ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Network Tour of Data Science

Project: Spammers on social network detection
Team no.20

Authors:
Hédi Fendri
Paul Jeha
Christina Mantonanaki
Nguyet Minh Nguyen

Supervisors:
Vandergheynst Pierre
Frossard Pascal

January 18, 2019

Contents

1 Introduction

In this project, we focus on detecting spammers in a social network. The ultimate purpose of this project is to label a user of our social network as being a spammer or not. The data set is collected from Tagged.com social network and the description is provided on [4]. There are 5.6 million users and 858 million links between them, each one is characterized by one relation (Eg: poke, message, profile view..), based on which we can build a graph. Unfortunately, the size of the data set is too enormous to be handled, so we had to sub-sample it by choosing only 5 relations out of 8. By considering only those relationship we have sampled our graph down to 783122 nodes.
Each user (node) has 3 features and is manually labeled as "spammer" or "not spammer". The aim of this project is to identify (i.e., classify) the spammer users based on their relational and non-relational features.

2 Method

Firstly, we will try to classify the nodes using the content features 2.1.1. We will keep adding these features one by one and compute the performance score to study their importance in detecting spammers on the social network.
Secondly, we will try to extract some relational features between the users 2.1.2 based on the topology and connectivity of the graph. Then, we will classify the nodes based on all the preceding features. We will see in details the different features and classification methods in the following subsections.

2.1 Features extraction

2.1.1 Content Features

In order to solve our main problem, we studied the features which could help us in identifying spammers. With the original dataset, we already have three basic features describing each node:

- Age range: the age group of a user, set to bins every 10 years.
- Time after validation: represents the time after joining the social network.
- Gender.

2.1.2 Graph based features

Furthermore, we should exploit the graph structure to compute features. Indeed, relations between nodes are provided with the dataset, we can then extract some graph-based features and add them to the content based features to build our final training matrix.
We created those features based on each type of relation separately. Since not all the nodes have edges for every type of relations, there are missing values, which we fill with mean values of the corresponding feature (filling them with 0 gives us the similar result).

We use Python's library, networkX, to generate features based on the graph. We use five graph analytic methods to compute the features described as follow:

- Pagerank: this will give us the ranking of a node inside our graph, which could tell us about the importance of one particular node. We expect that the spammer nodes would have a low ranking since those nodes would send more links to other nodes than receive from others[1].
- In degree, out degree, degree: This feature gives us the degree of a node. Similar to the argument above, with this feature, what we expect is that the spammers are prone to send more to others than receive back [2].
- Triangle counting and clustering coefficients: a triangle is a set of three nodes, where each node has a relationship to all other nodes. Clustering coefficient: for a normal person, if he/she is popular in the network, then another person knowing him/her would also know other people. We expect a spammer to have a suspiciously high clustering coefficient since spammers often try to spam as many people as possible [4].

We have 5 relations available, each corresponds to one sub-graph that we can work on separately. We then compute the six graph-based features described above on each sub-graph. Consequently, we have a feature vector for each node as follows: $\begin{bmatrix} X_{r1} & X_{r2} & X_{r3} & X_{r4} \end{bmatrix}$. Where X resprents the node, and r represent the relation. For each relation base-the graph features are computed as follows :$X_{ri} = \begin{bmatrix} X_{ri}^{m1} & ... & X_{ri}^{m7} \end{bmatrix}$

2.2 Features discussion

We are going to discuss the meaning of the features. Indeed, we can analyze the distribution of different features between the spammers and non-spammers:

2.2.1 Time after subscription

First, let's take a look at the time after subscription for the spammers and for the non-spammers. This is an interesting feature as we will see later on. Let's note something, we can easily discriminate the spammers from the non-spammers with this plot 1. Indeed the spammers are usually very recent, they just subscribed. The time after subscription for the non-spammers is more uniform. We have old and new users in this case. However, we think that this feature is not very relevant. Indeed, the fact that a majority of the spammers have a lifespan of less than 0.1 unit of time means that they stop their activity at this time. Therefore they are not active anymore, and detecting them would be "useless" as it is too late to stop them. In conclusion, this feature is good to discriminate the spammers from the non-spammers, but it comes too late.

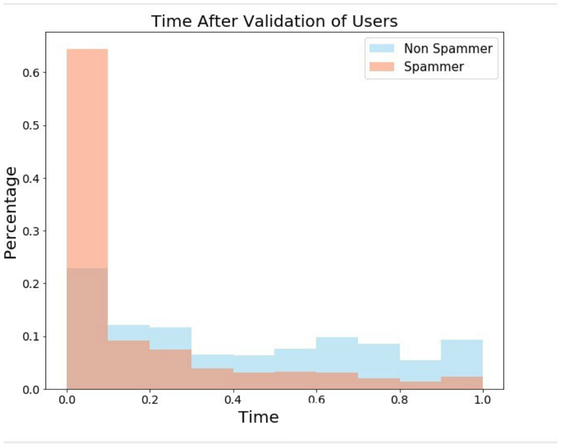

Figure 1: Distribution of users function of the Time after validation

2.2.2 Gender and age range

We can also think that maybe, spammers and non-spammers are discriminated by their age range or their gender. We could imagine that a majority of the spammers are 20 - 30 years old male. Let's look at the following plot: 2, 3 to see if such a pattern exists:

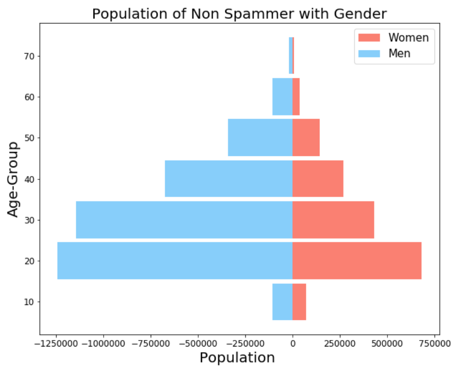

Figure 2: Demography of Non-Spammer

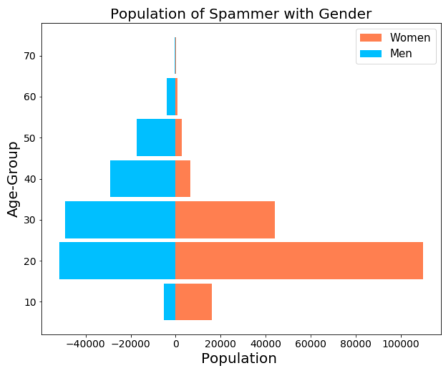

Figure 3: Demography of Spammer

Contradicting to what we have expected, in the spammer's society, young females are in fact who runs "the world". On the other hand, in a non-spammer's society, men are more dominant than women.

2.2.3 Degree distribution

Another intuitive idea that we can have is that spammers spam a lot of other nodes. So we can imagine that maybe, for a certain relation type we will have a big out-degree for the spammers, compared to non-spammers. Let's look at the plot 4 and see what conclusion can be drawn.

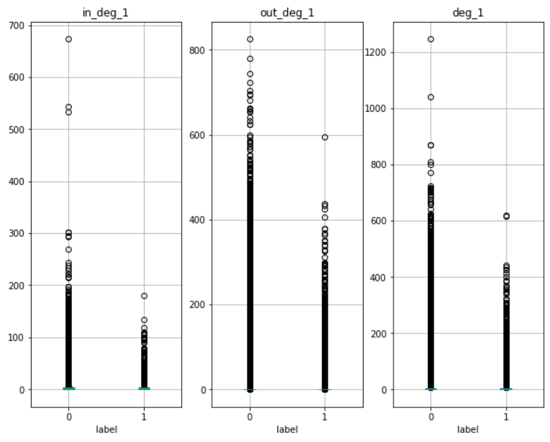

Figure 4: Degree distribution for relation 2

However, according to what we have studied, it turns out that the distribution of degrees of spammers and non-spammers share some similarity. They both have very low means and variance, which results in a tiny green boxplot. On the other hand, the outliers of non-spammers (may be the popular users), could have a very high degree compared to the outliers of spammers, which may complicate our approach to detect spammers using the in and out degrees.

2.3 Classification method

With the aim to classify our nodes into spammer or non-spammer, we use 5 methods to choose the most efficient one. Following are the methods we employ:

1. Support Vector Machine (SVM). The choice of using SVM as a classification algorithm is based on some related works. Many studies have used SVM to detect spams on social networks [5, 3]. There are two hyper-parameters we need to optimize: Gamma (γ) and the regularization parameter

2. K-Nearest neighbor (KNN) is a non-parametric method used for classification. To classify a sample we look at the label of the k closest training sample and choose the one that occurs the most. It is a model that we will use mostly when we have a small number of features. Indeed it performs the best in a small dimension. Two Hyper-parameters we need to optimize: The number of nearest neighbors K and the distance metric L.

3. Random Forests creates a set of decision trees from a randomly selected subset of the training set. It then aggregates the votes from different decision trees to decide the final class of the test object Two hyperparameters we have to tune: The number of decision trees in the forest and the number of features considered by each tree when splitting a node.

4. Linear discriminant analysis (LDA) is most commonly used as dimensional reduction technique while maintaining the class-discriminatory information, thus it has some practical uses as a classifier. This method has no hyperparameters to tune.

2.4 Hyper-parameters optimization

To optimize these hyperparameters, a simple k-fold cross-validation is used, where k = 5 is chosen as the number of folds. A grid-search over the values of Hyper-parameters is performed for each classifier algorithm. I.e., the

folds are iterated over and for each training fold, the classifier is trained.
The error found by this cross-validation is stored in a validation matrix containing a performance score for each fold and for each hyper-parameter. The validation scores are then averaged over all folds, reducing the dimensions of the validation matrix by one. Optimal hyper-parameters are found by choosing those with the highest fold-averaged validation score.

2.5 Evaluation metric

The dataset in this project is very imbalanced, indeed 90% of the dataset is made of non-spammers. If we decided to use the accuracy metric and predicted only non-spammers we would have a 90% accuracy! Obviously, this result does not reflect how bad our result is. Therefore it is important to choose another metric to evaluate our results.
To be able to evaluate our model we split our datasets into training and testing sets with a ratio of 0.35. We optimize the hyperparameters and train our model using the training set and evaluate it using the testing set. One classic evaluation metric that we will use is instead of the *accuracy* is the F-measure:

$$\text{F-measure} = 2\frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{1}$$

The F score (F-measure) is defined as the weighted harmonic mean of the test's precision and recall. Precision is the proportion of positive results that truly are positive. Recall is the ability of a test to correctly identify positive results to get the true positive rate. This technique takes into consideration the misclassification for each class and computes the performance score for each of them. This way, we can weight the global performance score according to the size of each different class of our dataset. To have a graphic visualization of the performances for different models the precision-recall curve is also plotted. The ROC (a plot of the true positive rate against the false positive rate.) curve could also be useful. We report then the area under Precision-Recall curve (AUPR) and the area under the ROC curve (AUROC).

3 Results

3.1 Classification using only content features

The table 1 shows the average results of classification using only content features. Firstly we classify the nodes using age range and genders as features. Secondly, we add the time after validation as the third feature. Results are depicted below:

Table 1: Classification with graph-based features using different algorithms.

classifier type	F-measure	Accuracy	AUROC
SVM age+gender	0.56	0.65	0.63
SVM age+gender+time	0.64	0.72	0.7
KNN age+gender	0.56	0.65	0.63
KNN age+gender+time	0.64	0.73	0.72
LDA age+gender	0.56	0.65	0.63
LDA age+gender+time	0.66	0.68	0.69
Random forest age+gender	0.56	0.66	0.64
Random forest age+gender+time	0.65	0.74	0.72

According to our classification results, we see that using only the age and gender do not provide us with adequate performance. However, when we add the Time after validation feature, the classification quality is noticeably improved by 7 - 8 percents. This effect is what we have expected from the study on the Time after validation feature

3.2 Classification using content and graph-based features

In this part, we added the graph-based features described in section 2.1.2. We compute the performance score. Results are depicted in the table 2 bellow:

Table 2: Classification with graph-based features using different algorithms.

classifier type	F-measure	Accuracy	AUROC
SVM	0.65	0.69	0.69
KNN	0.64	0.73	0.71
LDA	0.65	0.68	0.68
Random forest	0.66	0.74	0.71

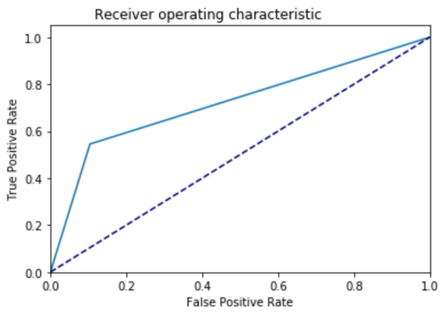

Figure 5: ROC curve using Random Forest

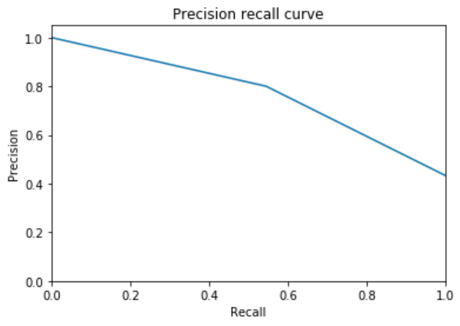

Figure 6: Precision-Recall curve using Random Forest

From the results table, we observe that even though we have added more features, the performance is not better than the previous one. There are some explanations for this consequence. First, although we added more features, we are using the same number of data points, which may cause us trouble from the curse of dimensionality. Thus, we should try to work with a larger amount of data. Second, we have not checked the quality of the feature, so we do not know if some features are actually beneficial for our classification. To improve this, in further study, we would try to use forward selection or backward elimination to extract the best features for our classification. Finally, we anticipate that maybe the spammers themselves know how to camouflage well enough so that by using our classical algorithms it would be challenging to detect them.

4 Improvements

In order to improve the performance of our classification, a feature we could have studied is a sequential based feature. Indeed, as each relation is an action it occurs at a certain point in time. Thus for two same nodes, we can create a feature collecting all the time of action for each relation. Then we can look at how close two actions are, for example, considering two nodes (1 & 2) and the relation "message" and "saw the profile". Two non-spammers could take 10 minutes between seeing the profile and sending the first message, whereas for a spammer only maybe 10 seconds. Thus, it could have been an interesting feature to study, which we would surely do for further study if possible.

References

[1] Akoglu, J. Y. L. Discovering opinion spammer groups by network footprints. In *Proceedings of the 2015 ACM on Conference on Online Social Networks* (New York, NY, USA, 2015), ACM.

[2] Johan Ugander, Brian Karrer, L. B., and Marlow, C. The anatomy of the facebook social graph.

[3] Pathan, S., and Goudar, R. H. Detection of spam messages in social networks based on svm. *International Journal of Computer Applications 145*, 10 (2016), 34–38.

[4] Shobeir Fakhraei, James Foulds, L. G. M. S. Collective spammer detection in evolving multi-relational social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2015), ACM, pp. 1769 – 1778.

[5] Zhu, Y., Wang, X., Zhong, E., Liu, N., Li, H., and Yang, Q. Discovering spammers in social networks. In *AAAI Conference on Artificial Intelligence* (2012).