Troll Vulnerability

Christos Spatharis
MSc Student
University of Ioannina
Dept. Computer Science & Engineering
cspatharis@cse.uoi.gr

Abstract

Online social networks are great source of information and are being used by millions of people worldwide. The behavior of the users in the online community defines the reliability of the network. Like all social networks, Slashdot has many malicious users who are harming the community with their harmful comments. In this paper, we present graph-based and machine learning approaches in order to detect which users are going to be trolled as well as which posts are more vulnerable to such attacks. For the first task, we use the signed graph of the Slashdot network to find the benign users that are vulnerable to troll attacks, while for the second task we acquire the posts and perform classification techniques to identify the vulnerable ones.

1 Introduction

Users in online communities contribute with posts, comments, votes and reports. These are essential for the integrity and trustworthiness of the sites. As a result, the community moderators force some rules in order to prevent unwanted behaviors. Users who are not abide by the rules, are subsequently banned from the community.

In every community, users develop relationships that can be characterized as positive or negative. Positive links between them show agreement, friendship in the real or online world or approval. On the other hand, negative links show disagreement, disapproval or conflict between the users. The role of Social Signed Networks (SSN) is to create a graph with those positive or negative edges between the users. In Slashdot, a user can mark another user as friend or foe. Malicious users often show approval towards other malicious users and it is common to form alliances before attacking a user or a post.

Our first goal is to use the signed graph of the Slashdot network in order to separate benign from malicious users. The separation is a result of decluttering operations as shown in [1]. Subsequently, we use these sets of users to determine which of the benign user is more vulnerable to troll attacks. As the users grow in community, they develop relationships, many of which are with malicious users. As a result, we think that malicious users tend to be more aggressive on users who have previously marked them as foes. With this in mind, we measure how close a benign user is to his foes. Our implementation differs from the other techniques as it seeks for benign users who are more vulnerable to troll attacks, while the others trying to determine which users are malicious or going to be malicious in the future.

Our second goal, is to mine comments from Slashdot posts and train a classifier to determine whether a new post will be trolled or not. In Slashdot, users can up-vote or down-vote a comment, but the moderators of the community are the only ones who have the power to delete a comment. Also, moderators give a score to each post under an article. Posts with negative scores are considered trolls as the may contain inappropriate content. This problem was challenging, due to the fact that we had to acquire the data-set ourselves.

2 Datasets

We used two different datasets coming from the same online community, Slashdot. The first one, is the signed network of Slashdot users and the second one contains the comments obtained through

Metric	Benign	Malicious
In-Degree	53765	11455
Freak	36276	28944
Mod. Page-Rank	56190	22926

Table 1: First iteration.

Metric	Benign	Malicious
In-Degree	28945	10774
Freak	42673	6857
Mod. Page-Rank	42591	11008

Table 2: Final Decluttered Graph.

scrapping.

Specifically, Slashdot signed graph consisted of 79.120 vertices and 515.397 edges between them. The amount of negative links is 23.9% ,while the reciprocity is 18.5%.

A signed social network, is the one in which a user *u* can have either a positive or negative relationship with another user *v*. Formally, it is a directed, weighted graph G = (V, E, W), where V are the users of the network, $E \subseteq VxV$ are the edges between them and W: E $\rightarrow$ [-1, +1] the weights. Positive edges (+1) show friendship between users, while negative edges (-1) show disapproval.

Our second data-set consists of more than 3000 articles along with the features we extracted that are related to them. The data collected, contains articles from September 2016 till January 2017. Although, we could have collected more, it is a time-consuming process that requires a lot of pre-processing. This data-set contains the text of the article, the number of comments, the number of words, the tags and the author.

3 User Vulnerability

In this chapter, we will discuss the techniques used to separate the the users of a community into benign and malicious and after that we will present our measure for determining whether a user is vulnerable to troll attacks or not.

3.1 Centrality Measures for Signed Social Networks

Here, we will introduce some of the well-known metrics for measuring the centrality of the nodes of an SSN. Given an SSN G = (V, E, W), a node centrality measure is a function $R : V \rightarrow R$ that assigns a score to each user. In [1] the authors use 8 different metrics to extract the centrality of each node. but we make use to three of them in order to acquire the scores. The metrics used, described here:

Freaks
To obtain the centrality, this metric only considers the negative incoming edges of each node.

$$centrality(u) = \sum_{u \epsilon V | W(u,v) < 0} W(u, v)$$

This definition, tells us that if a node has many negative incoming edges, then he has a reputation for being a bad user.

In-Degree
As the name suggests, this metric computes centrality after summing over all of the incoming edges of a node. This metric favors the popular users who also have many enemies.

$$centrality(u) = \sum_{u \epsilon V | (u,v) \epsilon E} W(u, v)$$

The difference with the previous metric, is that the previous one can falsely identify a popular user as malicious because he has many negative edges.

Modified Page-Rank
As we already know, Page-Rank is used in directed graphs with positive edges. As a result, a signed graph is not suited for the algorithm. So we need to modify it, as proposed in [3] to take account both positive and negative links. The approach consists of separating the negative and positive edges into to two different graphs and computing the Page-Rank score for the nodes on each one. Finally, the centrality score is calculated as the difference between the Page-Rank scores of the positive edges graph(PR+) and the scores of the negative edges graph (PR-).

$$centrality(u) = (PR+) - (PR-)$$

3.2 Decluttering Operations

A decluttering operation is a function $f : G \rightarrow G$ that transforms a graph by removing some of its

edges. For all G = (V, E, W), if f(G) = G' = (V', E', W'), then V = V', $E \subseteq E$ and for all $e' \epsilon E'$, W'(e') = W(e').

As shown in [1], the separation of benign from malicious users can be implemented after breaking down the original signed graph. In our approach, we use some of the decluttering operations in order to isolate our benign set of users. In order to proceed with the operations, we have to first define the centrality of the nodes. Furthermore, we have to set an initial threshold so as to determine if a user is benign or malicious depending on his popularity.

After the first pass, we have already classified our users into two sets depending only on their popularity and the centrality metric we used. As we mentioned earlier, these metrics are not safe for immediate inference on our classification task. Consequently, we need to remove some edges which are not important to the final graph. Firstly, we remove edges between malicious users that have positive weight. In that way we reduce even more the centrality of those users, as it is often observed that malicious users endorse other malicious users in order to create their army before attacking.

Moreover, we remove positive edges from benign to malicious users. Many times, malicious users add a benign user as a friend and the benign user endorse them back out of courtesy. Also, malicious users often hide their true identity and try to lure benign users in order to increase their popularity.

Thereafter, we re-compute the centrality of every node of the network and come up with the final graph. The task of finding the vulnerable users only using the signed graph is very challenging as Slashdot did not provide any ground truth about users who are more vulnerable to troll attacks than others. So, our approach to determine the vulnerability of a user lies in finding out how "close" the benign user is to his enemies. In order to measure these distances, we can compute the number of negative incoming edges to our benign set of nodes. This can lead us to the number of enemies each benign user has.

Even though our assumption is based on online world phenomena where disapproval between users can lead to many conflicts, we cannot do any better just by examining the signed graph of the network. In the following section, we introduce machine learning to help us classify vulnerable posts.

4 Post Vulnerability

In this section, we use a different approach to detect vulnerable posts. We try to predict if a post is going to be trolled using machine learning techniques. Hereafter, we will address both articles and comments bellow them as posts but in the end we will focus on classifying the original articles as vulnerable or not. The collection of the data-set was the most time-consuming part of the project as there was no official API for Slashdot. As a result, we implemented our own scrapper in order to acquire useful data from the HTML pages. Finally, we extracted meaningful features to help us on with our prediction task.

4.1 Scrapping

Even if there are many signed graph data-sets online about Slashdot users, there are none available for posts made by editors and users. Slashdot's main page consists of popular articles written by Senior Editors. These articles tend to lure the majority of community users and have the most comments. On the other hand, users can also post their own articles but their posts usually have no comments. As a consequence, we cannot use the history of posts a user has made to come up with interesting results about each user.

We implemented the scrapper using *Python* and the module *pyquery*. It works only on Slashdot HTML pages as the parts we searched in the code can only be found in their structure.

4.2 Features

Before we proceed with our classification task, we need to extract some features that are related to the article. As observed in [2], useful post features can be the number of comments under a post, number of words and readability metrics. We think that readability metrics do not play significant role in our research, as the articles are being made by editors who are trying to preserve the integrity of the online community. Also,

articles come along with some tags being given to them by the authors and are relative to the news presented in the article. Moreover, moderators of the online community assign a score for every user comment under an article.

Due to our need for more features, we included LIWC features. Linguistic Inquiry and Word Count (LIWC), was developed by James W. Pennebaker and is a computerized text analysis program that outputs the percentage of words in a given text that fall into one or more of over 80 linguistic (e.g., first-person singular pronouns, conjunctions), psychological (e.g., anger, achievement), and topical (e.g., leisure, money) categories *[source: Wikipedia]*. In order to obtain this kind of features, we use VADER Sentiment Analysis, a well-known python module. The sentiment analysis returns a positive, a negative and a neutral score based on the text given. It also returns a compound score which is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive). This is the most important score as the writers suggest. Finally, the features used in the training as well as in the testing process are being given in *Table 3*.

Features
LIWC Features
Number of Comments under the Article
Number of Words of the Article
Score for each Comment
Tags

Table 3: Classifier Features.

4.3 Prediction

As the ground truth was not given by any external source, we labeled each article based on the comments they received. Comments on Slashdot are thoroughly observed by the community moderators. As a result, moderators can up-vote or down-vote a comment. With that in mind, comments which negative voting score are considered as trolls. Hence, we say that an article has been attacked by trolls if it contains at least one comment with negative score.

As we mentioned earlier, we tried using all the features in *Table 3* but we ended up using those which gave us the best accuracy for our classifiers. The most dominant features were that from the LIWC analysis. The number of comments and the number of words are also critical to the success of the classifiers. All the features needed preprocessing before used in the classification task.

Classifier	1500 posts	3000 posts
Decision Tree	82%	82%
SVM	88%	86%

Table 4: Classifiers Accuracy based on two different length datasets.

We split our data-set to 70% training and 30% testing, and then we tried two classifiers over our sets measuring their accuracy. The data-set was repeatedly split into different train and test set and the average accuracy was considered. The first classifier was a Linear SVM Classifier, while the second was a Decision Tree Classifier. The best accuracy results came from the SVM classifier as can be seen in *Table 4*. Nonetheless, the results from the Decision Tree was also encouraging as the classifier's accuracy was always over 80%.

5 Results

Concerning the first task as we can observe in *Table 1, 2* we have separated the users into two groups. Also, it is obvious that some users of the original graph did not enter any category. That is due to the fact that after performing all these graph decluttering operations, some nodes of the graph lost all of their edges. The results we obtained after performing distance metrics between the benign and malicious nodes show us how close our benign node is to them. The discouraging part of this research is that we did not have any information available concerning the ground truth. So our result are based on the sociology of the network and the relationships between the users in it.

The results we obtained for our second task was far more encouraging than the first. Although no ground truth was given to us again, we managed to label the data. As presented earlier, the

Classifier	1500 posts	3000 posts
Decision Tree	75%	74%
SVM	88%	86%

Table 5: Classifiers Accuracy using only LIWC features.

results for our classifiers were very encouraging. SVM classifier ended up with mean accuracy of 88% while Decision Tree Classifier had mean accuracy of 82%. As we also observe in *Table 4*, nearly 1500 posts are already enough for our classifiers to reach their maximum accuracy. Still, *Table 5* shows how powerful LIWC features are to our research. Specifically, the SVM classifier is not influenced by the lack of the other features. On the other hand, Random Forest is a classifier that requires many features in order to achieve better accuracy. As for the precision and recall, we notice that they are higher for the vulnerable category. The drawback of our results was that there was no other implementation to test our results against. Our work presents a novel way to identify whether an article will be trolled.

6 Future Work

In this section we will present some ideas which we can use in the future in order to improve our results.

To begin with, concerning the user vulnerability we can try to link the graph with the comments of the users. Therefore, it will be more meaningful to determine whether a user who has many enemies is more likely to be trolled or the two approaches are not correlated.

Regarding post vulnerability, we could look out for more features to use. Features like up-votes or down-votes of the article or articles written by users can be used for our training process. Moreover, the tags that accompany the article could be encoded using One-Hot Encoding in order to make sense which tags produce the most inflammatory posts. Finally, we could try to train more classifiers rather than the two used in our research.

In our work, we only took into consideration the articles written by some editors of the community. As we mentioned before, users can also post their own articles but they are not useful as the comments bellow them are few. On the other hand, we want to find a way to extract the replies under a user comment found in an article. This is very hard to be done, due to the fact that the moderators hide the offensive replies marking them as trolls. Our goal is to go one step further and classify user comments under an article into vulnerable or not.

References

[1] S. Kumar, F. Spezzano, and V. Subrahmanian. 2014. *"Accurately detecting trolls in slashdot zoo via decluttering"*, in Advances in Social Networks Analysis and Mining (ASONAM).

[2] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec. 2015. *"Antisocial behavior in online discussion communities"*, in Proceedings of ICWSM.

[3] M. Shahriari and M. Jalili. 2014. *"Ranking nodes in signed social networks"*, in Social Netw. Analys. Mining.