Spam Detection for an Online Dating Website

Jonathan Brophy

Abstract— **The presence of spammers in various social networks is a long standing problem that has seen the evolution of more sophisticated and complex spam campaigns. New and advanced techniques are necessary to combat these ever changing efforts to maintain a spam-free environment. Motivated by the multi-relational data from the dating website, *if(we).com*[1], this paper shows the advantage of using graphical and sequential features to improve spam detection for this domain. The multi-relational structure of the social network is modeled as a graph of vertices and edges, where the vertices represent users, and the edges represent the relations between users. All relations between users are time-stamped allowing for extraction of sequential features in addition to the graphical features. The graphical features are created with *Graphlab Create*[2], and the time-stamped nature of the relations enable the creation of k-gram sequential features. The experiments show that the inclusion of these features improve the efficacy of spam detection in this domain while using AUROC and AUPR as the primary metrics for evaluation of this model.**

I. Introduction

For the vast majority of businesses and social networks, spam is treated as an unsolicited and unwanted aspect of their domain. The demand for more powerful and adept spam detection systems are needed to keep up with the increasingly advanced and varying spam campaigns that inhibit the full potential of a users' experience.

The users marked as spam in this dataset[3] have already passed an initial spam filter, which is why a more powerful model is needed to identify these spammers. One important aspect to note about this dataset is that it is content-less. There is no information about what messages these users send to one another. The only aspects known are demographic information about the user, how long a user has been in the system since registration, and a time-stamped list of actions between users for a period of ten days.

[1]Formerly *tagged.com*

[2]https://dato.com/products/create/

[3]https://obj.umiacs.umd.edu/tagged_social_spam/index.html contains an anonymized sample of the full dataset used in [1].

There has already been significant work done in this domain as of now [1], so the main purpose of this report is to replicate those results and possibly improve upon them by exploring features that were either missed or not taken into account. Once a baseline linear model has been created, then more advanced methods such as collective classification will start to be explored.

II. Background

A. Users

To get a better understanding of the data in this domain, it is useful to know the basic statistics about the users:

Non-Spammers:	5,270,494
Spammers:	336,953

We can see that there is a high class imbalance with only %6.1 of the users as spammers.

If we look at other information such as demographics, we start to get a more comprehensive picture about the users in this domain.

TABLE I
Average Demographics for Users

	Sex	Time Since Registration	Age
Non-Spammer	0.31	0.41	30.3
Spammer	0.54	0.15	26.7

Table I above shows the average demographic information for spammers and non-spammers. Males are encoded as 0 and females as 1, so a non-spammer is more likely to be male, whereas the number of male and female spammers are roughly split in half. Also, spammers tend not to be in the system as long as non-spammers once they are registered. One final thing to note is that the average age for a non-spammer is about 30 years old and a spammer is around 27 years old.

Slicing the data even further based on the user's age, which is binned into groups of ten years, will tell us more about the spammers.

We can see from Table II that the majority of users collected in this data are mainly 20 and 30 year olds. Also, the distribution of non-spammers is roughly

TABLE II
NON-SPAMMERS VS. SPAMMERS BY AGE GROUP

Age	10	20	30	40	50	60	70
NS	177k	1.92m	1.57m	944k	485k	143k	27k
S	22k	161k	93k	36k	20k	5k	800

proportional to the distribution of spammers, based on these age groups.

B. Relations

For the roughly 5.6 million users in this data set, there are roughly 800 million actions involving some type of interaction between a subset of them (not all users who registered for the site participated in any actions).

I will refer to an action between two users as a relation. There are 8 anonymous relations in this data set, which was collected over a ten day period. Each relation contains the type of action (wink, message, visiting a profile, etc...), the day (one of the ten days taken to record this data), time of day, the user who performed the action, and the user who received the action.

Using this information, a model of these relations can be constructed using a graph, where the users represent vertices and the relations represent edges between them. Since each relation is also time-stamped, this provides the opportunity to create time and sequenced based features. The relational structure of this network is the focus of this paper, as the features created from these relations will aid in generating a more accurate probability prediction of whether or not a user is a spammer.

III. METHODOLOGY

This paper focuses on three different approaches to improving spam detection in this domain. The first approach comes from the extraction of time-based features using the relations in the data set. The second involves building features around the sequential nature of the relations. Finally, a number of connected networks are modeled and mined to create a set of graphical features.

A. Time-Based Features (X_T)

Since the the anonymized day for each relation is given, a feature x_d is created which averages the days for which a user performs their actions. The idea is that a spammer may possibly perform the majority of their actions during a certain day, perhaps during the week where most people do not have as much free time.

The second feature x_t is the average time of day a user performs their actions. Users who are spammers might perform actions during the day or in the middle of the night where the majority of users may not be as likely to perform any actions.

Finally, the third feature x_v is simply the amount of time passed since a user registers with the website. We have already seen in section II that a spammer is on average more likely to be labeled as a spammer shortly after registering.

Together, these values make up a small time-based feature set:

$$X_T = [x_d, x_t, x_v]$$

B. Sequential Features (X_S)

Sequence-based features have been useful in modeling various text classification and natural language processing tasks [2]. The same idea can be applied to time-series data to improve classification [3], and we can use this methodology for the relations in this domain.

To create a sequential feature set from these relations, the first thing to do is to sort all relations by their time-stamps and by the users performing the actions. After sorting, a bigram approach will be used to create the feature set where a single feature for a user u_1:

$x_{r_1 r_2}$ = # times u_1 performs r_2 immediately after r_1

Since there are 8 (0-7) different relations, there are 64 possible combinations of two sequential relations:

$$X_S = [x_{r_0 r_0}, x_{r_0 r_1} ..., x_{r_7 r_6}, x_{r_7 r_7}]$$

Even with around 5.6 million users and about 800 million relations, we should expect to see zeros for many of the features in X_S as it is not very likely that most users will perform every combination of relations generated in X_S. Therefore, picking a classifier with an appropriate regularizer may help combat this dilemma.

A tri-gram of sequential relations could be explored, but the number of features would quickly get out of hand and exacerbate the data sparsity problem mentioned above.

C. Graphical Features (X_G)

As stated earlier, there are 8 relations, and we can thus construct a network of users pertaining to each relation. For example, if we consider only the relation r1, then we can connect the users who have performed

r1 to their respective destination users. This creates a web of connections only involving r1, and from this network we can compute graphical features based on the following methods:

1) Triangle Counting: This method will count the number of triangles a vertex is involved in, giving some importance to how connected a user is [4]. The number of triangles a vertex is involved in is used as a feature.

2) K-Core: As an iterative approach, each round removes the least connected vertices in the graph and assigns those vertices the number corresponding to the round it was removed in [5]. For example, if a vertex was removed on the first round, it would get a 0, but if it were removed on round 7, it would get a 7. The round in which the vertex is removed is used as a feature.

3) Graph Coloring: The idea of graph coloring is to assign a color to every vertex making sure not to color two adjacent vertices the same [6]. The color id assigned to each vertex is used as a feature.

4) Pagerank: This is a well known algorithm to rank web pages. Web pages with lots of incoming links are viewed as more important. The same idea can be applied to our relational graphs [7]. The importance given to a vertex by the pagerank algorithm is used as a feature.

5) Weakly Connected Components: This method computes how many weakly connected components a given vertex is involved in. A weakly connected component is defined as a group of vertices where any two vertices in that group are directly connected [8]. The id and size of the weakly connected component the vertex is involved in are used as features.

6) Degree: The in-degree of a vertex represents the number of times a user is the recipient of an action, and the out-degree of a vertex represents the number of times a user is the one performing an action. Both the in and out degree of a vertex are used as features.

Each of these methods were computed on all 8 relational graph models giving a total of 40 graphical features for each user:

$$X_G = [x_{r_0}^{m_1}, x_{r_0}^{m_2}, ..., x_{r_0}^{m_6}, x_{r_1}^{m_1}, ...]$$

Where $X_{r_0}^{m_1}$ is the feature in X_G corresponding to the graph with relations of type r_0 where method m_1 has been applied. All of the graph models and features mentioned above have been created using Dato's efficient and scalable *Graphlab Create*[4].

[4]https://dato.com/products/create/

IV. Experiments

Experiments are performed on the features proposed in section III. First, the time features are tested using three different linear classifiers: Logistic Regression, Random Forests, and Gradient Boosted Trees [9] implemented using Scikit-Learn[5]. Then the sequential and graphical features are tested in isolation, and finally all features are modeled into one framework and evaluated.

A. Hyper-Parameter Tuning

Since the data set is rather large, ∼5.6m users to classify, all 3 classifiers are tuned using grid search cross validation on a subset of the training data to speed up the computational time. Grid search contains the ability to maximize different evaluation metrics while tuning these hyper-parameters. After some preliminary experimentation, roc_auc was the chosen metric to maximize as it seemed to improve both AUROC and AUPR scores on many experimental test sets.

For logistic regression, 3 hyper-parameters are tuned: penalty, C, and the class weight. Penalty is the regularizer, l1 or l2. C is the inverse of the regularization strength, where smaller values of C specify stronger regularization. The class weight can be either either none or balanced. If none, then the weight for each class is one. If balanced, then the weight of each class is inversely proportional to the class frequencies in the data set. Since there is such a high class imbalance, having a balanced class weight hyper-parameter may help focus the model on the rare spammer cases.

In the random forest classifier, 4 hyper-parameters are tuned: number of estimators, split criterion, number of max features, and the class weight. The default number of estimators is 10, but these experiments test a range of estimators from 5 to 20. The split criterion will be tuned to gini or entropy. If n is the number of features, then the experiments will try sqrt(n) and log2(n) for the number of max features. Finally, The class weight hyper-parameter is the same as in logistic regression, where it will be tuned to either none or balanced class weights.

The only hyper-parameter that will be tuned for gradient boosted trees will be the maximum depth for the trees. This is the only tuning done for this classifier for two reasons: First, differing values for the max depth of the trees will have the most profound effect on the accuracy of the model. Second, after preliminary experimentation, tuning this model was taking far too

[5]http://scikit-learn.org/stable/

long, and adding more hyper-parameters only increases the number of combinations to tune, exacerbating the problem. Thus different values of only max depth are tried from 1 to 7. The idea of having a max depth of 1 means having many decision stumps, which acts like an l1 regularizer and can learn a concept even from a sparse signal. Having a higher max depth will create more powerful trees, and can potentially learn a more complex function.

B. Metrics

Since spammers make up only %6.1 of the users, then a model classifying every user as a non-spammer would get an accuracy of $\sim$%94, producing misleading results.

A better metric is the ROC curve, which takes into account the true positive rate and the false positive rate for varying threshold levels. Even with this improved metric, though, ROC can still fall victim to class imbalance if not handled appropriately [10]. A large change in false positives will not be very noticeable in an ROC curve since the number of irrelevant items dwarf the number of relevant items.

Thus, even better metrics to use for this problem are precision and recall, from which a precision-recall curve can be created from differing thresholds. Precision and recall do not consider true negatives, making them suitable for a highly imbalanced class problem such as this.

Visually, ROC and AUPR curves are nice to look at and easy to compare, but we can also calculate the area under these curves to give one number for each metric and compare them.

C. Time-Based Features

Table III shows the classification results with time-based features X_T outlined in Section 3A for 3 different linear models.

TABLE III
CLASSIFICATION USING TIME-BASED FEATURES

Model	Acc	AUROC	AUPR
Logistic Regression	0.593	0.774	0.361
Random Forests	0.817	0.775	0.361
Boosted Trees	0.94	0.775	0.361

All of the models do a comparable job in terms of AUROC and AUPR for this small feature set, but boosted trees is able to maintain high accuracy in addition to AUROC and AUPR.

D. Sequential Features

After tuning the hyper-parameters using this feature set, logistic regression chose an l2 regularizer and a C value of 10, which indicates that the model did not want a strong regularizer. Both logistic regression and random forests chose balanced weight classes which makes sense as they try to deal with the high class imbalance. Boosted trees chose a max depth of 7, which allows it to have more complex trees per round compared to decision stumps.

TABLE IV
CLASSIFICATION USING SEQUENTIAL FEATURES

Model	Acc	AUROC	AUPR
Logistic Regression	0.697	0.723	0.210
Random Forests	0.948	0.731	0.342
Boosted Trees	0.947	0.755	0.341

Analyzing the results from Table IV, it seems as though logistic regression may not be a powerful enough model to take full advantage of the sequential feature set. Random forests and boosted trees do comparatively well, with boosted trees slightly outperforming random forests in terms of AUROC.

E. Graphical Features

The graphical feature set seems to be a better predictor than both the time-based and sequential feature sets. Table V shows that the performance of logistic regression still lags behind random forests and boosted trees, where boosted trees once again slightly outperform random forests.

TABLE V
CLASSIFICATION USING GRAPHICAL FEATURES

Model	Acc	AUROC	AUPR
Logistic Regression	0.750	0.749	0.270
Random Forests	0.949	0.742	0.395
Boosted Trees	0.949	0.790	0.392

F. Full Framework

Since random forests and boosted trees have consistently outperformed logistic regression in the last three experiments, this experiment focuses on just one algorithm to create the linear model, boosted trees. Boosted trees continues to slightly outperform random forests, and its boosting effect naturally takes the class imbalance into account, where it focuses more on the

examples it gets wrong. This also allows more time to explore different feature combinations without having to run them on the other two linear models.

For this experiment, all of the previously mentioned feature sets: X_T, X_S, and X_G are combined in combination with two additional features: sex and age, which encapsulate the demographic features X_D of the user:

$$X_D = [x_s, x_a]$$

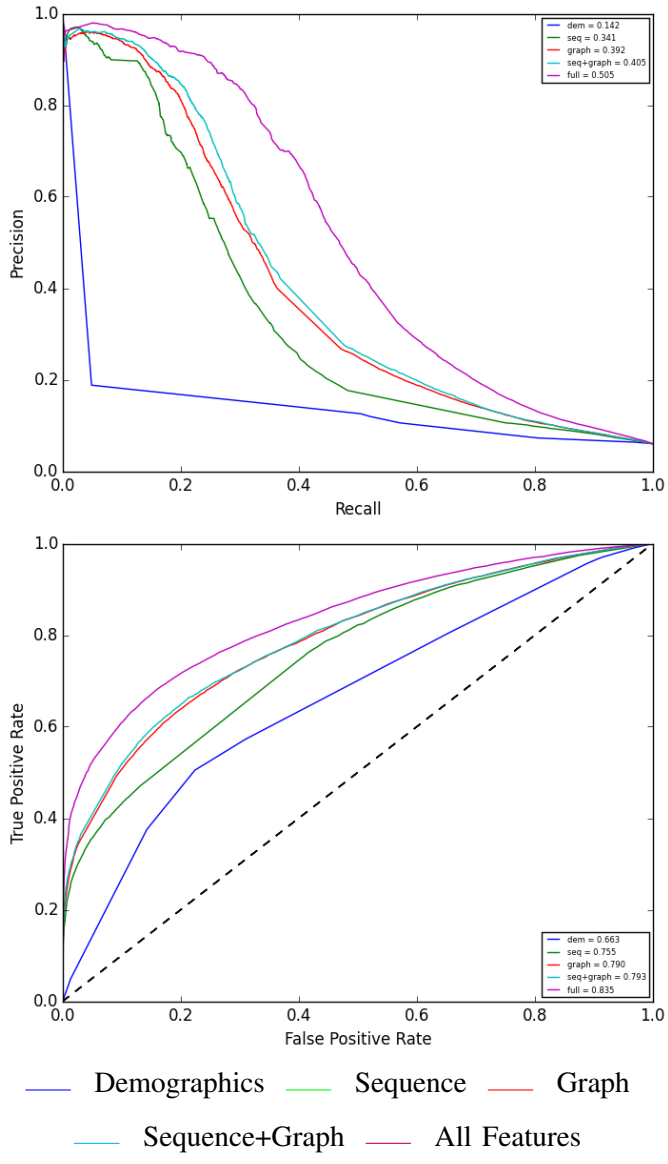

Fig. 1. Precision-Recall Curves (Top) and ROC Curves (Bottom) for Different Feature Sets.

According to Figure 1, the demographic features are the least discriminative attributes for this data set. The graphical features perform slightly better than the bigram sequences, but combining the sequential and graphical features does not improve the efficacy of the model by much. The results for the full framework in the last line in Table VI show the best scores for AUROC and AUPR compared to all other feature sets. This could be due to the time feature set X_T added into the full framework, where the time passed since registration attribute seems to be a highly discriminative feature.

TABLE VI
BOOSTED TREES WITH DIFFERENT FEATURE SETS

Features	Acc	AUROC	AUPR
$[X_D]$	0.94	0.663	0.142
$[X_S, X_G]$	0.95	0.793	0.405
$[X_D, X_T, X_S, X_G]$	0.955	0.835	0.505

V. CONCLUSIONS

After replicating and comparing to many of the experiments in [1], the results in this paper are not quite as good for two reasons: First, the data made available to the public for this domain [6] is only a sample of the data used in their paper. Second, several relations have been taken out of the public data set for legal reasons, where these relations could possibly improve the efficacy of the model. Also, the spammer labels have been updated with the release of this public data set.

Taking the above into consideration, I believe the results from these experiments serve as a solid baseline linear model to improve upon with more advanced techniques such as collective classification. The next step is to take the credibility reports of users into account in order to more accurately decide who the spammers are.

REFERENCES

[1] Fakhraei, S.; Foulds, J.; Shashanka, M.; and Getoor, L. 2015. Collective Spammer Detection in Evolving Multi-Relational Social Networks. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Pages 1769-1778.

[2] Wang, X.; McCallum, A.; and Wei, X. 2007 Topical N-grams: Phrase and Topic Discovery, with an Application to Information Retrieval. ICDM '07 Proceedings of the 2007 Seventh IEEE International Conference on Data Mining. Pages 697-702.

[3] Xing, Z.; Pei, J.; and Keogh, E. 2010. A brief survey on sequence classification. ACM SIGKDD Explorations Newsletter.

[4] Schank, T. 2007. Algorithmic aspects of triangle-based network analysis. Phd in computer science, University Karlsruhe.

[6] https://obj.umiacs.umd.edu/tagged_social_spam/index.html

[5] Alvarez-Hamelin, J.; DallAsta, L.; Barrat, A.; and Vespignani, A. 2005 Large scale networks fingerprinting and visualization using the k-core decomposition. In Advances in neural information processing systems (NIPS).
[6] Jensen, R.; and Toft, B. 2011. Graph coloring problems. John Wiley & Sons.
[7] Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1999. The pagerank citation ranking: Bringing order to the web. Technical report.
[8] Pemmaraju, S.; and Skiena, S. 2003. Implementing discrete mathematics: Combinatorics and graph theory with mathematica.
[9] Drucker, H. 1997. Improving Regressors using Boosting Techniques. submitted to the 1997 conference on Machine Learning.
[10] Davis, J.; and Goadrich, M. 2006 "The relationship between Precision-Recall and ROC curves." Proceedings of the 23rd international conference on Machine learning, ACM.