ATLANTIS PRESS
Advances in Computer Science Research (ACRS), volume 54
International Conference on Computer Networks and Communication Technology (CNCT2016)

A New Approach to Detect User Collusion Behavior in Online QA System

Zhen-hui ZHU[a],Zhi YANG[b] and Ya-fei DAI[c]
Institute of Network Computing and Information Systems, Peking University,
Beijing, China
[a]zhenhui_zhu@pku.edu.cn, [b]yangzhi@net.pku.edu.cn, [c]dyf@net.pku.edu.cn

Keywords: Collusion, Sybil Detection, Interaction Weight, Clustering Coefficient.

Abstract. Sybils and Sybil attacks are problems born with social networks. In online QA application Afanti, Sybil users collude with each other to mimic normal users and lower the possibility to be detected. In this paper, we stated this phenomenon and put forward a new approach that can detect user collusion behavior which cannot be detected before. In this approach, we defined interaction weight between users to describe the collusion, clustered users by this weight and labeled users as Sybil or normal through clustering coefficient. We also proposed a plan for deploying this approach in large-scale system and pointed out the key part and how to improve it. The experiment shows the accuracy of our approach is 93.5% in detecting Sybil questioners and 97.4% in Sybil answerers. Our approach can also recall many Sybils which cannot be detected by the original detection system.

Introduction

Sybils and Sybil attacks are big problems born with social networks. In a Sybil attack, a user creates multiple fake identities, known as Sybils, to unfairly increase their power and influence within a target community [1], sometimes to gain money profits. Many studies have been done on these topics.

Fake opinion, fake user activity and fake identity are three main topics these studies focus. For fake opinion, content-based, graph-based and behavior-based [2, 3, 4] approaches are used in studies. Recent studies also show a trend in combing these methods [5]. For fake activity, behavior pattern, graph structure and domain information [6, 7, 8] are good ways to detect these activities. Fake identity is the most important part in Sybil detection, since finding these identities help we reduce fake opinion and fake activities. Most studies on this topic are done through network-based features [1, 9] or behavior patterns [10, 11].

Despite the fact there are lots of research on all kinds of social networks, as far as we know, little has been done on the social network of online QA system.

Online QA systems and websites are gaining their popularity day by day. Users in these applications or websites have different activities compared to traditional social networks. For example, Afanti is an online QA education application, which provides platform for students to ask questions and for teachers to answer (students can answer questions and teachers can ask questions if they want). Many teachers can answer the same question, but only one answer can be chosen as the best by the student. Teachers whose answer to a question has been chosen as the best can get reward points, which can be converted to money. These incentives have led to a lot of cheating behaviors.

Fig. 1 shows Afanti's design. In this application, questions put forward by students come into question pool, which have hundreds of thousands new questions a single day, and teachers choose questions from the pool to answer. We can theoretically think

teachers choose questions randomly due to the large volume. However, Sybil users do not behave so randomly. The interactions between Sybils appear cooperation.

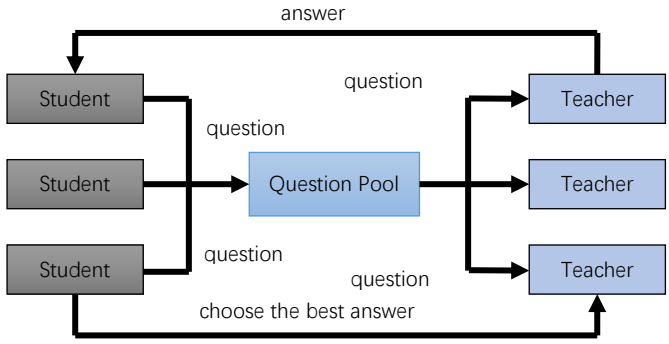
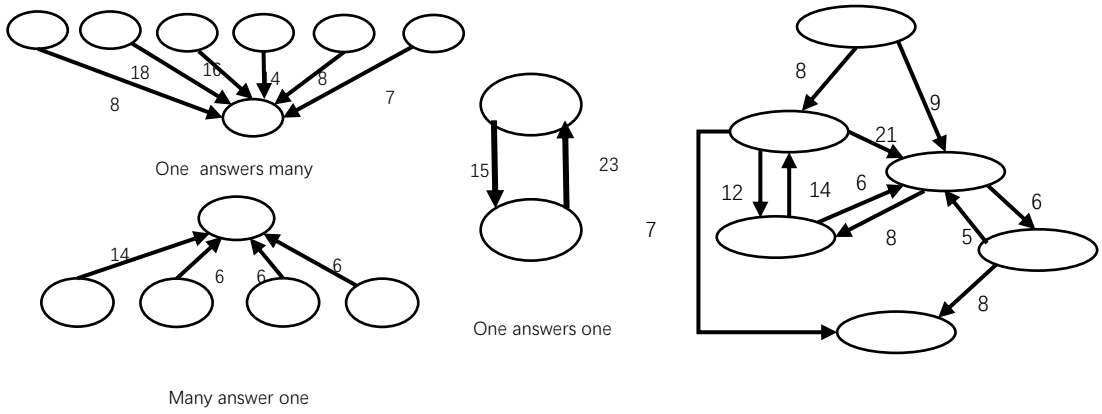


Figure1. Afanti's design

This application has its original detection method focusing on some cheating patterns. (1) If the interaction question number between a student and a teacher is higher than some threshold and occupy a big enough irregular proportion, for example 96 out of 100 questions are answered by one person, the system will detect this and label them as Sybil. (2) Based on (1), if many teachers almost only answer one specific student's questions or one teacher answers almost all several students' questions (all need to be higher than a threshold and occupy a big proportion), the system label all of them Sybil. The present detection method is able to detect the traditional one-answers-one, one-answers-many, many-answers-one cheating patterns in Fig. 2(a). It has good accuracy, but soon many Sybil users realize this and change their cheating patterns to collusion. That is, several Sybil users cooperate together to mimic normal users and to reduce the possibility to be detected. In Fig. 2(b), several teachers answer several students' questions, so every teacher or student's interaction proportion is not high. This makes them unable to be detected by the original method.



(a)Existing detected patterns(b)Collusion pattern

Figure2. Collusion patterns (Arrow starts from questioners to answerers.)

In this paper, we focus on these many-to-many collusion behaviors and propose a new approach to detect Sybil users.

Proposed Approach

Sybil users cooperate to earn profits. No matter how hard they try to mimic normal users, their collusion behavior always have implicit connection there. Collusion also means one Sybil user need to collude with other Sybil users. Based on this, catching one Sybil user, the users he interacts are with a higher possibility to be Sybil.

Seeing that and try to maximize the use of the application's original Sybil detection method, we propose the following approach to catch the collusion behavior.

Getting Dataset

We crawled Afanti application's log data under their permission. The data is from October 2015 to March 2016. We first get 1200 Sybil users and 1740 normal users from original detection method as two kinds of seeds. These seeds were carefully chosen to make sure some of them ask questions and some answer them. Then we crawled all interaction users with these seeds and users they interact with, that is, all one-hop and two-hop neighbors of the original seed set. All users in this application have their unique ids, and the crawled log's format is (questioner id, timestamp, question id, best answer's answerer id) tuple. Some basic information about this dataset is in Table 1.

In experiments, we remove all users whose total actions, or called action degree, (an action is either putting forward a question or answering one) are less than 10, for these users may just come to this application and do not have enough behavior patterns.

Table 1. Basic information about dataset

Dataset	Seed num	Total users in a group	Total questioner	Total answerer	Users both ask and answer	Total edges in a group
Sybil Group	1200	182445	165713	20545	3813	942779
Normal Group	1740	135383	103588	37270	5475	315914

Defining Interaction Weight

Since questioners' proportion in the data group is higher, our approach starts from the questioner side. To capture the collusion behavior and inspired by [7], we define interaction weight between two questioners as Eq. 1.

$$w_{ij} = \frac{|A_i \cap A_j|}{|A_i \cup A_j|} \tag{1}$$

Where A_i means the user set who answered user i's questions. The same answerer can appear several times in the set if his answers to user i's questions have been chosen as the best several times. The interaction weight w_{ij} between user i and j is answerer set's Jaccard similarity coefficient. If teachers choose questions randomly in the pool, the chance one teacher answers random two users' questions should be very small. While Sybil users' weight will be larger because of their collusion.

Clustering and Classification

After calculating the weight, we can measure the collusion between any two users. Now a collusion graph can be drawn. In the graph, a node denotes a single user, while an edge denotes the interaction. The larger the interaction weight, the thicker the edge. Thick edges may indicate collusion, while thin edges may be caused by randomness.

For clustering, we choose several edge weight threshold, then remove all edges whose weight are lower than the threshold. The left graph forms several clusters. For each cluster, to measure the inside collusion, the global clustering coefficient and cluster size is computed in Table 2. We then use it to classify each cluster as normal or Sybil. The process is in Fig. 3.

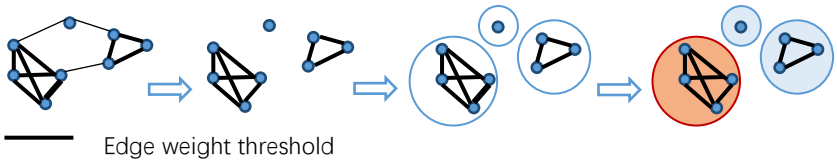


Figure3. Generate clusters and classify them

Sybil teachers are computed only when we detect Sybil questioner cluster.

Experiments

In experiments, we use 6 different edge weight thresholds:0.02, 0.05, 0.1, 0.15, 0.2, 0.3. When the threshold is too small (0.02, 0.05), almost all nodes, normal and Sybil, are connected, there are no cluster patterns. For thresholds bigger than 0.1, the final result doesn't improve much with the increasing of threshold. So we only list three edge thresholds 0.1, 0.15 and 0.2's results in Table 2.

Table 2. Experiment result

Dataset	Weight threshold	Total nodes left	Node num in the cluster	Cluster proportion	Global clustering coefficient	Average edge count
Sybil Group	0.1	11592	10799	93.2%	0.803	905
	0.15		10715	92.4%	0.903	700
	0.2		10655	91.9%	0.932	577
Normal Group	0.1	2133	130, 560	6.1%,26.3%	0.733,0.545	7,134
	0.15		552	25.9%	0.647	89
	0.2		551	25.8%	0.579	55

Questioner Side

We start from the questioner's part. As Table 2 indicates, after removing low action degree nodes, 11592 out of 165713 questioners in Sybil group, 2133 out of 103588 questioners in normal group are left. The phenomenon low degree users occupy a big proportion shows that Afanti is in its expanding period and there are many new users.

We observe the clustering coefficients of these two groups are different and can distinguish them easily. With the increment of weight threshold, the clustering coefficient for Sybil group becomes higher and is always higher than the one for normal group, while clustering coefficient for normal group decreases.

Let's focus on the result of threshold 0.15, in collusion part, 92.4% nodes are still connected while only 25.9% nodes in normal part connected. The two clusters' clustering coefficient are different with a gap of 0.256. This different left proportion and different clustering coefficient may be explained by the average edge count of two groups. In the Sybil group, the average edge count is 7.9 times of normal group's value.

To check the clusters' nodes are normal or Sybil, we randomly choose about 5% of these clusters and give them to Afanti to do a hand verification. We choose clusters with 0.1 and 0.15 threshold in normal group, and one cluster with 0.15 threshold in Sybil group since the other clusters are similar to these ones.

The verification checks users' log, question or answer's content, timestamps and so on. The result shows, all 2+28+27=58 chosen users in normal group clusters are normal, and 500 out of chosen 535 users in Sybil group cluster are Sybil. Taken the whole cluster with high (low) clustering coefficient as Sybil (normal), the accuracy of our method for detecting Sybils is about 93.5%. This shows a weight threshold no less than 0.1 is suitable for finding the clusters with high accuracy. And comparing the Sybil

proportion in these two groups, the hypothesis that users interact with Sybils (normal users) are with a higher possibility to be Sybil (normal) is valid.

Back to the 0.15 threshold experiment, for these 10715 detected Sybil users, 5258 of them cannot be detected by Afanti's original detection method, which means adding our approach to the detection system can increase the recall with a big improvement.

In a nutshell, the clusters generated by our detection method from the users interact with seed users can be labeled as normal or Sybil by its clustering coefficient. For dataset of Afanti, an interaction weight threshold as 0.1 or 0.15, and separation threshold between 0.75-0.8 is enough to generate these clusters and label them. With just some seed Sybil users, this method is able to detect many Sybils who have collusion behaviors which cannot be detected before.

Answerer Side

This part is conducted only when we find Sybil questioner clusters. We use the 10715 detected Sybils in this experiment, and define Sybil answerer's collusion to be that one user answers at least 4 users and at least 2 times each user in the Sybil cluster. This detects 199 answerers Sybil and they interact with 99.7% of the 10715 detected questioners, and about 10% cannot be detected before. We also conduct a hand verification and 194 of them are Sybil, the accuracy is 97.4%.

Discussions

While our approach has high accuracy in detecting Sybil users, we find some parts of the experiment that worth discussing.

First, the seed set. When we choose the seed set, we carefully choose the seeds so that some of them ask questions and some of them answer. We think that choosing different seed groups may affect the final result. Of course, in real system, we can directly use all the interaction data to do the classification, that may consume more resource, but reduce the worry about seed set selection.

Second, our method pays more attention to questioner side. This improves the using experience in this application, for less people can put forward low quality questions and help Sybil teachers earn money now. But for directly reducing money paid to Sybils, methods focusing on detecting Sybil teachers will be more useful.

Deployment in Large-Scale System

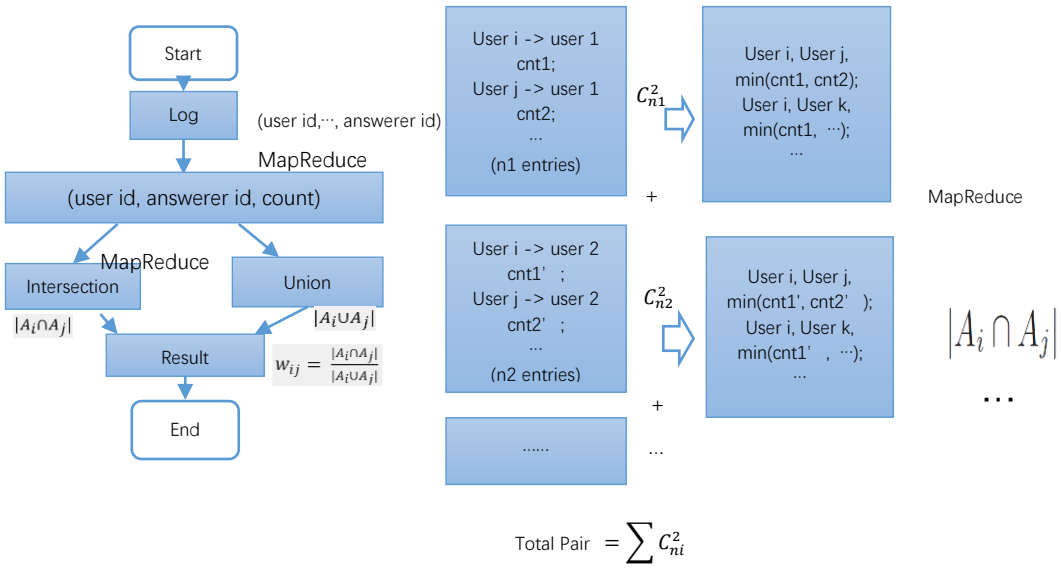
Our experiment shows the collusion detection method is effective in the small dataset. Considering Afanti is an application which had over 20,000,000 registered users and 500, 000 new questions a day by the end of January 2016 [12], deploying a real detection method in large-scale system is needed. We propose the detection method in MapReduce since Afanti use it in its backend. Fig. 4(a) shows the flow.

First, we calculate (user id, answerer id, count) tuple from log to record questioners, their best answer's answerer id and the interaction count between them. If user j's two answers to user i's two questions have been chosen as the best, the count is 2. Second, we compute Eq. 1's numerator and denominator, the intersection and union set size of user i and j's answerer set. The same answerer can appear several times in the set if the count is bigger than 1. Third, a division is done to generate the interaction weight.

After getting this weight, simple BFS method is efficient to find the clusters. To calculate clustering coefficient, existing library or writing programs as needed are both feasible because the system can randomly choose several nodes and calculate their global clustering coefficient for simplicity.

During our deployment, we find the key part is computing the intersection and union set of answerers. Fig. 4(b) shows the detail. When computing the intersection set, after we get the (user id, answerer id, count) tuple, we fix the answerer id and compute every pair of users in this group to get the intersection number. Then MapReduce is used again to get the total intersection answerer set size.

For Afanti with a DAU in $10^5 - 10^6$range, the total pair number will be around $10^{10}$for a single day, which consumes too much time and computing resource. We can filter some data for simplicity. For example, if we ignore all tuples whose count is 1 (and this is workable for in experiments we find most Sybils ask more than 1 question to other Sybils), then the computing pairs will be reduced to 10^8 level.



(a) Flow of the approach(b)Intersection set computation detail

Figure4. Details of the approach deployed on MapReduce

Conclusions

This paper presents an approach to detect Sybil users' collusion in online QA system. The method has three stages. First, calculating every two users' interaction weight to form a weighted undirected graph. Second, all edges whose weight is smaller than the threshold are removed, leaving several clusters in the graph. Third, separating normal clusters from Sybil ones through clustering coefficient. Sybil answerers can be detected easily after we get the Sybil questioners. Lastly, we propose a feasible plan for deploying this method in large-scale system and point out the key part and how to improve it. The experiment shows the accuracy of our approach is 93.5% in detecting Sybil questioners and 97.4% in Sybil answerers. Our approach can also recall many Sybils the original system cannot detect before.

References

[1] Z. Yang, C. Wilson, X. Wang, T. Gao, B.Y. Zhao and Y. Dai. Uncovering social network Sybils in the wild. In IMC, 2011.

[2] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In WWW, 2006.

[3] G. Wang, S. Xie, B. Liu, and P.S. Yu. Review graph based online store review spammer detection. In ICDM, 2011.

[4] N. Jindal, B. Liu, and E.P. Lim. Finding unusual review patterns using unexpected rules. In CIKM, 2010.

[5] S. Rayana, L. Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In SIGKDD, 2015.

[6] A. Beutel, W. Xu, V. Guruswami, C. Palow and C.Faloutsos. CopyCatch: Stopping group attacks by spotting lockstep behavior in social networks. In WWW, 2013.

[7] G. Wang, T. Konolige, C. Wilson, and X. Wang. You are how you click: Clickstream analysis for Sybil detection. In USENIX Security, 2013.

[8] Y. Li, O. Martinez, X. Chen, Y. Li and J.E. Hopcroft. In a world that counts: Clustering and detecting fake social engagement at scale. In WWW, 2016.

[9] Z. Cai, C. Jermaine. The latent community model for detecting Sybils in social networks. In NDSS, 2012.

[10] Q. Cao, X. Yang, J. Yu, and C. Palow. Uncovering large groups of active malicious accounts in online social networks. In CCS, 2014.

[11] J. Xue, Z. Yang, X. Yang, X. Wang, L. Chen and Y. Dai. VoteTrust: Leveraging friend invitation graph to defend against social network Sybils. In INFOCOM, 2013.

[12] Information onhttp://www.cyzone.cn/a/20160126/289242.html