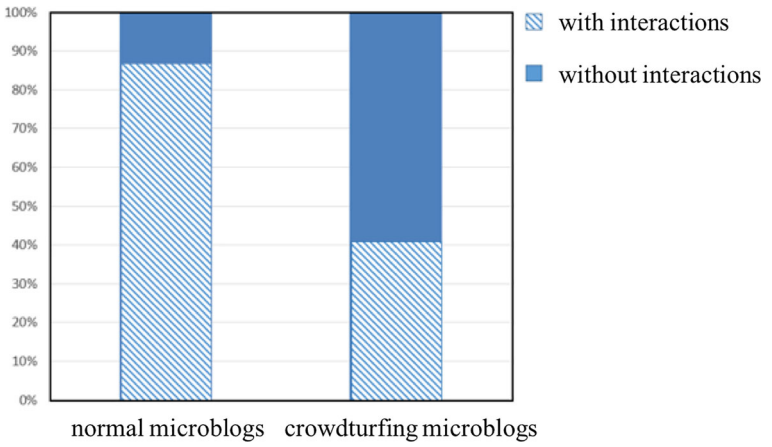


Figure 10 Interaction frequency for normal microblogs and crowdturfing microblogs

given microblog. Let $like(m)$ be the total number of likes in all of the comments of a given microblog. Then we have

$$\varphi^m_{int} = \frac{reply(m)}{|C|}$$

and

$$\varphi^m_{like} = \frac{like(m)}{|C|}$$

Having derived these features, we arrive at the user feature set as

$$\varphi^u = \{\varphi^u_l, \varphi^u_{fe}, \varphi^u_{fr}, \varphi^u_{re}, \varphi^u_r, \varphi^u_{oh}, \varphi^u_{thi}\}$$

and the message feature set as

$$\varphi^m = \{\varphi^m_{sim}, \varphi^m_{sen}, \varphi^m_{int}, \varphi^m_{like}\}$$

For $\varphi^u_i \in \varphi^u$, the item can be normalized by Z-score standardization thus:

$$\varphi^{u*}_i = \frac{\varphi^u_i - \mu}{\delta}$$

where μ and δ are the mean and variance of all items in φ^u respectively. Then the user feature vector can be deduced as $\varphi^{u*} = [\varphi^{u*}_1, \varphi^{u*}_2, \cdots, \varphi^{u*}_p]^T$, $p = |\varphi^u|$. Similarly, the message feature vector can be deduced as $\varphi^{m*} = [\varphi^{m*}_1, \varphi^{m*}_2, \cdots, \varphi^{m*}_q]^T$, $q = |\varphi^m|$.

We now calculate the priori category for users and messages by the logistic regression method. For user u, the probability of being a spammer is

$$\Pr(u = spammer) = \frac{\exp(\omega \cdot \varphi^{u*} + b)}{1 + \exp(\omega \cdot \varphi^{u*} + b)}$$

when $\Pr(u = spammer) > 0.5$, and the priori category for user u is "spammer", which means $x^p_u = 1$. Otherwise, the priori category for user u is "normal account", which means $x^p_u = -1$. Similarly, we can also calculate the priori category for message m, and get its priori category as y^p_m.

4.3 Modeling retweeting networks

The term "retweeting networks" refers to a graph where the vertices are users and messages, and the edges are the links between users and messages they retweet. As Figure 11 depicts, we can define the retweeting network as $G_r = (\mathcal{V}_u, \mathcal{V}_m, \mathcal{E}_r)$. Apparently, G_r is a subgraph of G defined in Section 4.1. Let $R_{n*k} \in \mathbb{R}^{|\mathcal{V}_u| \times |\mathcal{V}_m|}$ be the adjacent matrix of G_r, where $R_{ij} = 1$ if user $u_i \in \mathcal{V}_u$ retweets the message $m_j \in \mathcal{V}_m$ and $R_{ij} = 0$ if u_i does not retweet m_j. In the rest of this subsection, we propose three guidelines for modeling the retweeting network, and we build the corresponding optimization objective as components of the final objective.

Guideline R-1: A user and a message are more likely to be in the same category ($x_i = y_i = 1$ or $x_i = y_i = -1$) if a link exists between them In the situation of crowdturfing, if a user is a spammer, his retweeting tweets are more likely to be spams. At the same time, a crowdturfing microblog is more likely to be retweeted by spammers. Therefore the edge connecting a user and a message is an important clue to judging their categories. In other words, the following target needs to be minimized:

$$\Phi_r = -\sum_{i=1}^{n}\sum_{j=1}^{k} R_{ij} x_i y_j \tag{1}$$

where $n = |\mathcal{V}_u|$, and $k = |\mathcal{V}_m|$.

Guideline R-2: There is a positive correlation between users' categories and users' association strength Users' association strength is defined as $S^u \in \mathbb{R}^{|\mathcal{V}_u| \times |\mathcal{V}_u|}$, where S^u_{ij} is calculated by

$$S^u_{ij} = \sum_{\substack{m_h \in \mathcal{V}_m \\ R_{ih}=1 \\ R_{jh}=1}} \frac{1}{d^m_h} \tag{2}$$

where m_h is a message retweeted both by user u_i and user u_j, and d^m_h is the degree of vertex m_h in G_r. Studying our dataset, we calculate S^u. Our results show that the average

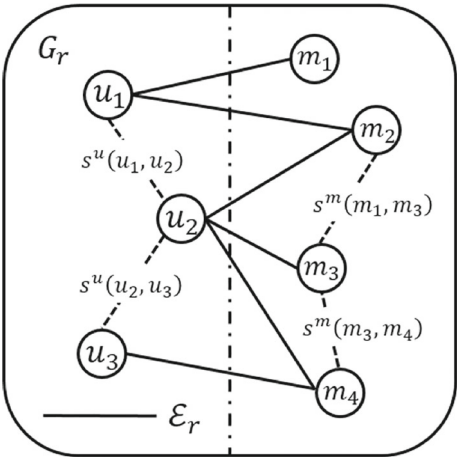


Figure 11 Retweeting network modeling

association strength when users are in the same categories is 0.0994 while the value declines to 0.0067 if users' categories are not the same. Based on these observations, the following target should to be minimized:

$$\Phi_s^u = \sum_{i=1}^{n}\sum_{j=1}^{n} S_{ij}^u \left(\frac{x_i}{\sqrt{d_i^u}} - \frac{x_j}{\sqrt{d_j^u}}\right)^2 \tag{3}$$

Guideline R-3: There is a positive correlation between messages' categories and messages' association strength Like users, the messages also have their association strength, as with $S^m \in \mathbb{R}^{|\mathcal{V}_m|\times|\mathcal{V}_m|}$, where S_{ij}^m is calculated by

$$S_{ij}^m = \sum_{\substack{u_h\in\mathcal{V}_u \\ R_{hi}=1 \\ R_{hj}=1}} \frac{1}{d_h^u} \tag{4}$$

where u_h is a user who retweets both message m_i and message m_j, and where d_h^u is the degree of vertex u_h in G_r. We also calculate S^m in our dataset, from which we found that the average association strength when messages are in the same categories is 0.0774 while the value declines to 0.0298 if messages' categories are not the same. Based on these observations, the following target should be minimized:

$$\Phi_s^m = \sum_{i=1}^{k}\sum_{j=1}^{k} S_{ij}^m \left(\frac{y_i}{\sqrt{d_i^m}} - \frac{y_j}{\sqrt{d_j^m}}\right)^2 \tag{5}$$

4.4 Modeling following networks

The term "following networks" refers to the graph where vertices are all users, and edges represent following relationships between users. As Figure 12 depicts, the following network can be denoted as $G_f = (\mathcal{V}_u, \mathcal{E}_f)$, which is also a subgraph of G. Let $F_{n*n} \in \mathbb{R}^{|\mathcal{V}_u|\times|\mathcal{V}_u|}$, where $F_{ij} = 1$ if user $u_i \in \mathcal{V}_u$ follows user $u_j \in \mathcal{V}_u$ and $F_{ij} = 0$ if u_i does not follow user u_j. One guideline to model the following network is as follows:

Guideline F-1: Two users are more likely to be in the same category if a following edge exists between them in G_f This guideline reflects the observation that spammers often

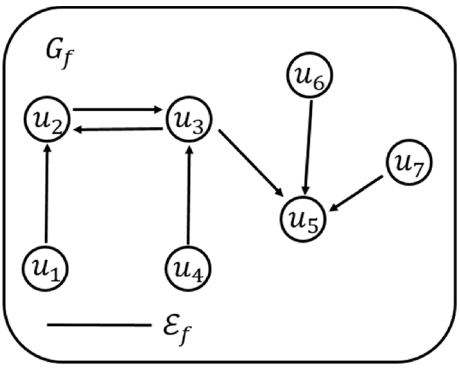


Figure 12 Following network modeling

follow each other so that their "friends" number will be larger. At the same time, however, normal users do not ordinarily follow spammers. According to this, the following target should be minimized:

$$\Phi_f = \sum_{[i,j]\in\mathcal{E}_f} \pi(i) P_{ij} (x_i - x_j)^2 \tag{6}$$

where $P_{ij} = \frac{F_i j}{d_i^{out}}$, d_i^{out} is the out-degree of vertex u_i, and $\pi(i)$ measures the importance of u_i. In the Sina platform, a user with a bigger value of $\pi(\cdot)$ tends to be a VIP account. $\pi(i)$ can be calculated by various methods. In this paper, we use Pagerank[16] to calculate $\pi(i)$, thus:

$$\pi(i) = \eta \sum_{[i,j]\in\mathcal{E}_f} \frac{\pi(j)}{d_j^{out}} + \frac{1-\eta}{|\mathcal{V}_u|}$$

where η is the damping factor in the PageRank algorithm.

4.5 Co-detection optimal target

Based on previous work in 4.2, 4.3 and 4.4, the optimal co-detection objective for the problem defined in 4.1 can be deduced as

$$\begin{aligned} \arg\min_{\mathbf{x},\mathbf{y}} \mathbf{F}(\mathbf{x},\mathbf{y}) \;=\; & \sum_{i=1}^{n} (x_i - x_i^p)^2 + \sum_{i=1}^{k} (y_j - y_j^p)^2 \\ & + \frac{\alpha}{2}\Phi_s^u + \frac{\beta}{2}\Phi_s^m + \gamma\Phi_r + \theta\Phi_f \\ s.t. \quad & \forall u_i \in \widetilde{\mathcal{U}}, x_i = x_i^p \\ s.t. \quad & \forall m_j \in \widetilde{\mathcal{M}}, y_j = y_j^p \end{aligned} \tag{7}$$

Note that for the vertex with explicit labels, x_i or y_j is constant, and the corresponding item $(x_i - x_i^p)^2$ or $(y_j - y_j^p)^2$ is also fixed. Therefore these items are only valid for vertices without labels. In order to simplify the calculation, we add two constraint conditions. Let $x_i^p = x_i$ if user u_i has been labeled and let $y_j^p = y_j$ if message m_j has been labeled. In the following section, we rewrite (7) in matrix form and infer the optimal solution of this target. Then we design an algorithm to solve our target problem.

5 Inference

In this section, we infer the optimal solution of (7), which can be changed in matrix form as:

$$\begin{aligned} \arg\min_{\mathbf{x},\mathbf{y}} \mathbf{F}(\mathbf{x},\mathbf{y}) \;=\; & \|\mathbf{x} - \mathbf{x}^{\mathbf{p}}\|_2^2 + \|\mathbf{y} - \mathbf{y}^{\mathbf{p}}\|_2^2 \\ & + \alpha\|\mathbf{A}\mathbf{x}\|_2^2 + \beta\|\mathbf{B}\mathbf{y}\|_2^2 \\ & - \gamma\mathbf{x}^{\mathbf{T}}\mathbf{R}\mathbf{y} + \theta\mathbf{x}^{\mathbf{T}}\mathbf{L}\mathbf{x} \\ s.t. \quad & \mathbf{C}\mathbf{x} = \mathbf{x}^{\mathbf{p}} \\ s.t. \quad & \mathbf{D}\mathbf{y} = \mathbf{y}^{\mathbf{p}} \end{aligned} \tag{8}$$

where:

- $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T$.
- $\mathbf{x}^{\mathbf{p}} = [x_1^p, x_2^p, \cdots, x_n^p]^T$.

- $\mathbf{y} = [y_1, y_2, \cdots, y_n]^T$.
- $\mathbf{y^p} = [y_1^p, y_2^p, \cdots, y_n^p]^T$.
- $\mathbf{x^T R y} = \sum_{i=1}^{n} \sum_{j=1}^{k} R_{ij} x_i y_j$.
- $C_{n*n} \in \mathbb{R}^{|\mathcal{V}_u| \times |\mathcal{V}_u|}$ is a diagonal matrix where:

$$C_{ii} = \begin{cases} 1 & i \in [1, l_u] \\ 0 & i \in (l, n] \end{cases}$$

Without loss of generality, we number the user vertice with explicit labels from 1 to l_u and the user vertice without labels from $l_u + 1$ to n. Similarly,

$$D_{ii} = \begin{cases} 1 & i \in [1, l_m] \\ 0 & i \in (l, k] \end{cases}$$

where l_m is the number of message vertice which have explicit labels.

As for $\|\mathbf{Ax}\|_2^2$, notice that in (3), $S_{ii}^u = 0$, $S_{ij}^u = S_{ji}^u$. Let $UniqueS^u = \{S_{ij}^u | 1 \le i < j \le n, S_{ij}^u \ne 0\}$ and $n_s = |UniqueS^u|$. Let $S_{k_i k_j}^u$ be the kst element S_{ij}^u in $UniqueS^u$. Then $\mathbf{A}$ is a $n_s \times n$ matrix and $\forall \mathbf{A}_{kt}, 1 \le k \le n_s, 1 \le t \le n$:

$$\mathbf{A}_{kt} = \begin{cases} \frac{\sqrt{S_{ij}^u}}{\sqrt{d_i^u}} & \text{if } t = k_i \\ -\frac{\sqrt{S_{ij}^u}}{\sqrt{d_j^u}} & \text{if } t = k_j \\ 0 & \text{otherwise} \end{cases}$$

Finally, the (3) can be rewritten as matrix form like:

$$\Phi_s^u = \sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij}^u \left(\frac{x_i}{\sqrt{d_i^u}} - \frac{x_j}{\sqrt{d_j^u}}\right)^2 = 2\|\mathbf{Ax}\|_2^2$$

Similarly, let $UniqueS^m = \{S_{ij}^m | 1 \le i < j \le k, S_{ij}^m \ne 0\}$ and $n_m = |UniqueS^m|$. Let $S_{k_i k_j}^m$ be the kst element S_{ij}^m in $UniqueS^m$. Then $\mathbf{B}$ is a $n_m \times k$ matrix and $\forall \mathbf{A}_{kt}, 1 \le k \le n_m, 1 \le t \le k$:

$$\mathbf{B}_{kt} = \begin{cases} \frac{\sqrt{S_{ij}^m}}{\sqrt{d_i^m}} & \text{if } t = k_i \\ -\frac{\sqrt{S_{ij}^m}}{\sqrt{d_j^m}} & \text{if } t = k_j \\ 0 & \text{otherwise} \end{cases}$$

Finally, the (5) can be rewritten as matrix form like:

$$\Phi_s^m = \sum_{i=1}^{k} \sum_{j=1}^{k} S_{ij}^m \left(\frac{y_i}{\sqrt{d_i^m}} - \frac{y_j}{\sqrt{d_j^m}}\right)^2 = 2\|\mathbf{By}\|_2^2$$

Let Π be a diagonal matrix where $\Pi_{ii} = \pi(i)$, then L is the laplacian matrix:

$$L = \Pi + \frac{\Pi P + P^T \Pi}{2}$$

According to [4], L is a symmetric matrix. Therefore:

$$\mathbf{x^T L x} = \sum_{[i,j] \in \mathcal{E}_f} \pi(i) P_{ij} (x_i - x_j)^2$$

Having transformed the final optimal target as matrix form in (8), we now infer the optimal solution. According to[5], the final target can be solved by the strategy of alternating iterative. That is, when solving $\mathbf{x}^{t+1}$ in the $t+1$ iteration, we fix $\mathbf{y}$ as $\mathbf{y}^t$. Then target (8) is equivalent to:

$$\begin{aligned} \arg\min_{\mathbf{x}} \mathbf{F}(\mathbf{x}) \ = \ & \|\mathbf{x}-\mathbf{x^p}\|_2^2 + \alpha\|\mathbf{Ax}\|_2^2 \\ & -\gamma\mathbf{x^T Ry} + \theta\mathbf{x^T Lx} \\ s.t. \quad & \mathbf{Cx} = \mathbf{x^p} \end{aligned} \tag{9}$$

Equation (9) is the convex function. We can get the optimal solution by the Generalized Lagrange multiplier method. The augmented Lagrangian function of (9) is:

$$\begin{aligned} \mathcal{L}_\sigma(\mathbf{x},\lambda) \ = \ & \|\mathbf{x}-\mathbf{x^p}\|_2^2 + \alpha\|\mathbf{Ax}\|_2^2 - \gamma\mathbf{x^T Ry} \\ & +\theta\mathbf{x^T Lx} + \lambda^T(\mathbf{Cx}-\mathbf{x^p}) \\ & +\frac{\sigma}{2}(\mathbf{Cx}-\mathbf{x^p})^T(\mathbf{Cx}-\mathbf{x^p}) \end{aligned} \tag{10}$$

Based on the alternating direction method of multipliers (ADMM), (10) can be update alternately. For simplicity, let $\omega = \frac{\lambda}{\sigma}$, the update strategy is:

$$\begin{aligned} \mathbf{x}_{k+1} \ = \ & \arg\min_{\mathbf{x}} \|\mathbf{x}-\mathbf{x^p}\|_2^2 + \alpha\|\mathbf{Ax}\|_2^2 - \gamma\mathbf{x^T Ry} \\ & +\theta\mathbf{x^T Lx} + \frac{\sigma}{2}\|\mathbf{Cx}-\mathbf{x^p}+\omega_\mathbf{k}\|_2^2 \end{aligned} \tag{11}$$

$$\omega_{\mathbf{k+1}} = \omega_\mathbf{k} + \mathbf{Cx_{k+1}} - \mathbf{x}^p \tag{12}$$

As (11) is a convex function, $\mathbf{x}_{k+1}$ can be derived from its first-order derivative. Let $\mathbf{F}(\mathbf{x}) = \arg\min_{\mathbf{x}} \|\mathbf{x}-\mathbf{x^p}\|_2^2 + \alpha\|\mathbf{Ax}\|_2^2 - \gamma\mathbf{x^T Ry} + \theta\mathbf{x^T Lx} + \frac{\sigma}{2}\|\mathbf{Cx}-\mathbf{x^p}+\omega_\mathbf{k}\|_2^2$, then we have:

$$\begin{aligned} \frac{d\mathbf{F}(\mathbf{x})}{d\mathbf{x}} \ = \ & (\mathbf{2I} + \mathbf{2}\alpha\mathbf{A^T A} + \mathbf{2}\theta\mathbf{L} + \sigma\mathbf{C^T C})\mathbf{x} \\ & -(\mathbf{2x^p} + \gamma\mathbf{Ry^t} + \sigma\mathbf{C^T x^p} - \sigma\mathbf{C^T}\omega_\mathbf{k}) \end{aligned} \tag{13}$$

Let $\frac{d\mathbf{F}(\mathbf{x})}{d\mathbf{x}} = 0$, then we have:

$$\begin{aligned} \mathbf{x}_{k+1} \ = \ & (\mathbf{2I} + \mathbf{2}\alpha\mathbf{A^T A} + \mathbf{2}\theta\mathbf{L} + \sigma\mathbf{C^T C})^{-\mathbf{1}} \\ & (\mathbf{2x^p} + \gamma\mathbf{Ry^t} + \sigma\mathbf{C^T x^p} - \sigma\mathbf{C^T}\omega_\mathbf{k}) \end{aligned} \tag{14}$$

Notice that $(2I + 2\alpha A^T A + 2\theta L + \sigma C^T C)$ can be a nonsingular matrix when α, θ, and σ change to the appropriate values.

Similarly, when solving $\mathbf{y}^{t+1}$, we can fix $\mathbf{x}$ as $\mathbf{x}^{t+1}$, then (8) is equivalent as:

$$\begin{aligned} & \arg\min_{\mathbf{y}} \mathbf{F}(\mathbf{y}) = \|\mathbf{y}-\mathbf{y^p}\|_2^2 + \beta\|\mathbf{By}\|_2^2 - \gamma\mathbf{x^T Ry} \\ & s.t. \quad \mathbf{Dy} = \mathbf{y^p} \end{aligned} \tag{15}$$

The solving process is very similar with the above. From (15), we have:

$$\begin{aligned} \mathbf{y}_{k+1} \ = \ & (\mathbf{2I} + \mathbf{2}\beta\mathbf{B^T B} + \sigma\mathbf{D^T D})^{-\mathbf{1}} \\ & (\mathbf{2y^p} + \gamma\mathbf{R^T x^{t+1}} + \sigma\mathbf{D^T y^p} - \sigma\mathbf{D^T}\omega_\mathbf{k}) \end{aligned} \tag{16}$$

$$\omega_{\mathbf{k+1}} = \omega_\mathbf{k} + \mathbf{Dy_{k+1}} - \mathbf{y}^p \tag{17}$$

Based on the inference above, we design an algorithm on Crowdturfing Microblogs and Spammers Co-detection, which is elaborated in Algorithm 1. Line 1-6 calculate the priori categories for users and messages. In line 7, we initialize $\mathbf{x}$ as $\mathbf{x}^p$ and $\mathbf{y}$ as $\mathbf{y}^p$ so that the algorithm can be converged faster. From line 8 to line 21 the algorithm alternately calculate $\mathbf{x}$ and $\mathbf{y}$ and for either of them, the algorithm update the value of $\mathbf{x}$ or $\mathbf{y}$ iteratively until their values become stable. At last, the algorithm return the final results of $\mathbf{x}^t$, $\mathbf{y}^t$.

Algorithm 1 CMSC algorithm.

Require: user vertice set $\mathcal{V}_u = \widetilde{\mathcal{U}} \cup \mathcal{U}$; users' features set φ^u; message vertice set $\mathcal{V}_m = \widetilde{\mathcal{M}} \cup \mathcal{M}$; messages' features set φ^m, auxiliary matrixs $\mathbf{A}$, $\mathbf{B}$, $\mathbf{L}$, $\mathbf{R}$, $\mathbf{C}$, $\mathbf{D}$; parameters $\alpha, \beta, \gamma, \theta, \sigma$.

Ensure: users' categories $\mathbf{x}$; messages' categories $\mathbf{y}$.

1: **for** all users $u_i \in \mathcal{V}_u$ **do**
2: **if** $u_i \in \widetilde{\mathcal{U}}$ **then** $x_i^p = x_i$
3: **else** $x_i^p = \text{LogisticRegression}(\varphi^{u_i})$
4: **for** all message $m_i \in \mathcal{V}_m$ **do**
5: **if** $m_i \in \widetilde{\mathcal{M}}$ **then** $y_i^p = y_i$
6: **else** $y_i^p = \text{LogisticRegression}(\varphi^{m_i})$
7: Initialize $\mathbf{x_0} = \mathbf{x^p}$, $\mathbf{y_0} = \mathbf{y^p}$, $t = 0$
8: **while** not converged **do**
9: Initialize $k = 0$, $\omega_k = 0$, $\mathbf{x}_0 = \mathbf{x}^t$
10: **while** not converged **do**
11: update $\mathbf{x}_{k+1}$ by (14)
12: update ω_{k+1} by (12)
13: $k = k + 1$
14: $\mathbf{x}^{t+1} = \mathbf{x}_k$
15: Initialize $k = 0$, $\omega_k = 0$, $\mathbf{y}_0 = \mathbf{y}^t$
16: **while** not converged **do**
17: update $\mathbf{y}_{k+1}$ by (16)
18: update ω_{k+1} by (17)
19: $k = k + 1$
20: $\mathbf{y}^{t+1} = \mathbf{y}_k$
21: $t = t + 1$
return $\mathbf{x}^t$, $\mathbf{y}^t$

6 Experimental setup

We extensively evaluated our method with related well-known baselines in various metrics. All experiments were implemented on a PC with Intel Core i7-4770 CPU @3.40GHz and 12GB memory. The development languages used were Python and Java which are editable in platforms such as JetBrains, PyCharm, and IntelliJ IDEA. The dataset is elaborated in Section 3. In the rest of this section, we introduce the experimental metrics and baselines, and then we present the performance in the detection task. Lastly, we analyze the contributions of each factor in our method.

6.1 Evaluation metrics

Evaluation metrics are the Precision-Recall Curve, the ROC Curve, and Precision@K.

Precision-Recall Curve The precision-recall curve shows the relationship between precision and recall where the x-axis shows recall value and the y-axis shows precision value. The area under the precision-recall curve measures an average precision (AP), which is defined as

$$AP = \int_0^1 P(R)d(R)$$

where P is precision and R is recall. The larger the AP value, the better this method performs.

ROC Curve The x-axis of a ROC curve represents the false positive rate (FPR) and the y-axis of a ROC Curve is the true positive rate (TPR). The area under this curve can be calculated as follows:

$$AUC = \int_0^1 T(F)d(F)$$

where T is the true positive rate and F is the false positive rate.

Precision@K We also use the Top-K Ranking metric, Precision@K, to denote the percentage of positive samples after ranking them by the predicted scores. Basically, Precision@K can be calculated as

$$Precision@K = \frac{TP}{K}$$

where TP is the number of positive samples in all Top-K samples.

6.2 Baselines

We compared our proposed method CMSC algorithm with the following baselines:

- **S3MCD:** S3MCD is put forward by [24]. It combines the relationships of users vs. users, messages vs. messages, and users vs. messages, and a target optimal objective based on these relations is then constructed.
- **LR:** Logistic regression is a common basic classification method that has been widely applied to various classification tasks.
- **DetectVC:** DetectVC [15] focuses on the relationship between crowdturfing spammers and their employers in the crowdsourcing platforms; it then uses a random walk method to detect spammers.
- **CrowdTarget:** CrowdTarget [18] focuses on the detection of crowdturfing microblogs. It combines features from the message contents, publishing time, and participants; it then uses the AdaBoost algorithm to detect the spams.

6.3 Parameter settings

All parameter values are selected by the grid search method. Specifically, for our CMSC method, related parameters are set as follows: $\alpha = 0.15, \beta = 0.75, \gamma = 0.04, \theta = 1650$. For the S3MCD method, parameters are set as: $\alpha = 0.1, \beta = 0.1, \gamma = 0.5, \lambda = 0.4$. In the following section, we report the performance of our method and other baselines in various metrics (Table 1).

Table 1 The performance of spammer detection or spam detection w.r.t different factors

Algorithm	AP(spammers)	AUC(spammers)	AP(spams)	AUC(spams)
CMSC	0.7621	0.9069	0.7844	0.8966
CMSC-XP	0.6746	0.8386	0.7841	0.8966
CMSC-YP	0.7535	0.9057	0.5004	0.5766
CMSC-XYP	0.6715	0.8320	0.4965	0.5694
CMSC-SU	0.7196	0.8713	0.7844	0.8966
CMSC-SM	0.7617	0.9060	0.6392	0.8385
CMSC-R	0.7078	0.8953	0.7504	0.8897
CMSC-F	0.7354	0.8869	0.7841	0.8966
CMSC-C	0.7325	0.8980	0.7823	0.8956
CMSC-D	0.7525	0.9040	0.7126	0.8466

6.4 Learning process

We first split the dataset into two disjointed subsets (about 4 : 1). The data in the first subset are fed into our algorithm with labels, while the data in the second subset are fed into our algorithm without labels. We also experimentally change the ratios of these two subsets and analyze the sensitivity in Section 8.

7 Results and analysis

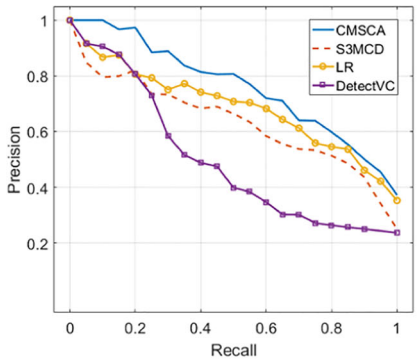
According to the basic setup mentioned in Section 6, we compare our method with other baselines in the Precision-Recall Curve, the ROC Curve, and Precision@K. As reported by these metrics, our method outperforms the others in the dataset.

7.1 Convergence analysis

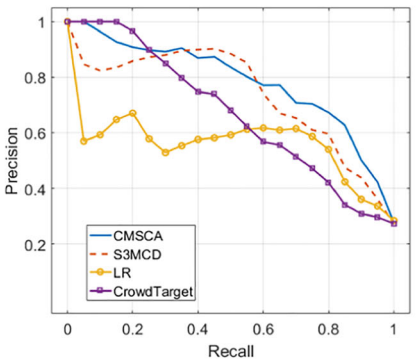
First, we evaluate the computing time of our method from its start to the convergence. The results are reported in Table 2, where the outer loop refers to the Execute Count of line 8 in Algorithm 1, the inner loop of users refers to the Execute Count of line 10 in Algorithm 1, and the inner loop of messages refers to the Execute Count of line 16 in Algorithm 1. We repeat the program several times and get five rows in Table 2, from which we can find that our method can converge in a small number of outer loops and inner loops. The total computing time is also acceptable (from 58s to 75s).

Table 2 Convergence of CMSC Algorithm

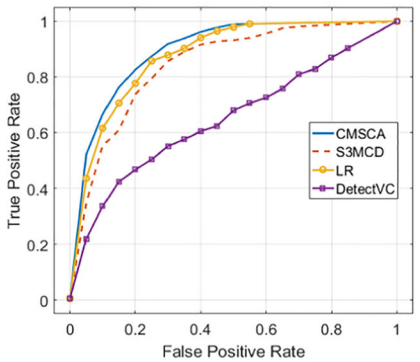
Test	Outer loop	Inner loop(users)	Inner loop(messages)	Computing time
1	4	158	87	65s
2	5	132	80	72s
3	4	137	79	58s
4	7	97	40	64s
5	6	109	68	75s



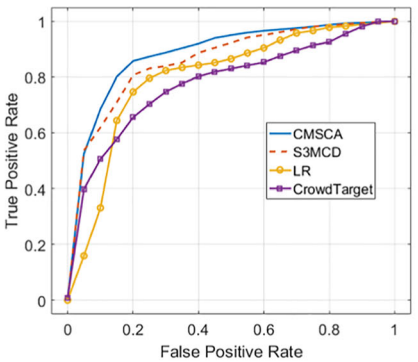
(a) The Precision-Recall Curve for users



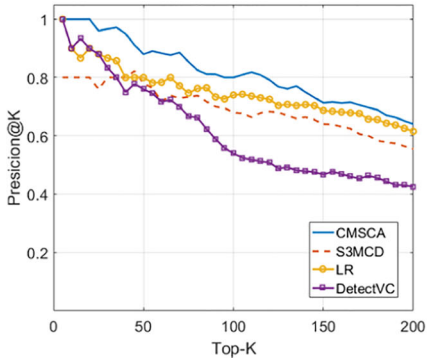
(b) The Precision-Recall Curve for messages



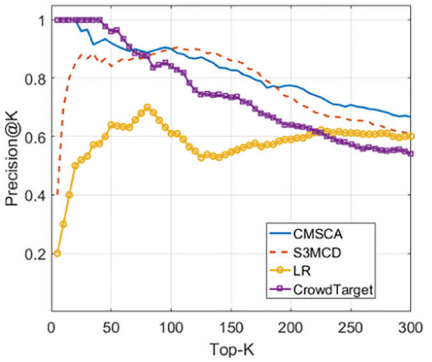
(c) ROC curve for spammers



(d) ROC curve for spams



(e) Precision@K for spamer detection



(f) Precision@K for spam detection

Figure 13 The Precision-Recall Curve for users and messages

Table 3 AP values of different methos

AP	CMSC	S3MCD	LR	DetectVC	CrowdTarget
Users	0.7621	0.6249	0.6889	0.4935	–
Messages	0.7844	0.7345	0.5425	–	0.6604

7.2 PR curve

The Precision-Recall Curve is shown in Figure 13, where Figure 13a shows the performance in spammer detection and Figure 13b shows the performance in spam detection. From the above figures, we are also able to calculate the corresponding AP values, which are listed in Table 3. Note that the DetectVC method is only used for spammer detection and the CrowdTarget method is only used for spams detection. These values demonstrate the distinct superiority of our method. Figure 13 and Table 3 yield the following observations: First, in terms of spammer detection, our method outperforms compared baselines by an average 28.8% in AP values. In terms of spam detection, ours lead the other baselines by an average 13.9%.

Specifically, CMSC performs much better than LR, which suggests that considering the topological properties in retweeting networks and following networks can truly improve the performance in the given task, since in this experiment, the LR method only considers user features and message features, ignoring the network topological properties.

As for the S3MCD method, it primarily considers users' individual features and content features; however, the new types of spammers and crowdturfing microblogs are insensitive to these traditional features, so the final results are not quite as good. In addition, in the curves of S3MCD in Figure 13a and b, especially in the spans where the Recall values ranges from 0-0.2, the curves both have low ebbs and then come up. This phenomenon occurs because in S3MCD needs ranked users or messages and returns top-k results; however, some normal users or messages also have relatively high ranks. With K values enlarge, this shortcoming is gradually alleviated, and the performance then recovers somewhat.

The DetectVC method performs the worst in spammer detection, which suggests that considering the following relationship only, as is done in the DetectVC method, is not sufficient for final detection results. In addition, if our method is compared with the CrowdTarget method, it is clear that considering only messages without spammers is also not sufficient for optimum results.

7.3 ROC curve

We also calculate the AUC values and draw ROC curves for the different methods. In Table 4, CMSC gets 0.9069 for spammer detection and 0.8966 for spam detection, leading by 2% and 3% over the second, respectively. As depicted in Figures 13c and d, CMSC

Table 4 AUC values for different methods

AUC	CMSC	S3MCD	LR	DetectVC	CrowdTarget
Users	0.9069	0.8528	0.8800	0.6628	–
Messages	0.8966	0.8684	0.8095	–	0.7863

Table 5 Precision@K for spammers (%)

Top-K	10	20	30	40	50	100	150	200
CMSC	100	100	97	95	88	80	71	64
S3MCD	80	80	80	80	78	68	64	56
LR	90	90	87	80	80	74	69	62
DetectVC	90	90	87	78	78	56	48	44

performs best both for spammer detection and for spam detection. The LR method comes in second for spammer detection and S3MCD comes in third. For spam detection, S3MCD performs better than the other baselines but worse than our method.

7.4 Precision@K

In this subsection, we calculate the Precision@K values when K ranges from 10 to 200 for spammer detection and from 10 to 300 for spam detection. The results are shown in Tables 5 and 6. According to these results, we draw the corresponding trend charts in Figure 13e and Figure 16h.

For spammer detection, when K is less than 20, CMSC performs at 100% in precision and then gradually declines when K gets bigger. This means that when K becomes larger, some normal users and messages sneak into our ranks. Even so, our method is still ahead of the other methods. For spams detection, the S3MCD and LR methods both improve sharply when K ranges from 0 to 50, which suggests that the results from these methods consist of some normal messages, but S3MCD and LR rank them very high. When K ranges from 100 to 180, we notice that S3MCD performs better than our CMSC method, which reflects that these microblogs contain a large number of advertisements and that S3MCD does better in features extraction from contents. CrowdTarget initially performs better than CMSC, because CrowdTarget explores the characteristics of this part of marketing microblogs more fully. However, when $K > 60$, its performance declines substantially, showing that the features CrowdTarget extracts are not suitable for all the crowdturfing microblogs and the method is not robust enough. Notably however, our CMSC method perform relatively well in the whole range of K, especially when $k > 200$, demonstrating that our method is more robust. Besides, it reliably stays ahead of LR, showing that considering the topological properties of retweeting networks is truly useful for improving the method's performance.

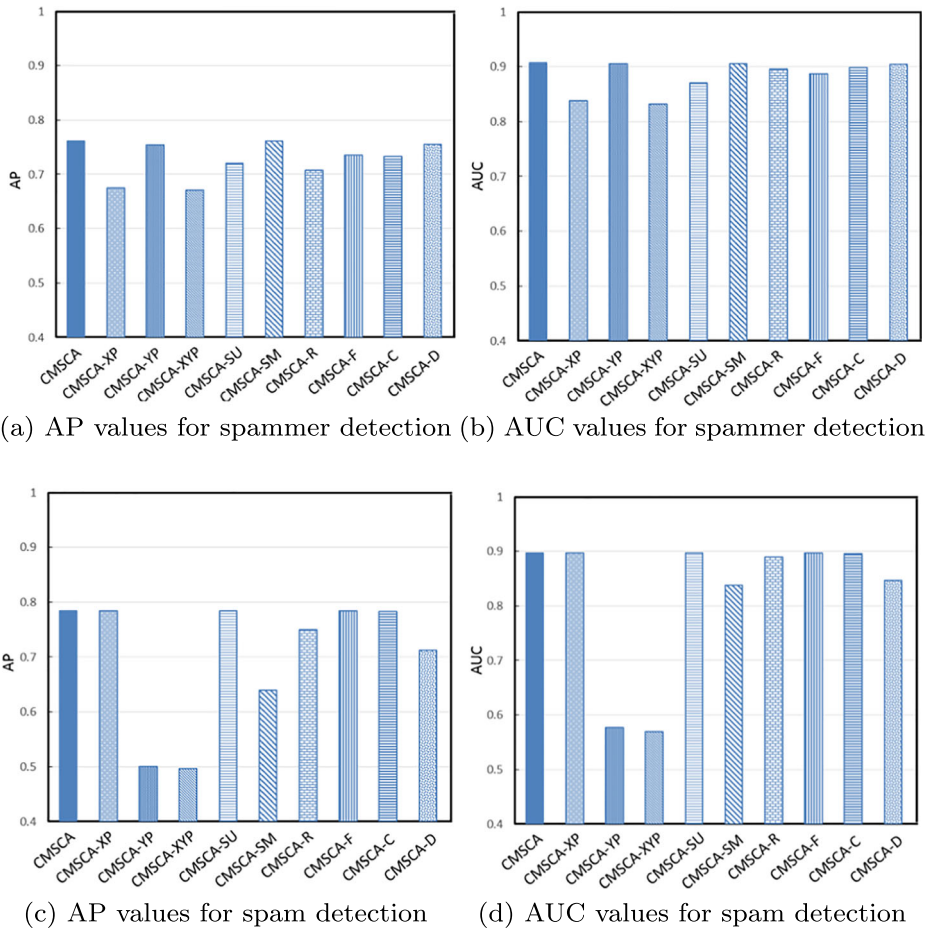
Table 6 Precision@K for spams (%)

Top-K	10	20	30	40	50	100	200	300
CMSC	100	100	97	93	92	90	78	67
S3MCD	70	85	87	85	84	90	74	61
LR	30	50	53	58	64	61	59	60
CrowdTarget	100	100	100	100	96	84	64	54

7.5 Contributions of different factors

Finally, we analyze the contributions of different factors in our method. To review, in our method, we consider the following aspects: the prior categories of users ($\mathbf{x^p}$) and messages ($\mathbf{y^p}$), users' association strength (defined in (3)), messages' association strength (defined in (5)), the relationship between users and their retweeted messages (defined in (1)), users' following relationships (defined in (6)), and the constraint conditions of labeled users (defined as $\mathbf{Cx} = \mathbf{x^p}$ in (8)) and labeled messages (defined as $\mathbf{Dy} = \mathbf{y^p}$ in (8)). We remove some of them and get the following variants of our method:

- **CMSC:** the complete method combining all aspects mentioned above.
- **CMSC-XP:** removing users' prior categories.
- **CMSC-YP:** removing messages' prior categories.



(a) AP values for spammer detection (b) AUC values for spammer detection

(c) AP values for spam detection (d) AUC values for spam detection

Figure 14 The contribution results of different aspects in the detection of spammers and spams

- **CMSC-XYP:** removing both users' and messages' prior categories.
- **CMSC-SU:** removing users' association strength.
- **CMSC-SM:** removing messages' association strength.
- **CMSC-R:** removing the relation between users and their retweeted messages.
- **CMSC-F:** removing users' following relationships.
- **CMSC-C:** removing the constraint conditions of labeled users.
- **CMSC-D:** removing the constraint conditions of labeled messages.

We repeat the test for all of the variants and their performances are listed in Table 1. According to these results, we draw Figure 14 to visualize the importance of the different aspects in the detection of spammers and spams; from that we draw the following observations:

- For spammer detection, when the priori category is removed from the CMSC, the method's performance goes down by 12% (from 0.7621 to 0.6715 in AP value), and 7.5% (from 0.9069 to 0.8386 in AUC value). Similarly, this decline is observed for spam detection (from 0.7844 to 0.5004 in AP value and from 0.8966 to 0.5766 in AUC value). This observation suggests that only considering relationships embedded in the following or retweeting networks in far from addresses the target task. It is therefore importance to combine the features of users and messages.
- The association strengths (messages vs. messages, and users vs. messages) proposed in this paper are also essential for spammer or spam detection because when we remove the association strength, the corresponding performance also go down significantly.
- When the relationship between users and their retweeted messages is removed, the method also performs worse than the original version, which suggests that the retweeting behaviors can reflect the categories of both uses and messages.
- In our method, the constraint conditions of labeled messages and users lead to faster convergence. However, from the results, we also notice that they are helpful for a better performance by reducing noise in the dataset.

From above analysis, we demonstrate that our method is truly valid for both spammer and crowdturfing microblogs detection.

8 Sensitivity analysis

In real-world scenarios, spammers/spams usually take a small portion of the total number of users/blogs. For our collected data, labeled spammers make up only 6% of all users, and labeled spams only 0.08% of all messages, which coincides well with the reality. However, our previous experiments used a portion of the dataset in which the percentage of spammers reached up to 23% and that of spams was 26%. Although the new percentages of spams and spammers are still relatively small, we strongly believe that it is very important to check whether our method still retain its outstanding performance when the proportion is reduced.

In this section, we further evaluate our method using different spammer percentages, spam percentages, and training percentages. We compared our method with previous baselines in terms of Precision, Recall, F1 score, AP value, and AUC value. From the total results reported, our method still remains the best in the detection of both spams and spammers, again confirming that our method has stability of performance. This is especially meaningful when datasets lack dense annotations.

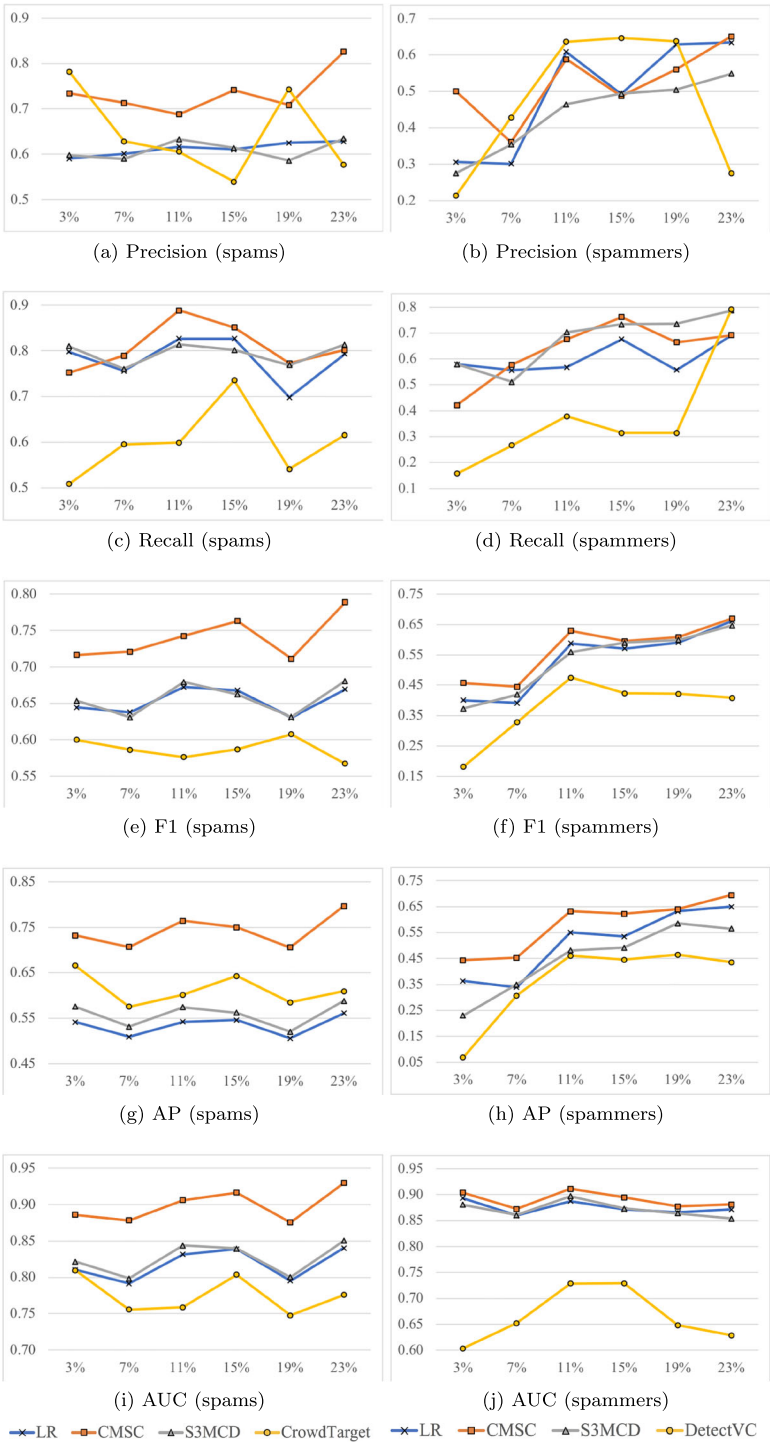


Figure 15 Detection performance w.r.t changing **SPAMMER** percentage

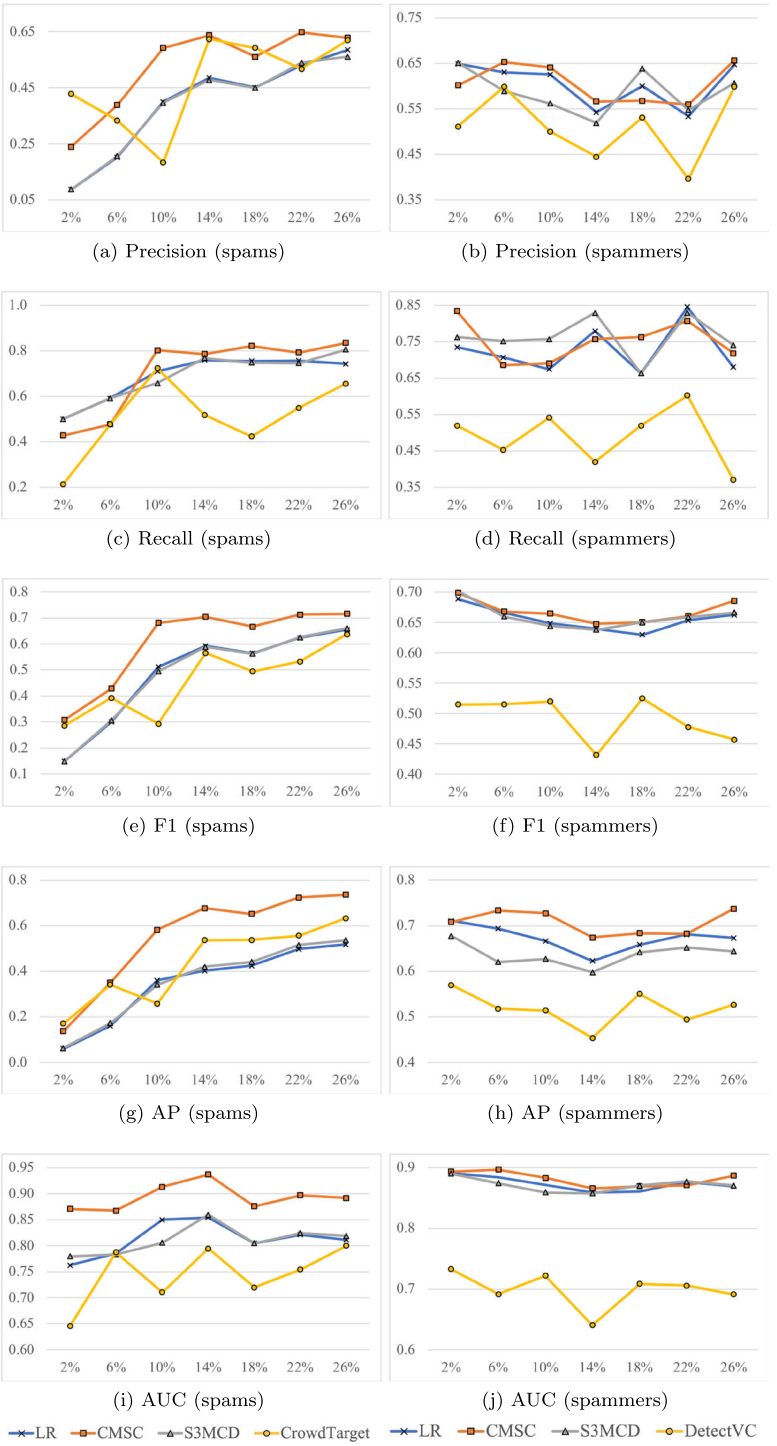


Figure 16 Detection performance w.r.t changing **SPAM** percentage

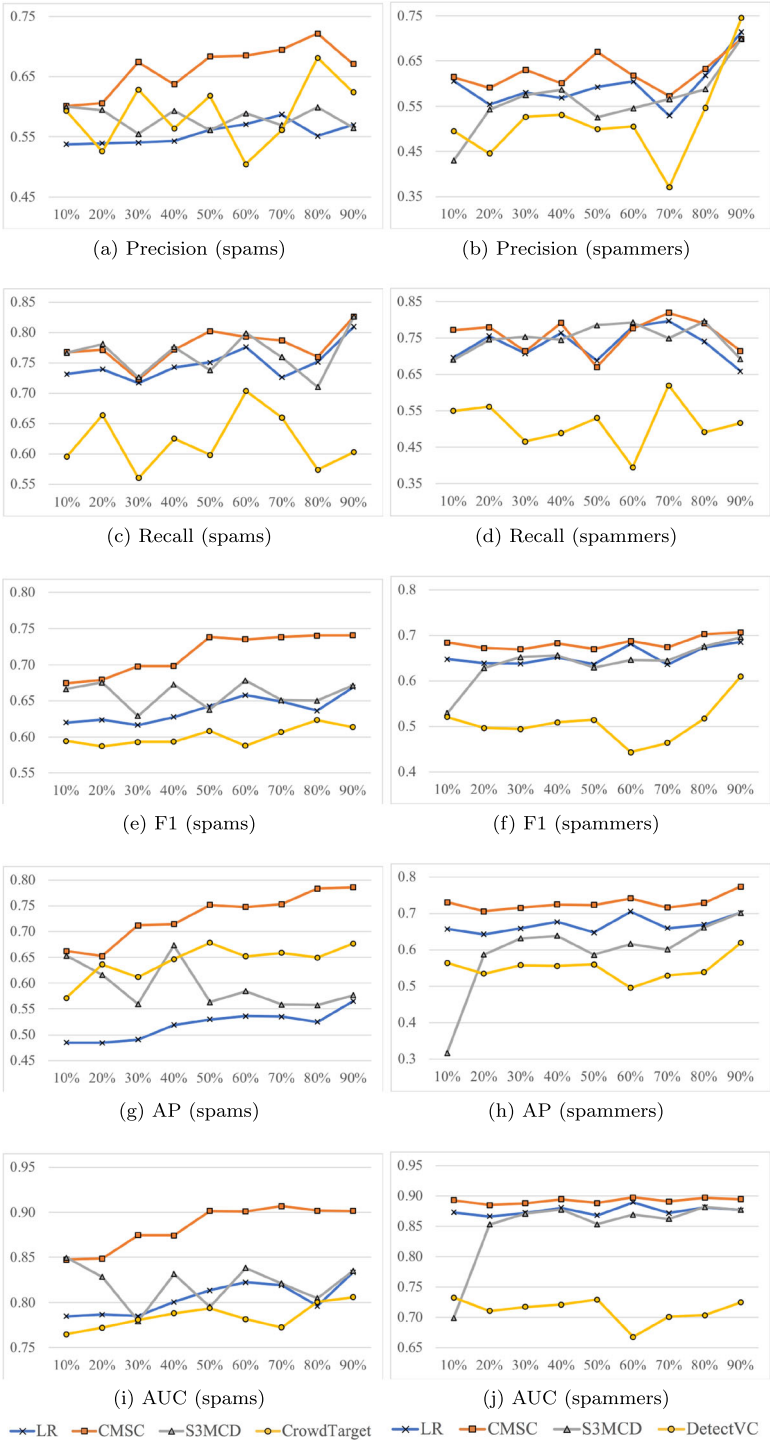


Figure 17 Detection performance w.r.t changing the percentage of **Labeled Data**

Different Spammer Percentages We adjusted our dataset, reducing the spammer percentage from 23% to only 3%. As depicted in the right column of Figure 15, our method performs relatively better than other spammer detection methods in which the spammer percentage ranges from 3% to 23%. The superiorities of the F1 and AP values are even more obvious when the percentage goes down. For the other metrics, our method still keeps its competitive performance. As may be seen in the left column of Figure 15, reducing the spammer percentage did not have much side effect on the final result of the spam detection. Thus our method also generally performed the best in spam detection.

Different Spam Percentages We reduced the spam percentage from 26% to only 2% by random selection. The detection performance can be seen in Figure 16, from which we can observe that our method still keeps its superior performance in various metrics regarding both spam and spammer detection. The results of spam detection is getting better along with the enlarging spam percentage and beat the competing methods. Changing the spam

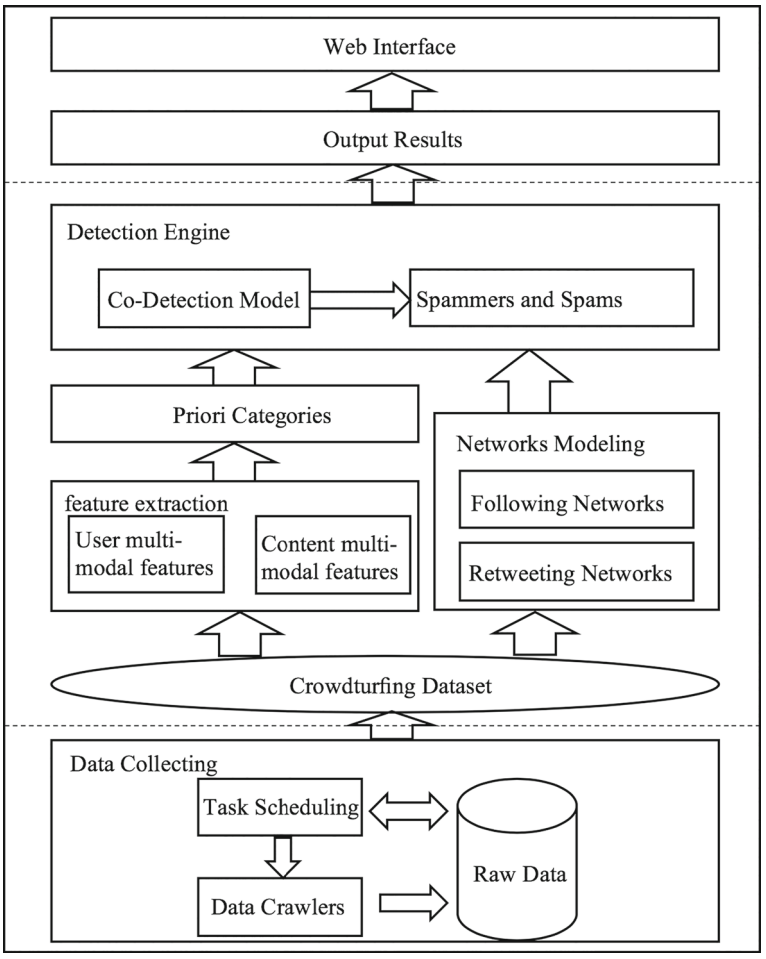


Figure 18 The framework of the prototype system

percentage also has a very limited impact on the spammer detection performance, which confirms again that our outstanding performance is very reliable in all of the above cases.

Different Percentage of Labeled Data In addition to changing the spammer or spam percentages, we here changed the proportion of labeled data. The proportion of labeled data was thus reduced from 90% to only 10%. The results are reported in Figure 17, from which we find that even in sparsely labeled cases, our method still beats the other methods. This is very meaningful because in practice, the cost of data annotation is often very high, so most datasets are sparsely labeled. Compared with traditional supervised methods, our semi-supervised method is better adapted to online social applications.

9 Prototype system

Finally, we develop a prototype system with our integrated work. The framework of our system can be seen in Figure 18 and includes three function layers: the data collection layer, the spammers and spams detection layer, and the interface layer.

In the data collection layer, we integrate our previous data collection engine, shown in Section 3, and construct a unified interface for Sandaha and Sina platforms (Figure 19). There are three primary modules in the first layer: a) a task scheduling module, which is designed for the distributed collecting task; b) a data crawler module, which is the core module for data collection; and c) a raw data module, which comprises our complex databases, including MySql, Mongodb, and Redis. The details of this layer has already been elaborated in Section 3.2.

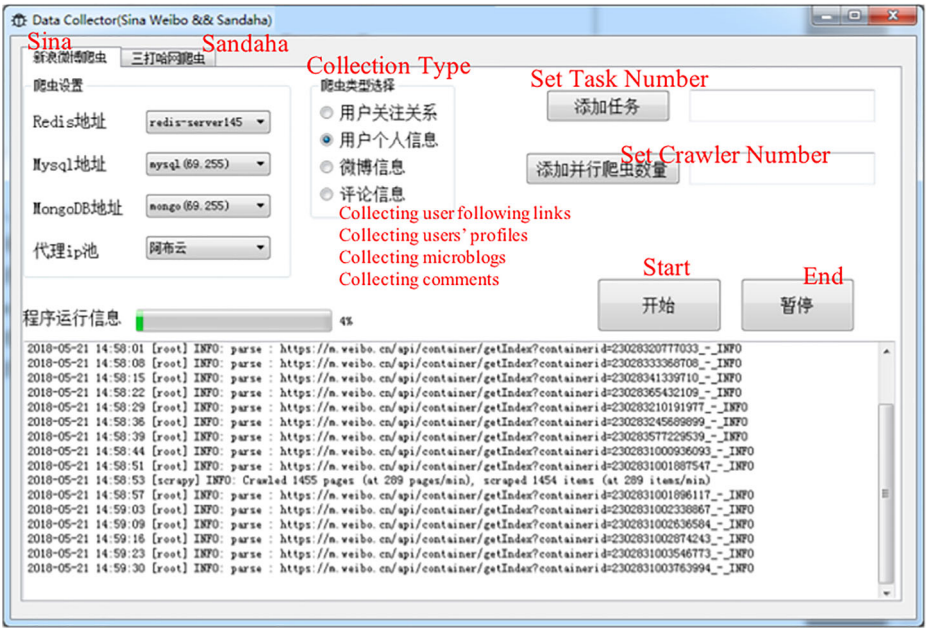
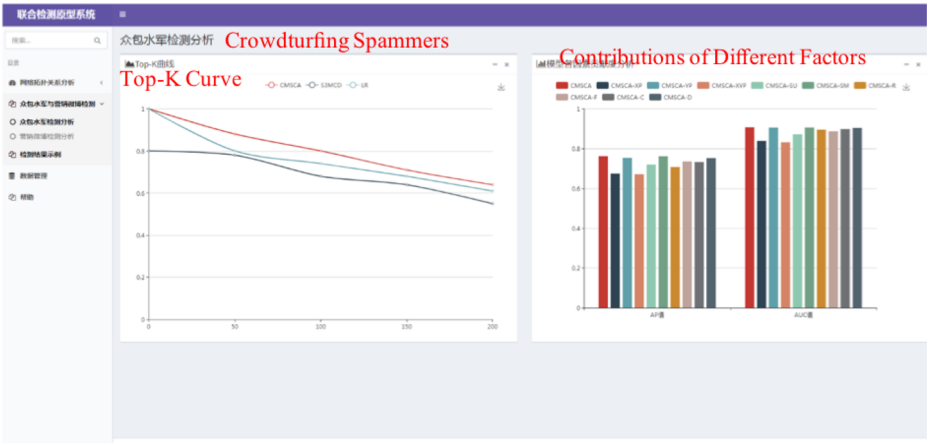
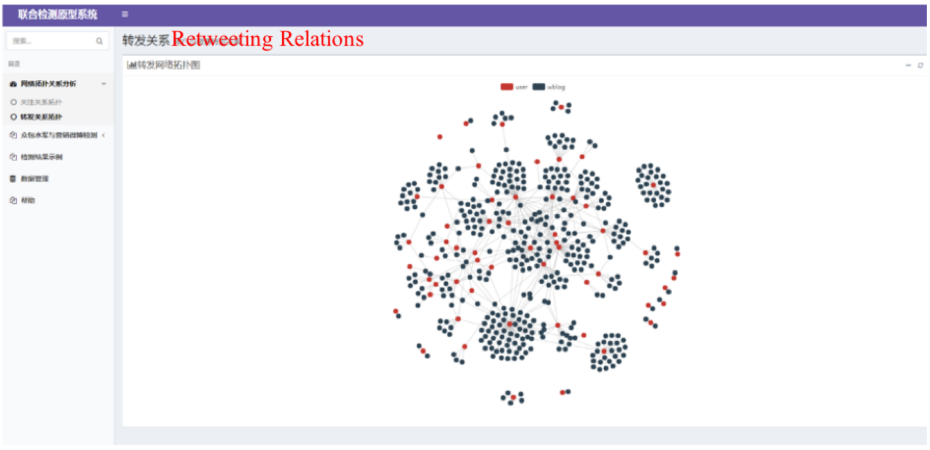


Figure 19 Data collection interface

(a) Crowdturfing Spams Results



(b) Crowdturfing Spammers Results



(c) Retweeting Topology Visualization

Figure 20 Snapshots of the prototype system

Based on these data, we introduce the core codes of our method and build the second function layer. The system first automatically extracts users' and contents' features, and meanwhile constructs both the following networks and the retweeting networks. The related results are then fed into our co-detection model and the system starts its learning and prediction process.

The final results, as well as other interesting visualization charts, are generated and sent into the third layer, which is developed by html/css/javascript and partly depicted in Figure 20a, b, and c.

The prototype system has good scalability. For example, the data collection module is designed as a distributed architecture. Users can temporarily add new crawlers through Figure 19 without breaking up on-going processes. We develop and preserve a number of APIs based on the data collected, such as users' following topologies, their retweeting networks, and so on. These results are organized in json format and can be used not only in our research, but also in related studies on community discovery, maximization of influence, and information diffusion. Our prototype system is available by request through the first author of this paper.

10 Conclusion

This paper propose a semi-supervised method to co-detect spammers and crwodturfing microblogs in the Sina platform. We combine users individual features, messages content features and the properties of following networks and retweeting networks. We also develop a robust system to collect data and analyze them. We extensively evaluate our method with related famous baselines, and the results report that our method performs much better both in spammers detection and microblogs detection. In the end, we develop a robust prototype system to further test the practicability of our method.

Acknowledgments This work is supported by National Key R&D Program of China 2017YFB1003000, National Natural Science Foundation of China under Grants No. 61972087, No. 61772133, No. 61472081, No. 61402104. Jiangsu Provincial Key Project BE2018706. Key Laboratory of Computer Network Technology of Jiangsu Province. Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No. BM2003201, and Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grants No. 93K-9.

References

1. Benevenuto, F., Magno, G., Rodrigues, T., Almeida, V.: Detecting spammers on twitter. In: Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference (CEAS), Vol. 6, p. 12 (2010)
2. Brown, G., Howe, T., Ihbe, M., Prakash, A., Borders, K.: Social networks and context-aware spam. In: Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work, pp. 403–412. ACM (2008)
3. Chen, T., Li, X., Yin, H., Zhang, J.: Call attention to rumors: deep attention based recurrent neural networks for early rumor detection. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 40–52. Springer (2018)
4. Chung, F.: Laplacians and the cheeger inequality for directed graphs. Ann. Comb. **9**(1), 1–19 (2005)
5. Ding, C.H., Li, T., Jordan, M.I.: Convex and semi-nonnegative matrix factorizations. IEEE Trans. Pattern Anal. Mach. Intell. **32**(1), 45–55 (2010)
6. Ghosh, S., Viswanath, B., Kooti, F., Sharma, N.K., Korlam, G., Benevenuto, F., Ganguly, N., Gummadi, K.P.: Understanding and combating link farming in the twitter social network. In: Proceedings of the 21st International Conference on World Wide Web, pp. 61–70. ACM (2012)

7. Hu, X., Tang, J., Gao, H., Liu, H.: Social spammer detection with sentiment information. In: 2014 IEEE International Conference on Data Mining (ICDM), pp. 180–189. IEEE (2014)
8. Jiang, M., Cui, P., Beutel, A., Faloutsos, C., Yang, S.: Catching synchronized behaviors in large networks: a graph mining approach. ACM Trans. Knowl. Discov. Data (TKDD) **10**(4), 35 (2016)
9. Kim, H.J., Chae, D.K., Kim, S.W., Lee, J.: Analyzing crowdsourced promotion effects in online social networks. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp. 820–823. ACM (2016)
10. Lee, K., Webb, S., Ge, H.: The dark side of micro-task marketplaces: characterizing fiverr and automatically detecting crowdturfing. In: ICWSM (2014)
11. Li, H., Chen, Z., Mukherjee, A., Liu, B., Shao, J.: Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In: ICWSM, pp. 634–637 (2015)
12. Liu, B., Luo, J., Cao, J., Ni, X., Liu, B., Fu, X.: On crowd-retweeting spamming campaign in social networks. In: 2016 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2016)
13. Liu, B., Ni, Z., Luo, J., Cao, J., Ni, X., Liu, B., Fu, X.: Analysis of and defense against crowd-retweeting based spam in social networks. World Wide Web, pp. 1–23 (2018)
14. Liu, L., Jia, K.: Detecting spam in chinese microblogs-a study on sina weibo. In: 2012 Eighth International Conference on Computational Intelligence and Security (CIS), pp. 578–581. IEEE (2012)
15. Liu, Y., Liu, Y., Zhang, M., Ma, S.: Pay Me and I'll follow you: detection of crowdturfing following activities in microblog environment. In: IJCAI, pp. 3789–3796 (2016)
16. Page, L., Brin, S., Motwani, R., Winograd, T.: The Pagerank Citation Ranking: Bringing Order to the Web. Tech. rep., Stanford InfoLab (1999)
17. Shu, K., Wang, S., Le, T., Lee, D., Liu, H.: Deep headline generation for clickbait detection. In: ICDM, pp. 467–476. IEEE Computer Society (2018)
18. Song, J., Lee, S., Kim, J.: Crowdtarget: Target-based detection of crowdturfing in online social networks. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 793–804. ACM (2015)
19. Stringhini, G., Kruegel, C., Vigna, G.: Detecting spammers on social networks. In: Proceedings of the 26th Annual Computer Security Applications Conference, pp. 1–9. ACM (2010)
20. Tam, N.T., Weidlich, M., Zheng, B., Yin, H., Nguyen, Q.V.H., Stantic, B.: From anomaly detection to rumour detection using data streams of social platforms. In: Proceedings of the Forty-fifth International Conference on Very Large Data Bases (VLDB'19). CEUR-WS.org (2019)
21. Thanh Tam, N., Matthias, W., Hongzhi, Y., Bolong, Z., Quoc Viet, H.N., Bela, S.: User guidance for efficient fact checking. In: Proceedings of the Forty-fifth International Conference on Very Large Data Bases (VLDB'19). CEUR-WS.org (2019)
22. Wang, G., Wilson, C., Zhao, X., Zhu, Y., Mohanlal, M., Zheng, H., Zhao, B.Y.: Serf and turf: crowdturfing for fun and profit. In: Proceedings of the 21st International Conference on World Wide Web, pp. 679–688. ACM (2012)
23. Wang, T., Wang, G., Li, X., Zheng, H., Zhao, B.Y.: Characterizing and Detecting Malicious Crowdsourcing. In: ACM SIGCOMM Computer Communication Review, Vol. 43, pp. 537–538. ACM (2013)
24. Wu, F., Shu, J., Huang, Y., Yuan, Z.: Social spammer and spam message co-detection in microblogging with social context regularization. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 1601–1610. ACM (2015)
25. Yang, X., Yang, Q., Wilson, C.: Penny for Your Thoughts: Searching for the 50 Cent Party on Sina Weibo. In: ICWSM, pp. 694–697 (2015)
26. Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B.Y., Dai, Y.: Uncovering social network sybils in the wild. ACM Trans. Knowl. Discov. Data (TKDD) **8**(1), 2 (2014)
27. Yuan, D., Li, G., Li, Q., Zheng, Y.: Sybil defense in crowdsourcing platforms. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1529–1538. ACM (2017)
28. Zhu, Y., Wang, X., Zhong, E., Liu, N.N., Li, H., Yang, Q.: Discovering spammers in social networks. In: AAAI (2012)

Affiliations

Bo Liu[1] · Xiangguo Sun[1] · Zeyang Ni[1] · Jiuxin Cao[2] · Junzhou Luo[1] · Benyuan Liu[4] · Xinwen Fu[3,4]

Xiangguo Sun
sunxiangguo@seu.edu.cn

Zeyang Ni
nizy@seu.edu.cn

Jiuxin Cao
jx.cao@seu.edu.cn

Junzhou Luo
jluo@seu.edu.cn

Benyuan Liu
bliu@cs.uml.edu

Xinwen Fu
xinwenfu@cs.ucf.edu; xinwenfu@cs.uml.edu

1 School of Computer Science and Engineering, Southeast Univerisity, NanJing, 211189, China

2 School of Cyber Science and Engineering, Southeast Univerisity, NanJing, 211189, China

3 Department of Computer Science, University of Central Florida, Orlando, FL, USA

4 Department of Computer Science, University of Massachusetts Lowell, Lowell, MA, USA