Detecting Fraudulent users in Stackoverflow

Enjal Parajuli
Boise State University
Boise, Idaho
enjalparajuli@u.boisestate.edu

1 INTRODUCTION

Stackoverflow is an on-line community of users who are involved in questioning and answering on a wide range of topics in computer programming. The trustworthiness or importance of a person is mostly based on the reputation(i.e. points). Each member earns additional reputation for posting quality questions or answers or both. The reputation might get deducted sometimes. The deduction happens often when the other people in the community find that the question or answer posted by a user is non-relevant or does not meet the stackoverflow quality.

Since the reputation of each person is determined by other users in the community, this creates a loophole for fraudulent activity. If a person **X** hates user **Y**, all user **X** needs to do is to go to every question or answer posted by user **Y** and down-vote it(i.e. mark it as non-relevant). This might decrease user **Y** reputation severely and thus decreasing his importance. On the contrary, if user **X** likes user **Y**, user **X** can actually up-vote all questions and answers posted by user **Y** and increase user **Y** reputation. Such users usually post unrelated answers or comments randomly on some other people useful questions or answers or comments. There might be a group of users who are involved in such kind of activity. Their victim might be a single user or a group of users. I call them fraudulent users and I am interested in finding out group/s of users involved in such activity.

2 RELATED WORK

2.1 Modeling dynamic behavior in large evolving graphs

The paper [2] talks about detecting role of each node in a network and detecting unusual temporal behavior transitions. They use number of mutual friends(NMF) and minimum description length(MDL) based role extraction algorithm which computes the node group memberships and then based on some previous timestamps a role transition model is generated and the anomaly score for each node is calculated for a given time stamp. Their model is designed for (a) identifying patterns and trends of nodes and network states based on the temporal behavior, (b) predicting future structural changes, and (c) detecting unusual temporal behavior transitions in a network. They have experimented on twitter, IMDB, facebook and many other datasets and have achieved a remarkable result compared to the baseline.

2.2 Event detection in time series of mobile communication graphs

The paper [1] extracts behaviors for each of the nodes in the networks based on the features for all nodes in the graph. The 12 features considered in this work are: 1) in-degree, 2) out-degree, 3)out-weight, 5) number of neighbors, 6) reciprocal neighbors, 7) number of triangles, 8) in-weight, 9) average out-weight, 10) maximum in weight, 11) maximum out-weight, and finally 12) maximum weight ratio on reciprocated edges egonet.

They first create a NxT matrix for each features where N represents the number of nodes and T represents the timestamps. So, for F-features there are 3D NxTxF matrix. Then they build a correlation matrix **C** representing the correlation of behavior between all pairs of nodes in the graph over a time window W using Pearsons rho. Then, they compute the principal eigen vector for each window W for all the nodes that falls in window W. They compare it to recent past eigen vectors detected over several previous time windows. If the current behavior is found to be significantly different than recent past, they flag the current time window as anomalous and report as an event has occurred.

The similarity between the "behaviors" is evaluated using the dot-product. For low similarity between a node's "behavior" and its past "behaviors", the node is reported as anomalous for the corresponding time window. They have used the techniques in sms networks and assigns the weights to each of the edges based on the number of sms sent between the nodes. They could not verify the effectiveness of the anomaly detection because of the lack of ground truth but they were able to find the time when the large deviation in behavior of the nodes were seen. This usually happened in occasion like New Year, Christmas, etc.

I am replicating the same process using the out-degree as a feature.

3 DATASET

The stack overflow dataset has been downloaded from http://snap.stanford.edu/data/sx-stackoverflow.txt.gz. The format of the dataset is **(u, v, t)** which means that user **u** made some activity on user **v** at time-stamp **t**. The data is organized so the only thing I did on the dataset before using it was to convert the timestamps to the actual date-time format.

The dataset is 1.6 GB in size and contains records of 2774 days i.e. starting from the year 2008 A.D. to 2016 A.D.. The dataset consists of 63497050 edges. It's a very big dataset and thus a reduction of the dataset is necessary. So, I decided to use the records from only 2016 A.D.. The year 2016 has a total of 65 days data having 435894 nodes and 2536525 edges.

I think this data set is appropriate because with this information we can visualize the user or group of users who are contributing more to the network at different time instances. We can also visualize if there is a group of people who are always making activity on a particular user or a particular group of users. We also get to analyze if a particular group of users are making more activity in a very short period of time. If a user is making activity way more than average people in the community, he might be a suspicious user. This information can also be studied from the provided dataset. It would have been more helpful if it had the data about the up-vote and down-vote though. Also, because the dataset has the actual user id, we can verify the result of our findings(fraudulent or non-fraudulent) by visiting their profile manually.

$$\rho_{X,Y} = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

Figure 1: Pearsons Correlation

4 METHODOLOGY

I am using un-supervised based anomaly detection because I don't have labeled data. I have decided to use the methods from the paper [1]. Below are the methods that are necessary to use the processes described in paper [1]. Figure 2 shows the general overall process. I will be using NetworkX, a Python package, for loading the stack overflow data, creating graphs and performing analysis on it.

4.1 Sampling of Dataset

The dataset of 2016 still was big for the machine on which I was going to perform the experiments on. So, I had to perform the sampling of the database. For sampling, I first grouped the data by source node and timestamp which gave the total number of activity performed by a each user in 2016. From the analysis, I noticed that the minimum activity was 1 whereas the maximum activity was 4998. Secondly, I divided the range 1 to 4998 in an interval of 50 and got at most 50 source node from each interval. This was done to make sure that the sampled data will have source nodes having all possible activity activity (i.e. to make the sampled data as realistic as possible). Finally, the sampled data had 817 nodes which is around 0.19% of the original nodes in the year 2016. Thus, the sampling resulted in a heavy loss of the original data but my machine could not handle more than this.

4.2 Selecting Graph Type

Networkx supports different types of graphs. Selecting a correct type of graph is crucial to performing a social network analysis because different type of graphs are suitable for different type of analysis. The provided dataset contains interaction between nodes at a given time-stamp such that the interactions between a given pair of nodes at different timestamps hold different significance. A directed graph with multiple interaction at different timestamps is more clearly represented by a multi-directed graph. Thus, multi-directed graph has been chosen to use with the stack overflow data.

4.3 Feature Selection and Window Selection

The paper [1] talks about 12 different node features as mentioned in Section 2.2. It would be interesting to perform experiments on each of these features and see which of the features perform best. Because of the limited time, I can only test using one feature.

I have selected to use out-degree as the feature. The feature selection is based on my hypothesis - "*Nodes that are answering or commenting on other persons questions, answers or comments following their normal behavior would generally have a high reputation. Nodes that performs activity in burst (i.e. anomalous pattern) or who performs many activities have less reputation on stack overflow if they are fraudulent users*".

I have decided to use a window of length 7 days to study the behavior of each node because some user may decide to be active on weekends while other may decide to be active on weekdays. Considering this general behavior of human being, I chose to go with a window 7 days.

4.4 Formation of out-degree matrix

An out-degree matrix of size T*N was necessary to store the out-degree of each of the nodes at each window. T=0 represents January 1, 2016 and T=65 represents March 6, 2016. There are data from 65 days and total nodes of 817. Therefore, the size of out-degree matrix is 65*817. To find the out-degree of on each day, the data with time-stamp belonging up to that day was taken and an aggregated out-degree was found. If a matrix named out-degree-matrix is used to store the out-degrees then out-degree[0][0] represents the out-degree of the node 0 on January 1, 2016 whereas out-degree-matrix[1][0] represents the out-degree of node 0 on January 2, 2016 and so on. Algorithm 1 shows the process to calculate the out-degree of every vertex at each day. It us based on unix timestamp manipulations.

Algorithm 1 Forms a out-degree matrix of T*N to store the out-degree of each vertex each day

```
procedure FINDOUTDEGREE(verticesOfInterest, originalDataFrame)
    startTime ← 1451606400        ▷ Timestamp starting of Jan 1
    endTime ← 1457222400          ▷ Timestamp ending March 7
    anchorTime = startTime                    ▷ Temporary state
    while (anchorTime < endTime) do
        tempGraph = nx.MultiDiGraph()  ▷ Create temp graph
        anchorTime = anchorTime + 86400 ▷ Increase by 1 day
        tempDF <- records from originalDataFrame that has timestamp less than anchorTime
        tempGraph.addEdgesFrom(tempDF)
        Find the out-degree of vertices of interest from tempGraph and add to out-degree matrix.
```

4.5 Forming a correlation matrix of dimension N*N

A window size of W=7 days is chosen. For each window correlation matrix C, where C(x,y) is a pearsonś correlation between each pair of nodes x,y over window W. Correlation matrix is calculated using the out-degree matrix. When the correlation matrix for the current window has been calculated, the window slides down one day. The sliding process creates a new window and computes the correlation over the this window. The process is repeated until the end of the data has been reached. As a result, a total of 59 C matrices will be created.

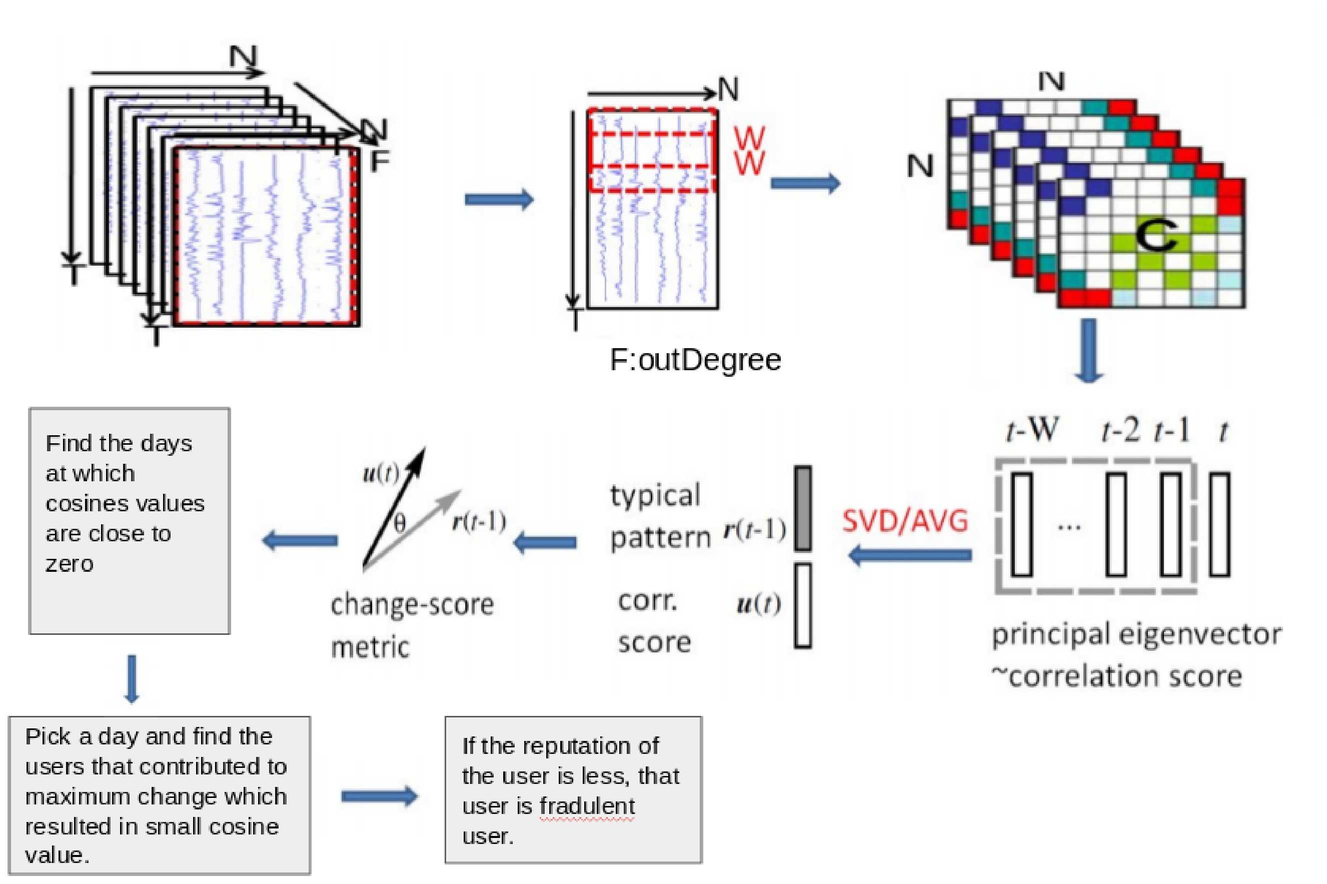
N
N
F
T
T
N
W
W
F:outDegree
N
N
C
t-W
t-2
t-1
t
...
principal eigenvector
~correlation score
SVD/AVG
typical
pattern
r(t-1)
corr.
score
u(t)
u(t)
θ
r(t-1)
change-score
metric
Find the days at which cosines values are close to zero
Pick a day and find the users that contributed to maximum change which resulted in small cosine value.
If the reputation of the user is less, that user is fradulent user.

Figure 2: General Block Diagram

Algorithm 2 Find Correlation Matrices for each window size W

```
procedure FINDCORRELATIONMATRICES($W, N, T, matrix$)
    $num\_of\_windows \leftarrow T - W + 1$
    for (i in range(num_of_windows)) do    ▷ Initialize matrices to zero
        $correlation\_matrix_i \leftarrow numpy.zeros(N, N)$
    for (k in range(num_of_windows)) do
        for (i in range(N)) do
            $a \leftarrow numpy.transpose(matrix[k : W+k, i : i+1)[0]$
            for (j in range(N)) do
                $b \leftarrow numpy.transpose(matrix[k : W + k, j : j + 1)[0]$
                $corr \leftarrow pearsonr(a, b)$
                $correlation\_matrix_k[i][j] = corr$
```

Algorithm 3 Returns the Average of Eigen Vector upto window (t-1)

```
1: procedure R($t - 1$)                          ▷ represents r(t-1) of Figure 2
2:     $sum \leftarrow numpy.zeros(N)$             ▷ Initialize array to zeros
3:     $average \leftarrow numpy.zeros(N)$         ▷ Initialize array to zeros
4:     for (i in range(t)) do                    ▷ Iterate through every window in i
5:         $curr \leftarrow eigen\_vector_i$       ▷ eigen_vector at window i
6:         $sum \leftarrow numpy.add(sum, curr.reshape(1, -1))$
7:     $average \leftarrow numpy.divide(sum, t - 1)$
8:     return $average$                          ▷ returns average of eigen vectors
```

The process of sliding the window is shown in the second step of Figure2. Algorithm 2 shows the steps for calculating the correlation matrices. Figure 1 shows the formula for calculating the pearsonś coefficient.

4.6 Finding the Eigen Vector

The process of finding eigen vector for each of the correlation matrices C was done by using the numpy.linalg.eig library. Among the eigen vectors, the principal eigen vector was chosen. The value for each node in the eigenvector can be thought as the activity of that node.

4.7 Calculating the change-score metric and anomaly detection

Cosine similarity was used to find the change-score of each node. To compute the change-score at a window number t, I first calculated the eigen vector of correlation matrix generated at window t and calculated the average of the eigen-vectors of all windows before 't' and secondly computed the cosine similarity between them. This gives a list of windows at which highest change-score was measure. Then I calculate the nodes that contributed to the highest change during those days. This was done by subtracting the eigen vectors calculated at window t to average of all the eigen vectors till window (t-1). The top 20 nodes that shows the highest deviation are predicted as anomalous users.

5 EXPERIMENTAL RESULTS

The experiments was ran the on my personal laptop. The hardware configurations are Intel Core i5-7200U CPU 2.50GHz X 4 with 7.6 GiB RAM.

During the calculation of the similarity score, the threshold for the similarity score was set to be 0.5. There were 26 windows that showed a similarity score of less than 0.5. The top 5 windows that showed the least cosine score were tracked. Then for each of five suspicious window, 10 of the users that contributed to the suspicious activity were tracked. To find the user that showed more suspicious behavior in any window w', element wise subtraction of the eigen vectors was done with the average of the eigen vectors up to window w'-1. The one that showed the highest difference were counted as suspicious node. Table 1, Table 2, Table 3,Table 4 and Table 5 shows the top 5 windows that showed the maximum suspicious activity and each table represents the top 10 stack overflow users that showed anomalous behavior. The top 5 windows that showed the anomalous behavior were found to be Jan 28 to Feb 3, Jan 8 to Jan 14, Jan 3 to Jan 9, Jan 14 to Jan 20 and Jan 6 to Jan 12.

Figure 3 shows how an ideal graph looks like when there is no suspicious activity. It is a straight line passing through the origin. Figure 4 shows the plot of the eigen vectors for window January 18 - January 24 with average of the eigen vectors up to January 25. This represents suspicious activity with cosine similarity score of 0.094. The dots in the scatter plots represents nodes. The nodes that does not falls on the straight line passing through the origin are responsible for the anomalous behavior. There are many nodes that does not satisfy the straight line behavior but it is difficult to select for every nodes so I have only predicted top 10 nodes as suspicious users.

6 EVALUATION

The outcome of this project gave the ids of a possible set of anomalous users that are performing the activity different from their general normal behavior. The correctness of the result was verified manually. As, the ids represent the actual ids of the stack overflow users, so the verification of the results was done by looking at the reputation of the users earned so far. The performance of project is be expressed as the percentage of the anomalous users detected correctly.

Unfortunately, every nodes that were detected as anomalous had high reputation greater than 9000 except for a user id 4606013 for window Jan 3 to Jan 9 which showed a reputation of 16. When I checked the profile of the user with reputation 16, it was shown that the user made only asked one question and he had not performed any other activity. So, we cannot say that this user is anomalous. Thus, I was unable to verify my results. It might be because I was using old data set and comparing the results with the recent one.

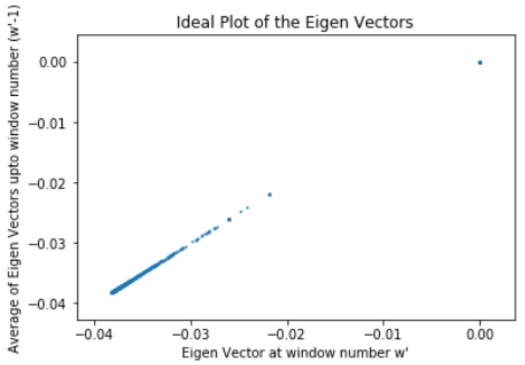

Figure 3: Ideal plot of Eigen Vectors between two different windows

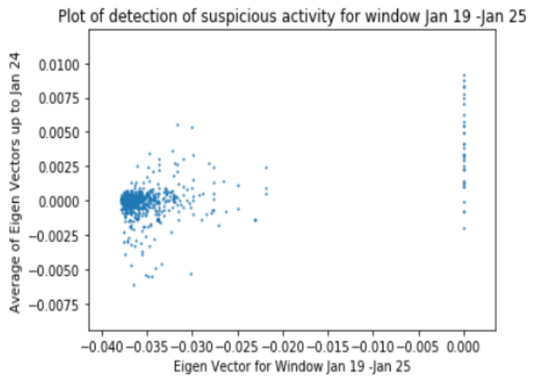

Figure 4: Plot of Eigen Vectors when anomalous activity was detected

7 CONCLUSION

It is unfortunate to see that the users that my project predicted as anomalous have higher reputation in stack overflow. There are four possible reasons that might have lead to such results. The first reason may be that the data-set was sampled at around 0.19% which might have lead to loss of useful information. The second reason may be the feature that was selected was not correct. The third reason might be that the research paper 2.2 was not suitable for this kind of project. The fourth reason may be that I was working on a dataset from 2016 but comparing the reputation at 2018. It's a 2.5 year gap and the users that were acting anomalously at that time may have been acting as a normal user at this time. So,the future work may include working on recent dataset and combining different features using deep learning approach or performing

Table 1: Predicted Suspicious User for window number 27 (i.e. during Jan 28-Feb 3)

User Ids	Reputation
467968	5027
4248328	55176
982161	32700
9204	104524
5104596	9627
10431	33653
131872	272760
3510736	51055
2649661	10612
1116364	11812

Table 2: Predicted Suspicious User for window number 7 (i.e. during Jan 8-Jan 14)

User Ids	Reputation
5202563	97681
2976878	44112
5535245	32203
6309	824143
3832970	303822
13302	567281
4279	259027
4573247	34581
16587	45374
2802040	73042

Table 3: Predicted Suspicious User for window number 13 (i.e. during Jan 14-Jan 20)

User Ids	Reputation
28804	283016
21475	65600
2976878	44112
7585	59291
2606013	68966
5202563	97681
1743880	87779
5535245	32203
2204926	11785
5044042	139594

experiments on each of the feature and choosing the one that performs better. So, there are enough room to improvise this project.

Table 4: Predicted Suspicious User for window number 5 (i.e. during Jan 6-Jan 12)

User Ids	Reputation
5202563	97681
5535245	32203
6309	824143
2606013	68966
2802040	73042
3832970	303822
7432	210032
1832058	41972
4573247	34581
330315	288214

Table 5: Predicted Suspicious User for window number 2 (i.e. during Jan 3-Jan 9)

User Ids	Reputation
1766831	65006
330315	288214
97337	197380
5202563	97681
3297613	141102
2606013	68966
4606013	16
50447	32120
1593860	56551
5067311	20259

REFERENCES

[1] Leman Akoglu and Christos Faloutsos. 2010. Event detection in time series of mobile communication graphs. In *Army science conference*. 77–79.

[2] Ryan A. Rossi, Brian Gallagher, Jennifer Neville, and Keith Henderson. 2013. Modeling Dynamic Behavior in Large Evolving Graphs. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM '13)*. ACM, New York, NY, USA, 667–676. https://doi.org/10.1145/2433396.2433479