Detecting Malicious Twitter Users Using Mixed Features

Program: Summer VRES
Student: Sam Hardy
Supervisor: Yue Xu
Submitted: 14.02.18

Abstract
Whilst online communities can generally be said to have a positive impact upon the information sharing process, unscrupulous users often abuse their open nature, using these communities as a platform to disseminate malicious content. Correctly identifying and classifying such malicious behavior has proven challenging, yet it's importance remains. Building upon previous work in this area, a promising selection of meta features and content-derived features was created for use within a range of classifiers. A pre-existing and pre-labelled dataset was used as the basis for a series of classification experiments which combined these features with a preliminary clustering process. The accuracy of the classifications across the different cluster segmentations utilizing both feature types was generally good, with no cluster segment offering a decisive accuracy difference.

1.0 Introduction

1.1 Problem Overview

Online social communities provide a rich plethora of opportunities for users to interact, learn and engage with one another, providing many benefits not readily realized within their physical counterparts. For one, online communities typically rely upon users as the primary contributors and curators of content, facilitating the bottom-up, serendipitous discovery of citizen experts.[1] The diversity, breadth, and specificity of topics which these online communities bring to their users is capable of facilitating positive, novel discussions and interactions at unparalleled speeds.

The openness and scope for interaction that these communities provide also presents as an opportunity for unscrupulous users to organize and disseminate misinformation, commercial spam and traditional phishing and malware attacks.[2] The need to correctly identify, prevent and deter such behaviors is important in preserving the integrity of these online communities. The problem becomes all the more difficult given the agility and innovative nature of malicious users; sophisticated methods are often adopted to conceal, blend and obfuscate malicious activity much to the malign of legitimate users and the communities they are a part of.[3] The current work seeks to canvas some of the more recent methods regarding malicious user classification and consolidate some of the more promising methods into a larger classification scheme, specifically within a twitter context.

1.2 Data Collection Approaches

Crucial to developing relevant and robust classification models is the collection of relevant data, specifically data that accurately canvases the profiles of malicious users. The use of social honeypots has previously been utilized to tempt and collect malicious users.[4] Social honeypot methods were originally borrowed from security practices of baiting malicious users and taking subsequent action upon identifying their malicious intent, in much the same way that network and system intrusion detection schemes are implemented. Social honeypot methods are useful because they are capable of automatically collecting evidence of content polluters in a way that is unobtrusive to legitimate users.

The original honeypot method as articulated by Lee, Caverlee and Webb (2010) involved deploying a small network of users to myspace and twitter, upon which, their subsequent interactions with other users was recorded and monitored for malicious activity. This honeypot method was refined further within Lee, Eoff and Caverlee's (2011) later work, which involved a more extensive honeypot deployment and collection. Within the later work, an interconnected network of 60 social honeypot users was deployed over the course of 7 months upon twitter. The network of inter-connected users was manipulated in such a way that their activities would tempt and maximize interaction with malicious and spamful users. During that time 36,000 candidate content polluters was collected for analysis.

Subsequent to this later study, the "social honeypot" dataset has proven useful within later works, serving as a complimentary dataset alongside a Sina Webo collection for one.[5] Previous studies have also utilized the twitter public streaming API to generate datasets[6] and twitter's REST API.[7] Datasets have also been constructed in temporal ways, by specifically harvesting event-driven data occurring around natural calamities, sport events, terror attacks.[8]

[1] Lee, Caverlee and Webb, 2010
[2] *Ibid*
[3] *Ibid*
[4] *Ibid*
[5] Liu, Lu, Yuo, Zhang and Itti (2016)
[6] Clark, Williams, Jones
[7] Singh, Bansal and Sofat (2016)
[8] Dewan and Kumaraguru (2017)

1.3 Feature Engineering Approaches

A similarly important aspect of developing accurate classification models relates to the features that are created and utilized within the classifier. A review of the literature reveals a rich variety of features, reflective of this area's intersection of computer science, sociology, and political science.[9] Consolidating these features into a coherent taxonomy is difficult, though some attempts have been made to do this.[10] The work of Cossu, Labatut and Dugue (2016) specifically attempted to profile twitter-based features. This work helpfully outlined that features may subsist under the broad headings of social network analysis, Natural language processing and domain-specific features (post counts, follower counts etc.). For present purposes, a distinction is made between readily available, domain-based "meta" features, (post counts, follower counts etc.) and derived, content-based features (post-content similarity, content distribution analysis).

Features derived from a user’s meta information have been used extensively within the literature. Twitter account longevity, average tweets per day, ratio of following and followers, percentage of bidirectional friends, URL tweet ratios and URL incidence have been previously used as meta features, with the incidence of external URL links reported as having good discrimination power.[11] Additional meta features relating to User Demographics (length of users name, description section), User Friendship Networks (num. followings, bi-directional friendships), User Content (posted tweets, number of annotations) and User History (temporal change in followers) have been previously utilized to classify malicious users.[12] To this end, combinations of meta features have been utilized to generate features with improved depth, extending to "reputation".[13] Of course, malicious user activity is often more sophisticated and nuanced than the proxies these features seek to represent. Malicious users routinely emulate the daily cycle of human activity, hoodwink legitimate accounts for their own purposes, and actively infiltrate communities in sophisticated ways.[14] A meta approach to feature construction can therefore be misleading at times, thus requiring more complex feature development.

Content-based, Natural Language Processing methods provide a more context-oriented approach to malicious user classification. NLP methods provide a flexibility beyond the constraints of a particular domain, having possible application within other social media contexts.[15] Average content similarity over all pairs of tweets for a particular user using cosine similarity has been previously reported as having high discriminatory power, in which it was revealed that that spamful users at least are more likely to repetitively post a particular message.[16] Similarly, the lexical pairwise dissimilarity coupled with the word introduction decay rate resulted in high discriminatory power when bot users from legitimate users.[17] Topic modeling has also been utilized with a degree of success utilizing Latent Dirichlet Allocation distributions, specifically document-topic distributions.[18] These LDA derived features similarly posited that malicious users are likely to repetitively post about certain topics in a variety of subtle ways that are difficult to explicitly formalize via strict rules. As a result, utilizing a more generally applicable document-topic distribution provides a more nuanced way to capture highly focused, yet subtly different content. Interestingly, such LDA derived features maintained favorable discriminatory power even when applied to a related, trans-lingual domain, Sina Weibo in that instance.[19]

[9] Cossu, Labatut and Dugue (2016)
[10] *Ibid*
[11] Liu, Lu, Yuo, Zhang, Itti and Lu (2016)
[12] Lee, Eoff and Caverlee, 2011
[13] Singh, Bansal and Sofat (2016)
[14] Clark, Williams, Jones, Galbraith, Danforth and Dodds, 2015
[15] *Ibid*
[16] Lee, Caverlee and Webb, 2010
[17] Clark, Williams, Jones, Galbraith, Danforth and Dodds, 2015
[18] Liu, Lu, Yuo, Zhang, Itti and Lu (2016)
[19] *Ibid*

1.4 Current Work
It has been proposed that a combination of readily available meta features and "additional" features may be required to meet the ever-evolving sophistication of malicious techniques.[20] The current work takes its lead from this observation and proposes to combine the most potent meta and content-based NLP features articulated within the previous works with an intermediate clustering process. Such a clustering process was first described by Lee, Eoff and Caverlee (2011) as a way of generating user taxonomies, and its use within the current work will play a similar role by partitioning users into focused and unfocused groups, upon which individual classifications are performed upon these segmented groups utilizing combinations of domain-centric meta features and NLP features.

The current work will utilize the pre-existing honeypot dataset as created by Lee, Eoff and Caverlee (2011) as the basis for its experiments. During these experiments, the current work seeks to specifically answer:

- What effect, if any, does clustering users utilizing content-based features have upon final user classifications?
- What effect does combining the domain-centric meta features prominent within previous literature with emerging NLP/content features have upon the final user classifications?

2.0 Methodology
2.1 Data Handling
The social honeypot dataset created by Lee, Caverlee and Eoff (2011) within their longitudinal study has been utilized. The dataset contains close to 40,000 legitimate and content polluting users as well as 7 months' worth of historical tweet content. To reduce the risk of confounding results duplicate entries filtered on User ID have been dropped from the main dataset, as well as any entries containing NAN values. Duplicate entries shared by the legitimate and content polluter user subsets have also been dropped, to ensure for complementary datasets. The main dataset was segmented based upon users and tweets, with relevant subdivisions implemented (legitimate users, content polluter users) within these larger divisions.

2.2 Feature Development
The specification of meta and content-based features is naturally predicated upon two distinct feature sets, and this has been reflected to a high degree when developing the features. Meta features were comprised of the number of tweets, followers, followings, length of about me, length of user name, average number of annotations per tweet, average number of links per tweet, average tweet length and total number of unique words were calculated for all content pollutant users and legitimate users. These meta features were statically calculated without the need for additional preprocessing or parameter specification

Content-based features were comprised of the document-topic distribution entropy calculated across all topics for a particular user, derived from an LDA distribution as specified by Liu, Lu, Yuo, Zhang, Itti and Lu (2016). This LDA distribution was further used to calculate each users GOSS and LOSS scores across each of the topics contained within the distribution.[21]

Calculating these content-based features required additional data preprocessing as well as parameterized input for constructing the LDA distribution used to derive the content-based features. A document-tweet corpus was created, comprised of the grouped tweets associated with a user. An Sklearn count vector paired with a WordNet Lemmatizer was then used to create an intermediate term frequency matrix. All punctuation and common English stop words were discarded from the tokenization process, whilst n-grams in the range of 1,2 were used to configure the count vector alongside minimum/maximum document frequency thresholds of 0.15 and 0.85 respectively. In turn, an Sklearn Latent Dirichlet Allocation model was used to generate a document-topic

[20] Clark, Williams, Jones, Galbraith, Danforth and Dodds, 2015
[21] Liu, Lu, Yuo, Zhang, Itti and Lu (2016)

distribution based upon the term frequency matrix consisting of 5 topics. Once these parameterized, intermediate processes produced the required document-topic distribution, the previously specified content-based features could be calculated.

2.3 Preliminary Clustering

Once the meta features and content-based features were generated and fused into a single set of master features, an Sklearn K-means cluster model was used to segment all users based upon their document-topic entropy and the number of unique words contained within their tweet document. Three clusters fitted across 300 iterations was specified, commensurate with the first research question of determining whether a user's focused, balanced, or unfocused tweet-document would impact upon final classification.

	Cluster Size	**Malicious Users**	**Legitimate Users**
Cluster 1	12,595	6,236	6,359
Cluster 2	12,635	3,747	8,888
Cluster 3	14,342	3,640	10,702
Total	39,572	13,623	25,949

Figure 1: Cluster Composition

2.4 Classification

A series of classifiers was then created and evaluated for each of the segmented clusters. The averaged accuracy score calculated using Kfold cross validation with 10 folds and a train-test split of 0.6/0.4 was retrieved utilizing an Sklearn decision tree, random forest, adaboosted decision tree and linear support vector classifier. The precision, recall, and f1 scores were also calculated for each of the generated classifiers.

	Classifier	Averaged K-fold CV Accuracy	Precision	Recall	F1
Cluster 1	DT	0.86 (+	0.86	0.86	0.86
	Random Forest	0.90	0.90	0.90	0.90
	Adaboost	0.90	0.90	0.90	0.90
	Linear SVC	0.78	0.80	0.74	0.73
Cluster 2	DT	0.88	0.88	0.88	0.88
	Random Forest	0.91	0.91	0.91	0.91
	Adaboost	0.91	0.91	0.91	0.91
	Linear SVC	0.79	0.88	0.87	0.87
Cluster 3	DT	0.91	0.91	0.91	0.91
	Random Forest	**0.94**	**0.94**	**0.94**	**0.94**
	Adaboost	0.93	0.93	0.93	0.93
	Linear SVC	0.79	0.85	0.82	0.83

Figure 2: Classification Results

3.0 Results

Interestingly, the preliminary clustering process yielded a series of segmentations which were more equal than unequal. Despite these initial similarities, the composition contained within each cluster segmentation deviated away from one another. Clusters two and three reported class-specific skews towards legitimate users, whilst cluster one reported a more even class-specific split. The final cluster segmentations revealed a sizeable class difference, with their nearly being double the number of legitimate users as compared to malicious users. Whilst attempts to mitigate the accuracy-skewing effects of the class imbalance were made via the utilization of stratified, randomized train/test sampling, the effects persist and were clear within the confusion matrix outcomes for each classifier.

Classification wise, the third cluster saw the highest accuracy, precision, recall and F1 scores. Incidentally this cluster was also the most numerate of the three, comprised of 10,702 entries. Generally speaking, the linear SVC performed poorly, consistently being outperformed by the Random Forest and Adaboosted Decision Tree variants. High accuracy rates were recorded within the third cluster, reaching accuracy as high as 94% when utilizing a random forest classifier. Across all clusters however, there doesn't appear to be any decisive trend favoring one over the other, it would therefore appear that the preliminary clustering and subsequent segmentation has had a minimal effect upon the classifiers ability to distinguish between malicious and legitimate users.

4.0 Future Developments

The current work could be augmented and improved in several ways. Undertaking a more rigorous analysis of the resultant cluster segmentations may provide insight as to what features to utilize when constructing the final classifiers. Different combinations of features for clusters of users (eg. GOSS scores for focused users) may provide a way to optimize the classification process. The use of additional cluster analysis (Silhouette scores, intra cluster similarity) and clustering algorithms (DBScan, Affinity Propagation) may improve the segmentation, by improving the feedback mechanism of the clustering. Finally, utilizing some sort of hyperparameter optimization when attempting to fit the classification models would serve to generally improve their accuracy.

5.0 References

J. Cossu, V. Labatut, N. Dugue, A review of features for the discrimination of twitter users: application to the prediction of offline influence, *Social Network Analysis and Mining*, 2016

E. Clark, J. Williams, C. Jones, R. Galbraith, C. Danforth, P. Dodds, Sifting robotic from organic text: A natural language approach for detecting automation on Twitter, *Journal of Computational Science*, 2015

P. Dewan, P. Kumaraguru, Facebook Inspector (FbI): Towards automatic real-time detection of malicious content on Facebook, *Social Network Analysis and Mining*, 2017

K. Lee, J. Caverlee, S. Webb, Uncovering Social Spammers: Social Honeypots + Machine Learning, *SIGIR*, 2010

K. Lee, B. Eoff, J. Caverlee, Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter, *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, 2011

L. Liu, Y. Lu, Y. Yuo, R. Zhang, L. Itti, J. Lu, Detecting "Smart" Spamers on Social Network: A Topic Model Approach, 2016

M. Singh, D. Bansal, S. Sofat, Behavioral analysis and classification of spammers distributing pornographic content in social media, *Social Network Analysis and Mining*, 2016

6.0 User Manual

6.1 Overview

The functionality of the program can be conceptualized as a pipeline of operations, with each operation occupying its own discrete script. This design is reflected to a high degree within the *scripts* and *python_notebooks* subdirectories. Each operation within the pipeline exists as a standalone script capable of executing independently of its sibling scripts (within reason), and, bar the preprocessing script, is coupled with it's own configuration file allowing for parameter control over a given scripts relevant behavior.

The following will detail the operations performed by each of the subscripts, the main wrapper script, the accompanying data transformations that occur throughout the pipeline as well as any necessary configuration involved with running the scripts. Environment configuration and dependency management can be resolved by reading readme file contained within the code base.

6.2 Preprocessing

The operations contained within the preprocessing script can be summarized as follows:

- Import raw honey pot datasets for both static and dynamic features
- Preprocess static features and dynamic features data frames
- Square static and dynamic feature data frames, ensuring that a mutual set of *user_id's* are shared between the static and dynamic feature datasets
- Export the finalized static features, and export the intermediate dynamic features data frames

Currently, only preprocessing for the honeypot dataset exists. There is also currently no configuration required for the preprocessing, as the preprocessing behavior and output is statically defined. The preprocessing script can be incorporated within the main wrapper script or executed in isolation by locating to the preprocessing sub-directory and calling ***python3 hp_preprocessing.py****.*

6.3 Feature Engineering

The operations contained within the feature engineering script can be summarized as follows:

- Import the intermediate, dynamic features data frame (consolidated tweet-document corpus for all users)
- Configure and generate a count vector based upon the user configuration
- Create a term frequency matrix by vectorizing the consolidated tweet-document corpus utilizing the generated count vector, applying lemmatization, stemming etc. during the process
- Generate dynamic features using a document-topic distribution derived from the term frequency matrix (GOSS, LOSS, document-topic distribution entropy), based upon the user configuration
- Export the dynamic features data frame

The feature engineering scripts are probably the most complex scripts contained within the project. Additional sub scripts utilized during the feature engineering process are contained within this directory as *nlp_vector_config.py* and *dynamic_features.py* respectively. For simplicity's sake however, the relevant operations are invoked in a decisive way within *hp_dynamic_feature_generation.py.*

Parameter configuration for the count vector and the lda modeler is contained within ***configs/hp_fe_config.json***. Within this configuration file the n-gram range as well as the document min/max incidence threshold can be specified. The number of topics to be generated by the lda modeler (and thus utilized by the document-topic distribution) as well as the number of iterations undertaken by the lda modeler can also be specified within this configuration file.

The feature engineering script can be incorporated within the main wrapper script, or executed in isolation by locating to the preprocessing sub-directory and calling ***python3 hp_dynamic_feature_generation.py*** in a similar way to the preprocessing script.

6.4 Clustering
The operations contained within the clustering script can be summarized as follows:

- Create a master feature data frame by merging the static and dynamic feature sets
- Perform Kmeans clustering using a selected number of features contained within the master feature data frame, appending the cluster allocation as an additional feature within the master data frame
- Segment the master data frame based upon the newly created cluster allocations, generate as many cluster data frames as specified within the user configuration
- Evaluate the composition of the newly segmented data frames
- Export segmented data frames and the resultant analysis

Parameter configuration for the Kmeans cluster model as well as the features to be utilized within this cluster model is contained within ***configs/hp_cluster_config.json***. The number of clusters as well as the number of iterations used to fit the Kmeans clusters can be specified. A Boolean scheme indexing across a selection of the features contained within the master feature data frame can be configured to select which features to utilize during the clustering process.

As always, the clustering script can be incorporated within the main wrapper script, or executed in isolation by locating to the clustering sub-directory and calling ***python3 hp_clustering.py***.

6.5 Classification
The operations contained within the classification script can be summarized as follows:

- Import all cluster-segregated data frames
- For each data frame, generate a set of models specified within the user configuration using unique features for each cluster that are also specified within the user configuration
- For each group of models, evaluate the accuracy of the models via precision, recall, f1, support and cross validated model accuracy
- Concatenate and export all results

Parameter configuration for the models to utilize during the classification process and the feature specification for each cluster is contained within ***configs/hp_classification_config.json***. The number of folds as well as the relative composition of training/testing data utilized during the kfold CV process has been set to values of 10 and 0.6/0.4. Similarly, to the clustering config, a boolean scheme that indexes across all available models can be configured to determine which models are utilized within the final classification for each cluster (global effect). Finally, the features to utilize within each of the clusters can also be specified, configuration for this process does require that a number of cluster configurations commensurate with the number of available/previously defined clusters is provided for use.

As always, the classification script can be incorporated within the main wrapper script, or executed in isolation by locating to the classification sub-directory and calling ***python3 hp_classification.py***.

6.6 Wrapper Script

The operations contained within the wrapper script can be summarized as follows:

- Call various sub-modules in combination with one another based upon the user configuration
- Execute these various configurations for both the honey pot dataset and the 14 million tweet dataset, based upon user configuration

The schema for the wrapper script follows the same Boolean scheme featured within other configuration files, indexing across all of the previously detailed sub-scripts. The wrapper script is purposed with providing a convenient way to call combinations of the subscripts, without having to locate into their containing directories to invoke their functionality.

6.7 Data Processing Scheme

During the pipeline's lifecycle, the honeypot dataset is divided, transformed and consolidated multiple times. Some of the more important intermediate changes (particularly the output of a particular subscript) are written as csv files, allowing the effect of subscripts to be examined or potentially piped/collected for use within other programs. Specifically, snapshots of the data are checkpointed:

- As they are originally contained, that is, as raw data
- After initial preprocessing; the state of the static features (inherited from a previous project) is such that the static features largely bypass the preprocessing phase, for the dynamic features, all tweet for a particular user are collated and grouped
- After dynamic feature generation; the state of the static features doesn't change at this point, however the dynamic feature set undergoes a substantial transformation (detailed within the feature engineering subscript)
- After the preliminary clustering; individual data frames segmented based upon the clustering results are saved

This state pipeline is once again highly reflected via the containing sub-directories. Appropriate directories for raw, preprocessed and final_feature datasets exist containing these various states.