METHODS FOR ANALYZING THE EVOLUTION OF EMAIL SPAM

by

BHARATH KUMAR NACHENAHALLI BHUTHEGOWDA

A THESIS

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Master of Science

September 2018

THESIS APPROVAL PAGE

Student: Bharath Kumar Nachenahalli Bhuthegowda

Title: Methods for Analyzing the Evolution of Email Spam

This thesis has been accepted and approved in partial fulfillment of the requirements for the Master of Science degree in the Department of Computer and Information Science by:

Professor. Daniel Lowd	Chair

and

Janet Woodruff-Borden	Vice Provost and Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded September 2018

THESIS ABSTRACT

Bharath Kumar Nachenahalli Bhuthegowda

Master of Science

Department of Computer and Information Science

September 2018

Title: Methods for Analyzing the Evolution of Email Spam

Email spam has steadily grown and has become a major problem for users, email service providers, and many other organizations. Many adversarial methods have been proposed to combat spam and various studies have been made on the evolution of email spam, by finding evolution patterns and trends based on historical spam data and by incorporating spam filters. In this thesis, we try to understand the evolution of email spam and how we can build better classifiers that will remain effective against adaptive adversaries like spammers. We compare various methods for analyzing the evolution of spam emails by incorporating spam filters along with a spam dataset. We explore the trends based on the weights of the features learned by the classifiers and the accuracies of the classifiers trained and tested in different settings. We also evaluate the effectiveness of the classifier trained in adversarial settings on synthetic data.

CURRICULUM VITAE

NAME OF AUTHOR: Bharath Kumar Nachenahalli Bhuthegowda

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, Oregon, USA
PES Institute of Technology, Bengaluru, Karnataka, India

DEGREES AWARDED:

Master of Science, Computer Science, 2018, University of Oregon
Bachelor of Engineering, Computer Science, 2014, PES Institute of Technology

AREAS OF SPECIAL INTEREST:

Machine Learning

Artificial Intelligence

PROFESSIONAL EXPERIENCE:

Software Engineer, MediaTek Inc, 2014-2016

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Prof. Daniel Lowd for the continuous support of my research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor. I would also like to thank my family and friends for their never-ending support and inspiration.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

Spam can be defined as unsolicited or unrelated contents sent to users in large numbers. They are most commonly associated with emails as email spam which are also called junk mails or unsolicited bulk emails (UBE) or unsolicited commercial emails (UCE) (Wikipedia contributors, 2018a). They are also associated with other domains like instant messaging, social network, blogs, forums, etc. They may be sent with commercial intent or with criminal intent. A large number of spam emails contain disguised links which could lead to phishing and fraudulent websites. They also increase network traffic and consume storage space. Email spam has steadily grown in the past couple of decades and has become a major problem for users, email service providers, and many other organizations.

Many studies have been made on the evolution of spam, by finding evolution patterns and trends based on historical spam data (Wang, Irani, & Pu, 2013) and by incorporating spam filters (Guerra, et al., 2010). As the spammers continually try to evade the spam filters and spam filters upgrade themselves to block the spam, it has become a never-ending race between spammers and the spam filters. Since the nature of spam emails is affected by evolving spam filters, it makes more sense when spam datasets are analyzed along with the filters.

In this thesis, we try to understand the evolution of email spam and how we can build better classifiers that will remain effective against adaptive adversaries. We compare and analyze different methods by incorporating spam filters along with spams to identify and explain trends in email spam. Here we view the spam filter as a binary classifier, which classifies emails into spam and non-spam.

The datasets used in this work are SPAM Archive (Guenter, 2018) and the Enron dataset (Klimt & Yang, 2004). SPAM Archive is an excellent source of spam emails which are collected by Bruce Guenter using honey-pot addresses from early 1998. In this thesis, we use emails collected from 2000 to 2016 as email spams. The Enron dataset was collected and prepared by the CALO Project (A Cognitive Assistant that Learns and Organizes). We have used Enron dataset as non-spam emails. The classifier is trained on these datasets with different settings. We then explore the trends based on the weights of the features learned by the classifier. We also compare the accuracies of the classifiers trained and tested in various settings. Inspired by the adversarial techniques proposed (Dalvi, Domingos, Sanghai, & Verma, 2004), we evaluate the effectiveness of the classifier trained in adversarial settings on synthetic data.

CHAPTER II

BACKGROUND: DEFINITIONS AND TOOLKITS

This chapter briefly covers the various definitions, techniques, and toolkits that are used in this thesis.

2.1 Logistic Regression

Logistic regression is a statistical model widely used for binary classification. It uses a logistic function (a sigmoid function) at the core of the method. The method involves estimating the parameters of the logistic model, which in turn predicts the log-odds of the probability of an event as a function of a linear combination of independent variables also called features. The outcome is often labeled as *'0'* and *'1'* which can also represent *spam/non-spam*, *pass/fail* etc. (Wikipedia contributors, 2018b)

2.2 SciPy

SciPy (Jones, Oliphant, Peterson, & others, 2001) is an open source Python framework for mathematics, science, and engineering. It includes many popular packages like NumPy, Matplotlib, Scikit-Learn, Pandas, Ipython etc. In this thesis, we have utilized some of these packages including Scikit-Learn (Pedregosa, et al., 2011), Matplotlib (Hunter, 2007) and pandas (McKinney, 2010) extensively for its experiments. The Scikit-Learn library contains efficient implementations of various machine learning algorithms and provides high-level APIs to access them (Buitinck, et al., 2013). Matplotlib is used to generate high-quality graphs. Pandas is used for data wrangling.

2.3 Data Preprocessing

One of the important and early steps to be performed during data mining is to prepare the data before feeding it to an algorithm. It is a technique of transforming raw data into a clean dataset. It involves formatting, cleaning, and sampling of data which are explained in detail below.

2.3.1 Data Formatting

The data available can be in various formats like tables in relational databases, HTML files, binary files, JSON files etc. Hence it is required to format the available data into suitable formats to work with.

2.3.2 Data Cleaning

Depending on the dataset to be worked on, data cleaning involves various tasks like removal or fixing of missing data, removal of HTML tags, removal of stop words, anonymizing or removal of certain attributes containing sensitive information.

2.3.3 Data Sampling

Depending upon the availabilities of quality/quantity of data, computational and memory resources, data needs to be sampled. For example, when the available data is larger than needed, or when the available resources cannot handle the huge amount of data, a smaller representative sample of it can be considered. Also, when the data is biased, for example, if the dataset contains a large number of spam emails but few non-spam emails, spam emails can be sampled to match the size of non-spam emails.

2.4 Data Transformation

Depending on the data to be worked on and the algorithm that is used, data should be transformed so that the algorithm can accept valid input and perform well. Some of the common transformations applied to the data are scaling, decomposition, and aggregation. When the features are numerical, some algorithms are sensitive to the range of values different features can take. In such cases, the data must be normalized. Some data represent complex concepts which can be hard to make use of when used directly by the algorithms. These data will be more useful when decomposed into constituent parts. For example, text data in an email can be vectorized using the bag-of-words model and then this vectorized data can be fed to the algorithm. The bag-of-words model is explained later in detail. Sometimes multiple features can be aggregated into a single feature which can be more relevant for an algorithm.

2.5 Data Tokenization

Data tokenization is a technique of breaking up a sequence of characters into small pieces such as words, phrases, sentences or other elements. These small pieces are called tokens. Some of the common heuristics used to break the sequence include breaking on white spaces, punctuation characters or newline characters. During the process, characters like punctuation marks and white spaces are removed.

Many libraries and tools are available to achieve this process. In this thesis, we use the library provided in Python by NLTK (Bird, Klein, & Loper, 2009) for tokenizing.

2.6 Bag-of-Words Model

Bag-of-words model is a representation of text data in vectorized form by considering the text as a bag (multiset) of words. In this model, grammar and word order are ignored keeping only the occurrence of the words. The two common ways of storing the representation in this model are term frequency representation where the frequency of occurrences of each word is stored, and the other way is storing only a Boolean flag indicating whether the word is present or not without considering the number of times it appears in the text.

But this representation could suffer from the lack of quality in the information by the presence of common words like articles and prepositions. The words that are very common and occur very frequently are called stop words and these words need to be removed to improve the quality.

Since the bag-of-words model ignores word orders, that is only the count of words matters, to capture spatial information within the text, n-grams are used. An n-gram is a contiguous sequence of n items like words, letters, syllables from a given sample of text. When n=2, it is called a bigram model. For example, a sentence like “Jack goes to college every day” in bigram can be represented as [“Jack goes”, “goes to”, “to college”, “college every”, “every day”].

2.7 Performance Evaluation Metrics

Following are the metrics that are used for evaluation and comparison in this research.

2.7.1 Confusion matrix

A confusion matrix, also known as the error matrix, is a table used to visualize the performance of the classification model on a set of test data for which the actual values are known. Some of the common terms used to describe confusion matrix are

True Positives (TP): Number of instances that are predicted positives and are actually positives.

False Positives (FP): Number of instances that are predicted positives but are actually negatives.

True Negatives (TN): Number of instances that are predicted negatives and are actually negatives.

False Negatives (FN): Number of instances that are predicted negatives but are actually positives.

A typical confusion matrix in table layout is shown in Table 2.1

		Actual	
		True	False
Predicted	True	True Positives	False Positives
	False	False Negatives	True Negatives

Table 1. Confusion Matrix

2.7.2 Accuracy

Accuracy of a classification model is given by below formula

Accuracy = (TP + TN) / (TP + TN + FP + FN)

2.7.3 False Positive Rates (FPR) and False Negative Rates (FNR)

False positive rate is the proportion of actual negatives that yields positive test outcomes and is given by the formula below

FPR = FP / (FP+TN)

False negative rate is the proportion of actual positives that yields negative test outcomes and is given by the formula below

FNR = FN / (FN+TP)

CHAPTER III

RELATED WORK

With the increase in spam content on the internet, much research has been done and new techniques have been proposed to combat against them. Some of the early efforts in automatic spam detection using machine learning techniques include Bayesian network classifiers (Friedman, Geiger, & Goldszmidt, 1997), naïve Bayesian approach (Sahami, Dumais, Heckerman, & Horvitz, 1998) and others. Many publications like (Dalvi et al., 2004), (Chinavle, Kolari, Oates, & Finin, 2009), (Biggio, Fumera, & Roli, 2009) employing adversarial strategies have been published. There have also been studies made on the evolution of email spam, like (Guerra, et al., 2010) and (D. Wang et al., 2013). The works which are closest to this research are (Dalvi et al., 2004), (D. Wang et al., 2013) and (Guerra, et al., 2010).

(Dalvi et al., 2004) formalizes the problem of adversaries adapting to classifiers to produce false negatives. They have used the naive Bayes classifier and extend its functionality to tackle the adversary. (D. Wang et al., 2013) explained the dynamic nature of spam emails by exploring the statistical ways of analyzing the evolution of email spam. They have used topic modeling algorithms on the contents of emails to investigate topic drift and they have also performed network analysis. Again, (Guerra, et al., 2010) studied the evolution of spam by incorporating spam filters along with the spam dataset. They explored the dynamic nature of spam emails by analyzing the change in rule sets of different releases of SpamAssassin, an open source spam filter, on testing with datasets. They have also explored the ways to cluster the spam messages according to filters' view.

CHAPTER IV

METHODOLOGY

4.1 Data Preparation

Firstly, we show the overview of the dataset used in our research by introducing SPAM Archive and ERON datasets. The SPAM Archive dataset is an excellent source of email spam and is collected by Bruce Guenter since early 1998. The website updates the dataset with monthly releases of new email spam. In this research, we have used the email spam collected from the year 2000 to 2016. Figure 1 shows a plot of number of emails received in each year. We can see that the number of spam emails is less from the year 2000 to 2005. Since the dataset covers spam emails collected for sixteen years, we get more insights on the evolution of email spam over these years.

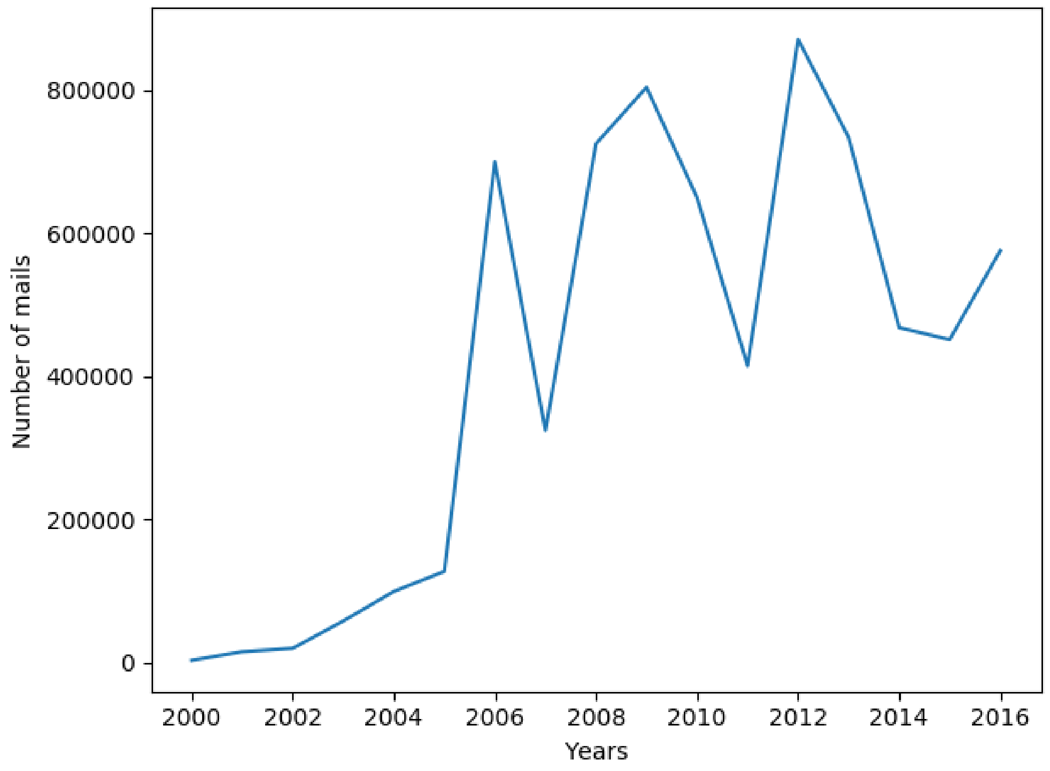

Figure 1. Spam counts in different years collected by SPAM Archive

The Enron dataset is a source for non-spam emails. It was collected and prepared by the CALO Project (A Cognitive Assistant that Learns and Organizes). The dataset contains emails from about 150 users, mostly senior management of Enron. The corpus contains around 19088 emails.

Both the datasets provide raw emails stored in flat files organized into folders. In this research, we consider only the text content of the emails, ignoring the headers and other multipart contents. In addition to the text contents, we add hour of the day and day of the week when the emails were received as additional features. We have used the *email* library provided in Python to parse and extract contents from raw email.

4.2 Data Transformation

Once the contents are extracted from the emails, it is required to convert the text content of the emails into a feature vectors which can be fed to a machine learning algorithm. To achieve this, we use CountVectorizer API provided by Scikit-Learn. In the process of this conversion, we remove stop words in addition to some of the extra words which cause bias while training. These extra words cause bias while training the model and hence are removed. We also ignore the words which appear less frequently in the mail. In the process, we also consider using bigrams as features. So, the vocabulary of the features includes single words and bigrams. To reduce the training time, we store the vocabularies obtained by CountVectorizer.

4.3 Logistic Regression

Once we have feature vectors ready, we use the logistic regression API provided by SciKit-Learn to train the model and test its accuracies. We use default values for the parameters that the logistic regression API offers, although we tried using different solvers and penalties. We consider default values as they gave better accuracies compared to other settings.

Now we describe the different settings that were used to compare the trends.

4.4 Models Trained On A Yearly Basis

In this experiment, the spam dataset is divided into sixteen datasets according to the year they belong. Sixteen new datasets are created by appending the non-spam dataset from Enron to each of the spam datasets. We split these datasets into train and test datasets. Logistic Regression classifiers are created by training on each of the train datasets separately. Each of these classifiers is tested on datasets from other years. Accuracies of each of the models are then plotted. This will give us the insights on how the performance of models trained on one year varies when tested on different years. For example, this shows how the classifier trained on the year 2000 dataset would perform on future emails over the period of time.

4.5 Model Trained On All Years

In this experiment, we try to analyze the performance of the classifier which is trained on email samples from all the years. This model is tested against each of the years and the accuracies are plotted. We can compare the accuracy of this model with a model trained on a single year.

4.6 Feature Comparison

Once the classifiers are trained, we can extract the weights of the features from them. The weight of a feature represents its influence in categorizing the email as spam or non-spam. The features with positive weights contribute for an email to be classified as spam and those with negative weights contribute for the email to be classified as non-spam. We extract the top features contributing towards spam and non-spam from each of the trained models and compare how significant they are in other years. For example, if we extract the top 10 features from the model trained in the year 2000 which contribute towards the spammy nature of an email, we can analyze where they stand and how their weights change over the years. We also try to group similar features to compare the group composition of the top 100 features over the years. This gives us insights on which group of features dominates and how they change over the years.

4.7 Robust Classifier

In this section, we describe how to build a better classifier that will remain effective against adaptive adversaries. This approach is inspired by the adversarial technique proposed by Dalvi et al. We assume that the spammers have complete knowledge of the classifier and they try to evade the filters by adding or removing or replacing some of the features in the emails. In order to capture the effort put by the spammers to evade the filters, these actions are associated with a cost. Finding the right cost for a particular action on a feature could be tricky. For example, the cost of replacing a keyword that sells the spammer's product will be higher than that for the other words. Since it can be subjective, to keep it simple, we assign a random cost varying between [1,3] to each of these actions on different features. Also, we assign a total budget for the spammer to achieve his goal. This makes sure that the spammer will not try to modify the email excessively. In this experiment, we have set the total budget to modify an email to 10. The spammer tries to achieve the goal of evading filter by incurring a minimum loss. Some of the heuristics to achieve this are by adding the features with the least weight to cost ratio or removing the features with the highest weight to cost ratio.

Thus, with this idea, we build a classifier which adapts to the adversary by training itself on the predicted adversarial data. We have the following experiment setup. At first, we sample 15270 emails, which is equal to the number of emails in non-spam dataset, from all years spam dataset. These emails are then combined with non-spam dataset to get a total of 30540 emails and again split into ten different equal sized datasets. Each of these ten datasets has nonoverlapping spam and non-spam emails. The training and adversary actions happen in an iterative fashion over these datasets. Figure 2 gives a visual description of the datasets used in the experiment. Initially, we have the datasets from D1 to D10. Once we train the classifier on D1, we get a classifier C1. The adversary now has complete knowledge of C1. Using that information, the adversary now attacks the dataset D2 to create $D2^1$. We record the accuracy of C1 on $D2^1$. This completes one iteration. Now we have multiple options to update our classifier. We can include either or both of D2 and $D2^1$, inclusive or exclusive of the previous dataset (i.e. D1) as the training dataset for the next iteration from which we get a new classifier C2. This process is carried on in the further iterations. Later we compare the performance of the classifiers C1 to C9. Since we have assumed random values for the costs, we repeat this experiment multiple times with different costs and different datasets and take the average of the performances.

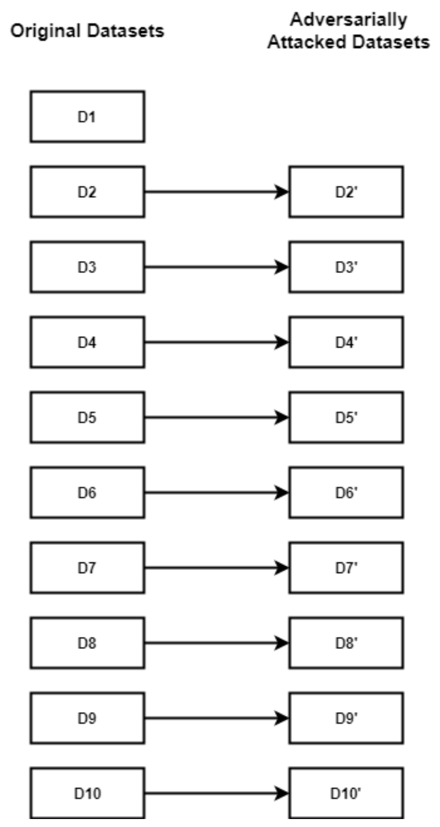

Figure 2. View of datasets used in adversary classification

CHAPTER V

RESULTS AND ANALYSIS

This chapter describes and analyzes the results obtained for the experiments conducted in the previous section.

5.1 Models Trained On A Yearly Basis

As described in the previous chapter, classifiers are trained with datasets from each year and are tested on datasets from other years and performance graphs are plotted. Since the number of graphs generated is large, we describe only a few here, namely the performance of the models that were trained on the years 2000, 2005, 2010, and 2015.

Figure 3 shows the accuracy of the classifier trained on a dataset from the year 2000 over different years. As we can see, the accuracy has dropped rapidly after 2001 and has remained almost constant over further years. The reason for this behavior can be due to fewer data in the training dataset. As we can see from Figure 1 from Chapter 4, the number of spam emails collected in 2000 is much less compared to that of later years. The other explanation could be the drastic change in the features used in spam emails over the years from that of features used in the year 2000.

Figure 4, Figure 5, and Figure 6 show the accuracies of the classifiers trained on datasets from the years 2005, 2010, 2015 respectively over different years. The graphs show similar variation in the accuracies for all these models and all these models have quite high accuracies over all years. This could be due to the presence of common features in all these years and large training datasets.

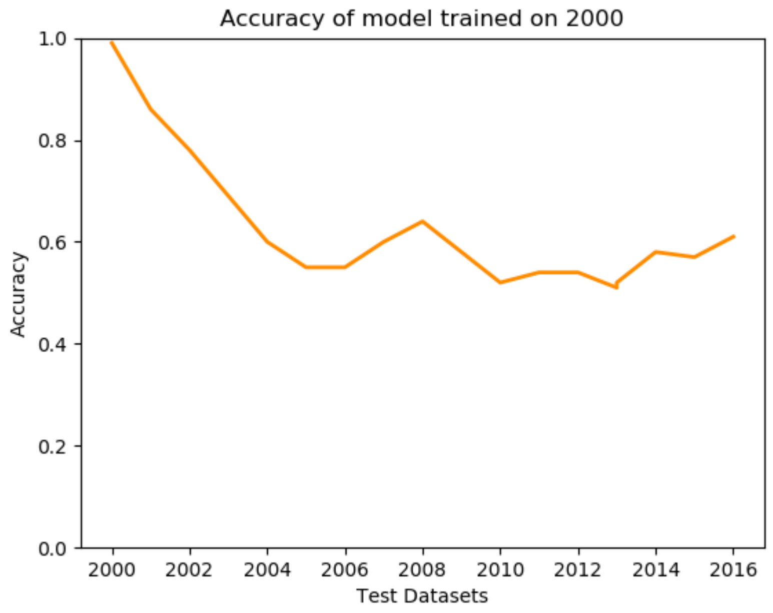

Figure 3. The accuracy of the model trained on a dataset from the year 2000 on different years

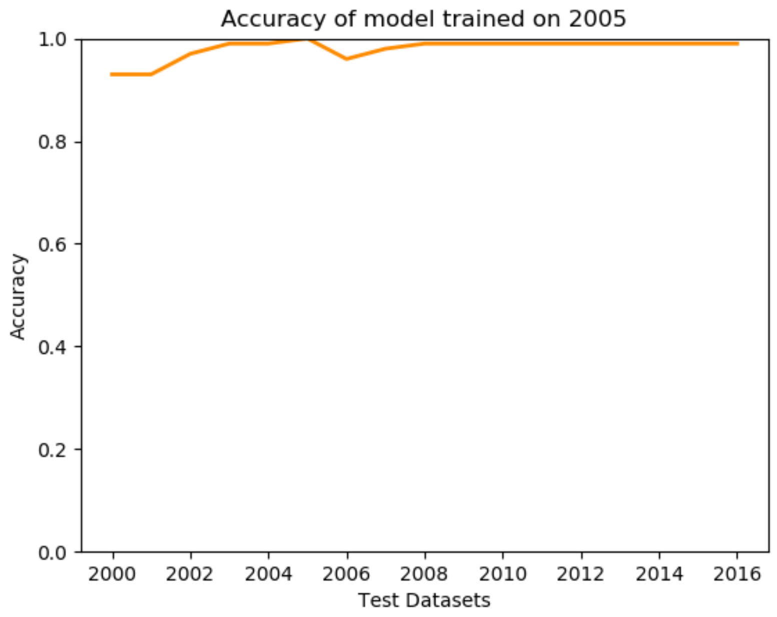

Figure 4. The accuracy of the model trained on a dataset from the year 2005 on different years

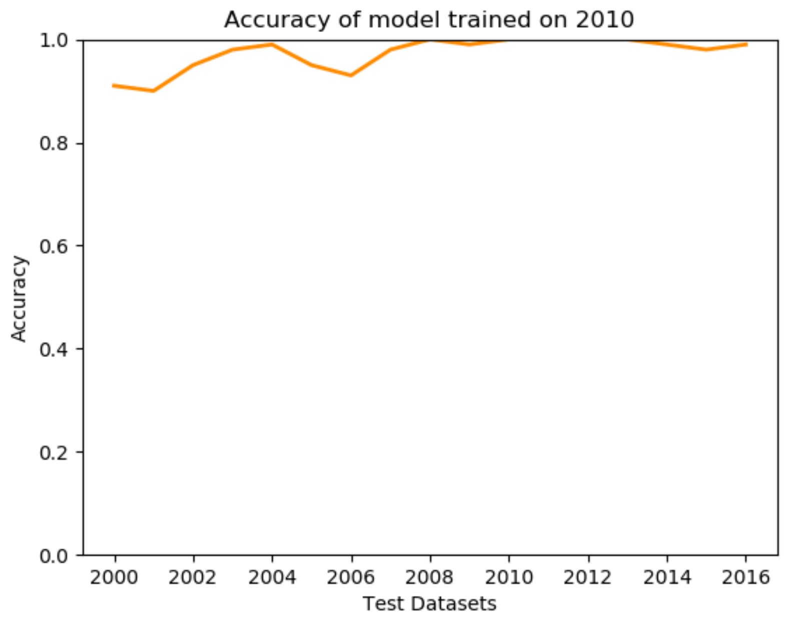

Figure 5. The accuracy of the model trained on a dataset from the year 2010 on different years

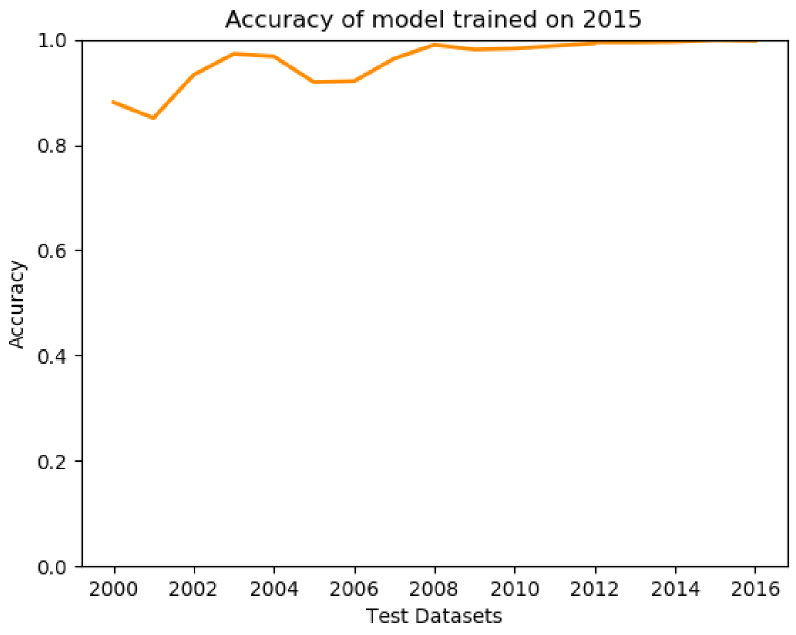

Figure 6. The accuracy of the model trained on a dataset from the year 2015 on different years

5.2 Model Trained On All Years

In this section, we analyze the performance of the model trained on email samples from all the years. Email samples from each year have been taken and tested multiple times. Figure 7 and Figure 8 shows the graphs of the model trained on different runs with different samples. The two figures vary in the ways the emails are sampled from the dataset. In Figure 7, the dataset is created by sampling a minimum of size of the dataset and 30000 emails from each year dataset. In Figure 8, the dataset is created by sampling 25 percent of emails from each year dataset. We can see that models from both Figure 7 and Figure 8 have better performance than the models trained with a dataset of a single year as shown in the previous section. The slight decrease in the accuracy for the years 2000-2002 in Figure 8 compared to that in Figure 7 is due to fewer email samples taken from these years.

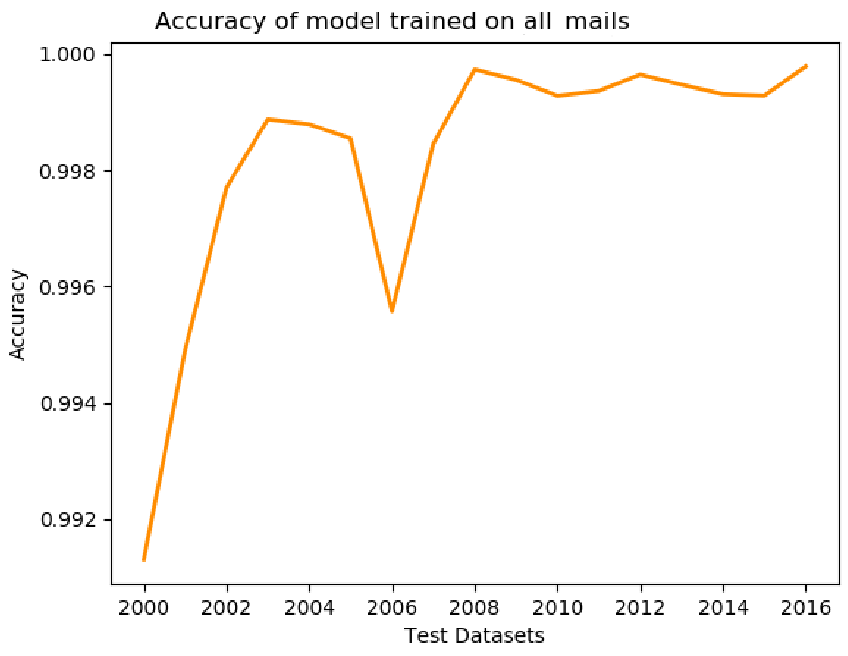

Figure 7. The accuracy of the model trained on a dataset from all years on different years (Trial 1)

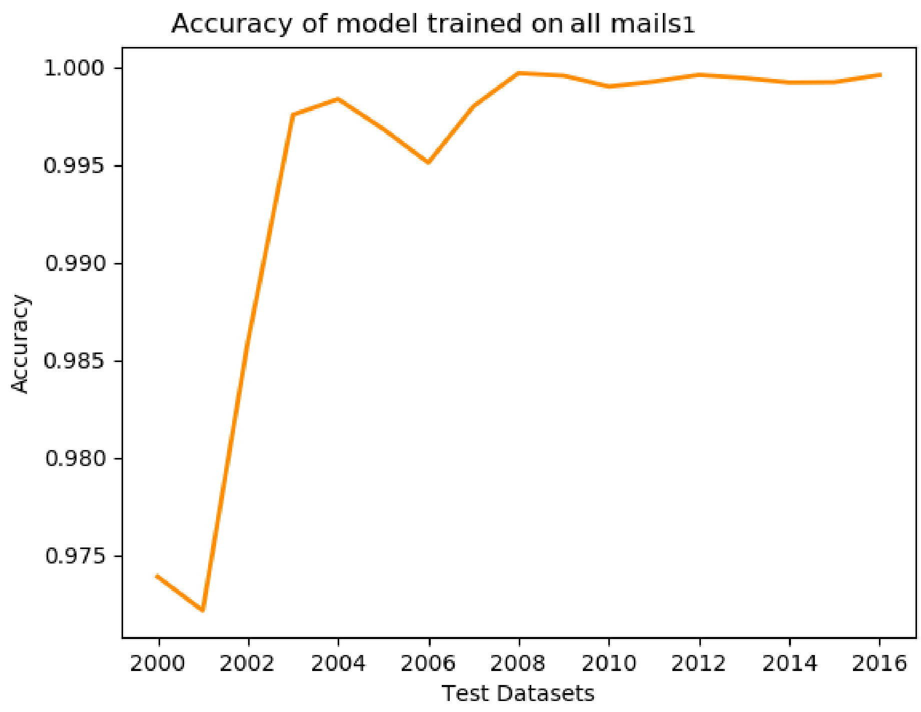

Figure 8. The accuracy of the model trained on a dataset from all years on different years (Trial 2)

5.3 Feature Comparison

Firstly, we show how the weights of the top 10 features contributing towards spam and non-spam nature of emails have changed over the time. By top 10, we mean the features with most negative weights and most positive weights. The features with negative weights contribute towards non-spam nature of the emails while the features with positive weights contribute towards spam nature of the emails. Again, since the number graphs generated is large in number we show some of the interesting ones.

5.3.1 Trends in non-spam features

In this section, we discuss the top 10 non-spam features. As we can see in Figure 9 and Figure 12, the top 10 features include a day of the week and hour of the day. As the non-spam dataset is the same for all the models, the lower weights of these features signify that spam emails were not received or less frequently received during those days of week and hours of the day. Figure 10, Figure 11 and Figure 13 show that features that appear commonly in business communications have lower weight, thus contributing towards non-spam nature of the emails.

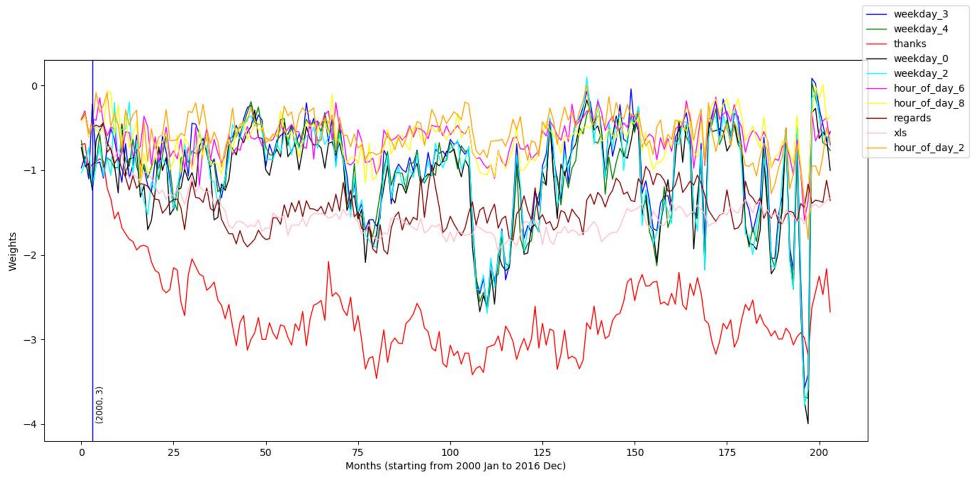

Figure 9. Comparison of weights of the top 10 non-spam features in the month of March 2000.

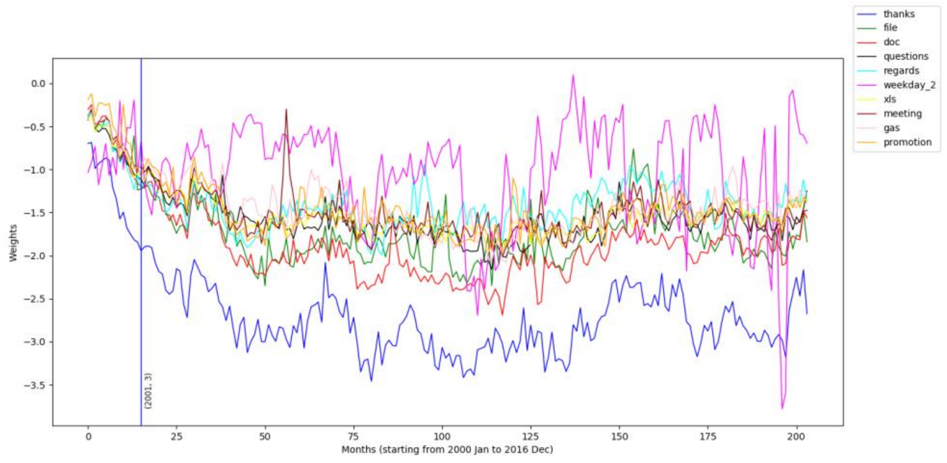

Figure 10. Comparison of weights of the top 10 non-spam features in the month of March 2001.

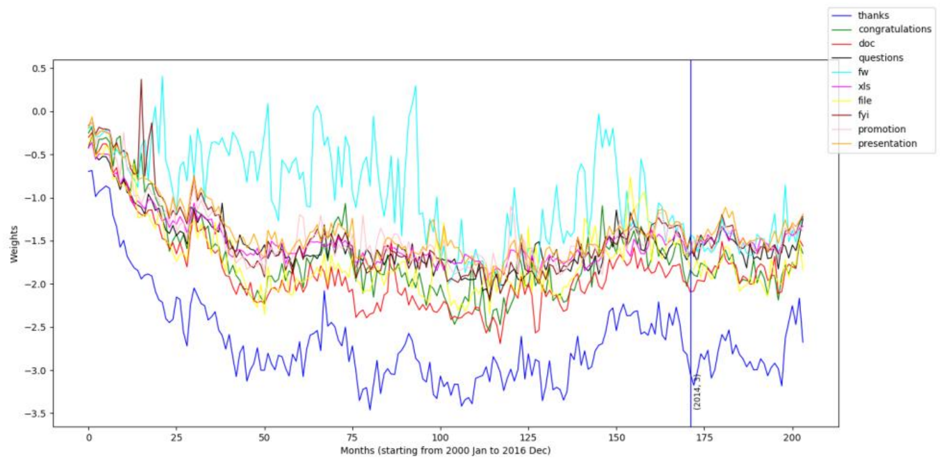

Figure 11. Comparison of weights of the top 10 non-spam features in the month of March 2014.

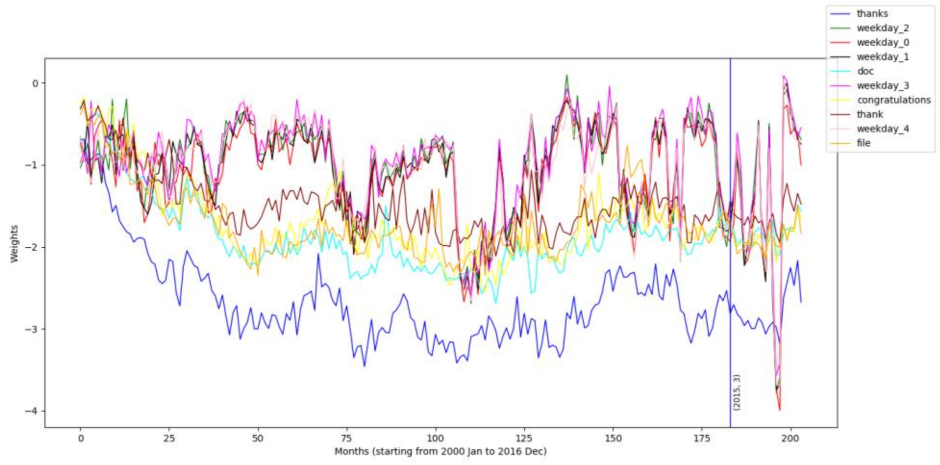

Figure 12. Comparison of weights of the top 10 non-spam features in the month of March 2015.

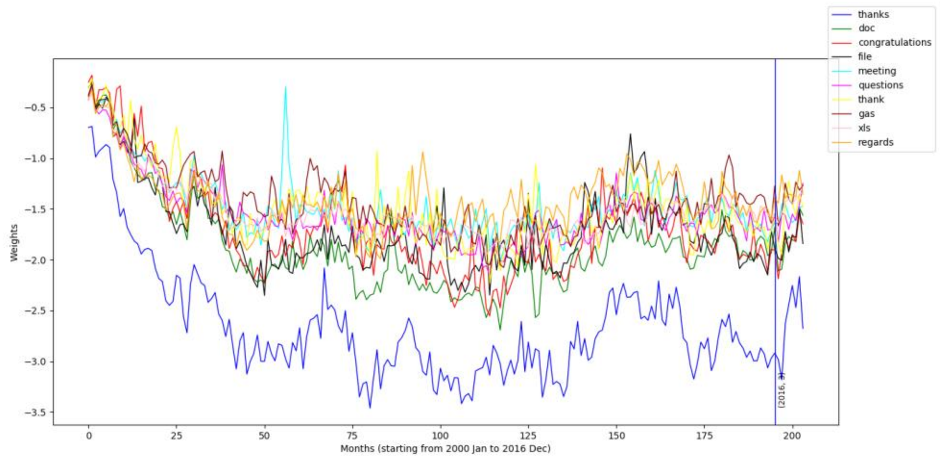

Figure 13. Comparison of weights of the top 10 non-spam features in the month of March 2016.

5.3.2 Trends in spam features

In this section, we discuss the top 10 spam features. As we can see in Figure 14 and Figure 15, the usage of words like ‘Viagra’, ‘free’ and ‘money’ are dominant during the years 2000 and 2001 but they diminish in later years. Figure 16 and Figure 17 shows that features like ‘shift_jis’, ‘ads’, ‘ru’ have higher weights and thus contributing towards spam nature of the emails. ‘Shift jis’ is a character encoding for the Japanese language which indicates that these emails are most probably sent from Japan. Also, the presence of feature ‘ru’ indicates that the emails could contain links or URLs containing Russian domain name. Overall, in the recent years, spammers have tried to evade filters by introducing different character encoding in the spam content.

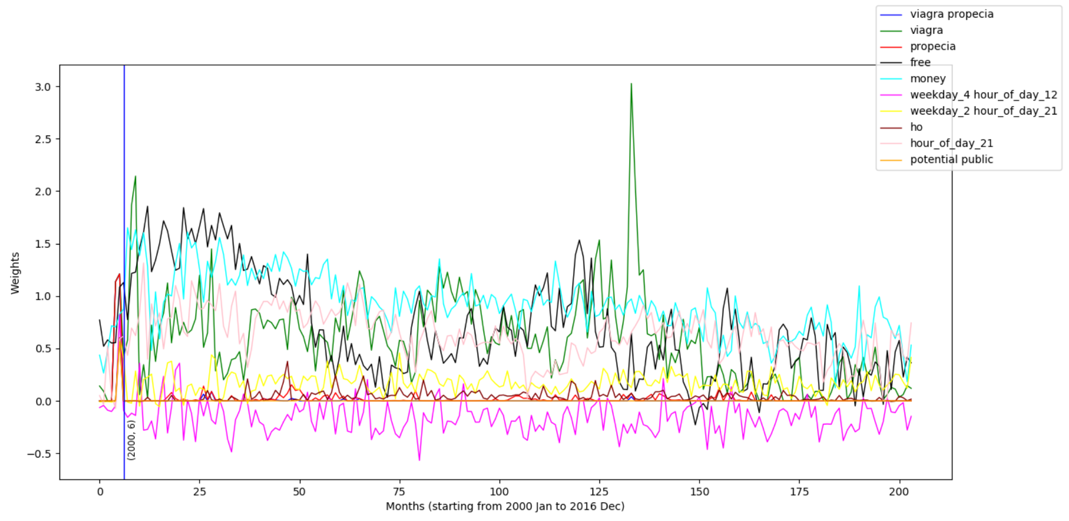

Figure 14. Comparison of weights of the top 10 spam features in the month of June 2000.

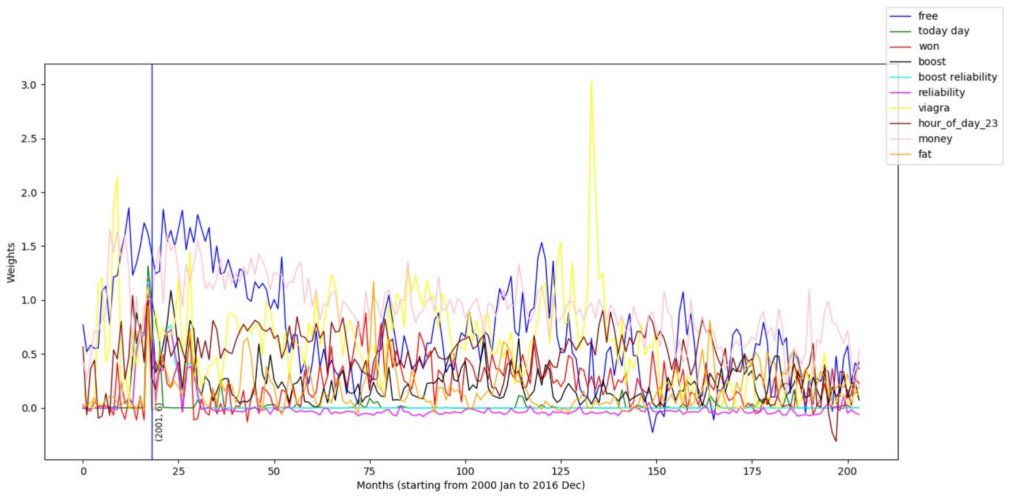

Figure 15. Comparison of weights of the top 10 spam features in the month of June 2001.

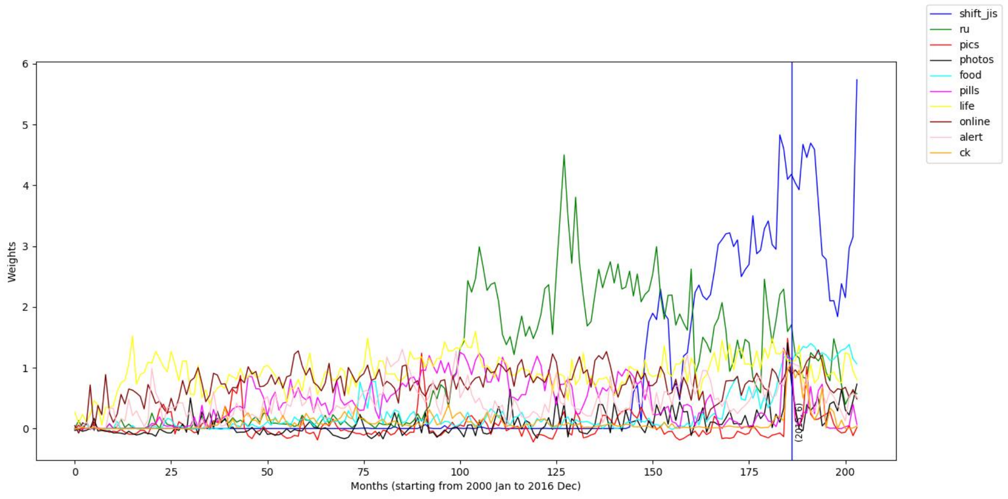

Figure 16. Comparison of weights of the top 10 spam features in the month of June 2015.

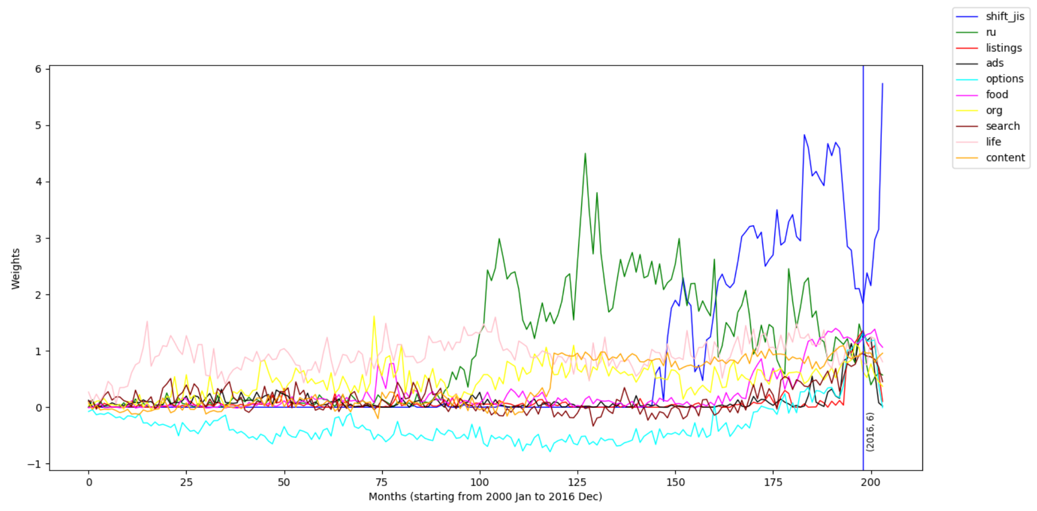

Figure 17. Comparison of weights of the top 10 spam features in the month of June 2016.

5.3.3 Trends from a model trained on all years

In this section, we analyze the trends of the features obtained in the model trained on datasets from all years. Figure 18 shows the comparison of the top 10 spam features. We can see that most of the features are character encodings. This indicates that spammers might have started using encodings as a new way to evade the filters or it could be an artifact of using Enron dataset as non-spam dataset, as the dataset was collected during 2000-2001 which could lack the usage of encodings. Figure 19 shows the comparison of the top 10 non-spam features which includes features that are commonly used in business communications. The presence of features like “dave” and “daren” in the the top-10 non-spam words is probably due to the nature of the Enron dataset, and this would not be the case in most datasets.

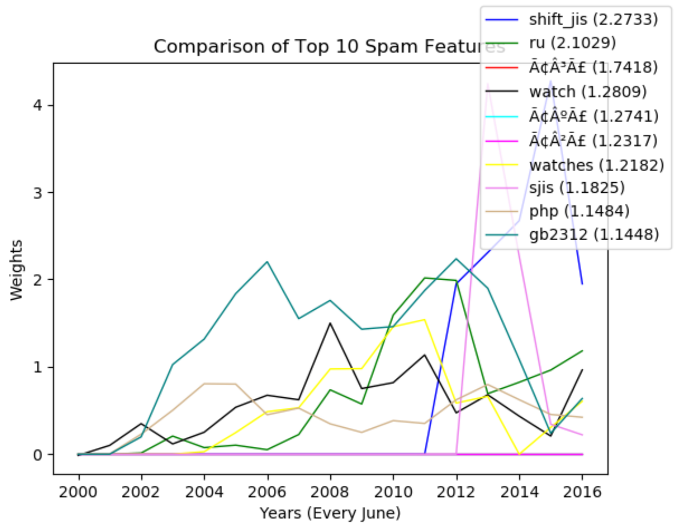

Figure 18. Comparison of weights of the top 10 spam features of all year model.

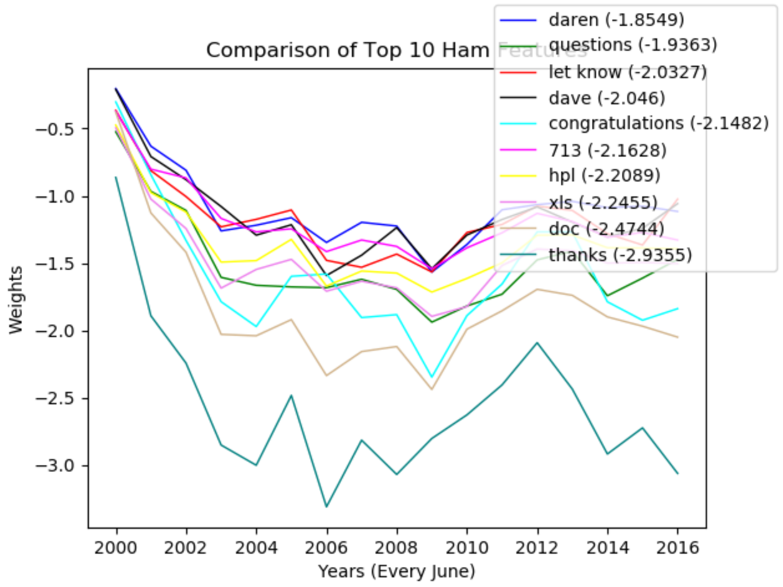

Figure 19. Comparison of weights of the top 10 non-spam features of all year model.

5.3.4 Analysis of group composition

After extracting the top features from the model, we have manually labeled each feature in some of the known categories. These categories include dictionary words, non-dictionary words, medical words, encodings, timestamp, domain names, and vulgarity words.

In the Figure 20 and Figure 21 we can see that during 2000 and 2001, though the number of vulgarity and medical terms in emails were low, they were the top features with high average weights. In the Figure 22 and Figure 23 we can see that during 2015 and 2016, the domain names and encodings were the top features with high average weights.

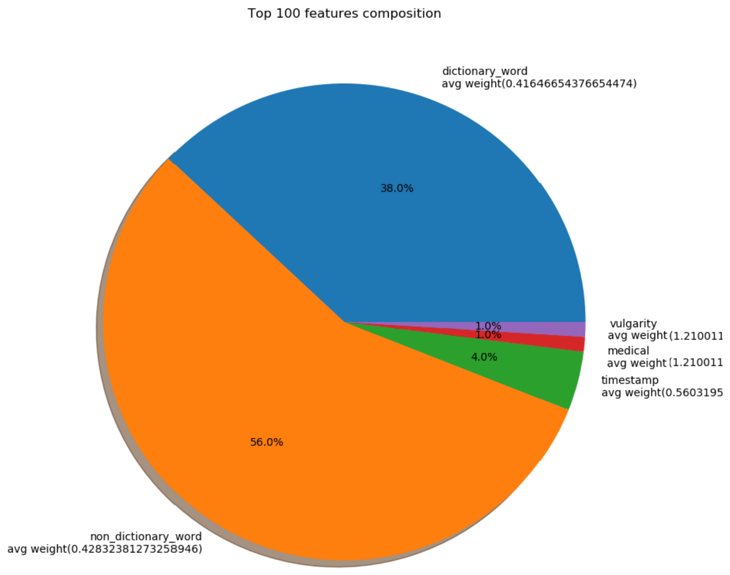

Figure 20. Feature composition of the model trained on a dataset from June 2000.

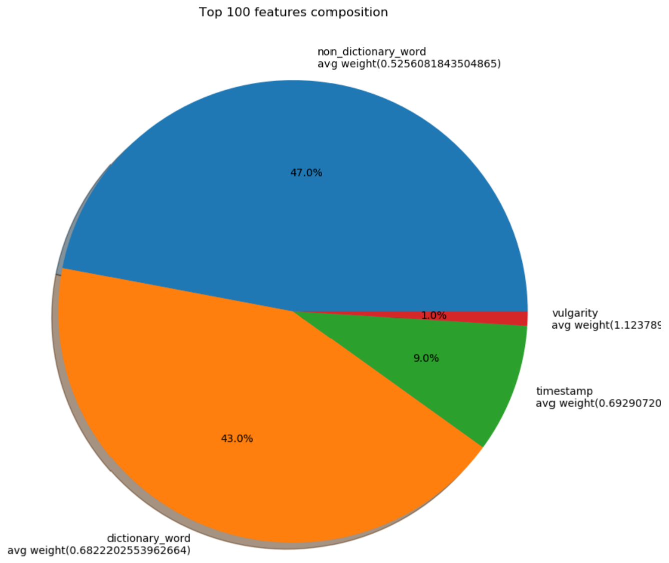

Figure 21. Feature composition of the model trained on a dataset from June 2001.

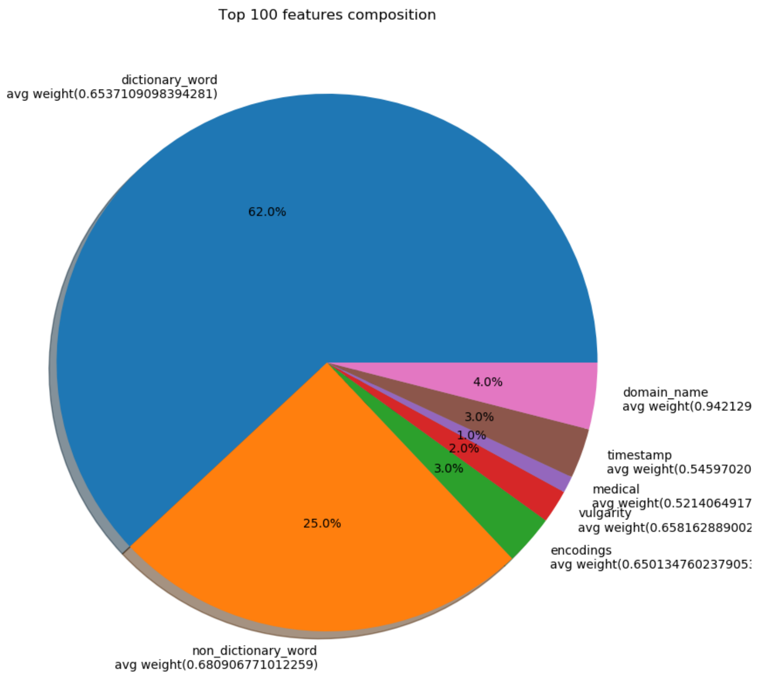

Figure 22. Feature composition of the model trained on a dataset from June 2015.

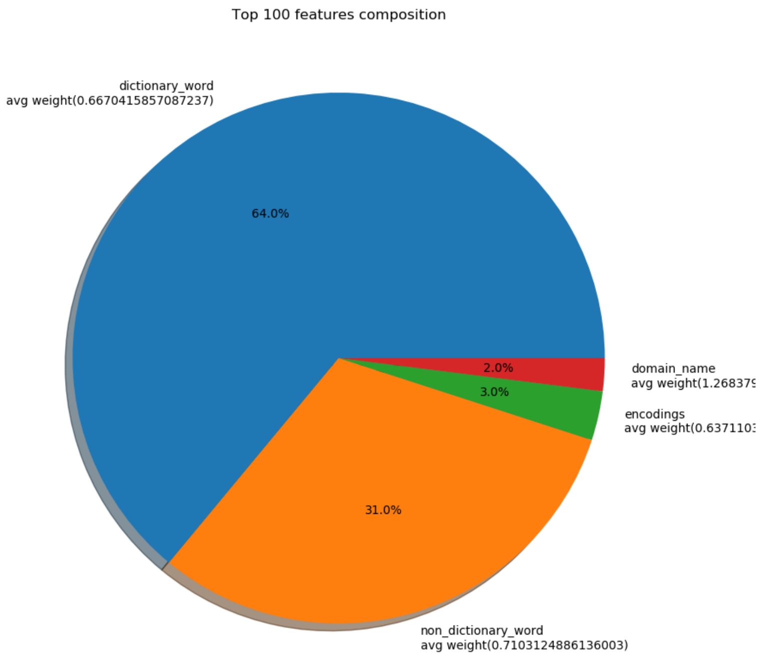

Figure 23. Feature composition of the model trained on a dataset from June 2016.

5.3.5 Performance of model trained in adversarial settings

In this section, we show the performance of the model trained in the adversarial settings. We capture the performance of the model by keeping the false positive rate fixed. We achieve a fixed false positive rate by modifying the decision boundary of the classifier. We do multiple experiments by varying the false positive rate, the total budget available for the spammer, and the cost vector. These experiments are repeated with 500 different datasets and the performance is averaged.

Figure 24, Figure 25, and Figure 26 show the performance of the model when trained with the fixed false positive rate of 0.03 and a total budget of 10, 15 and 20 respectively. In this experiment, the adversarial action is performed by removing the highest weight to cost ratio words. As we can see, the model trained cumulatively on the adversarial datasets (Yellow) performs the best in all the scenarios with the model trained cumulatively on both adversarial and non-adversarial dataset (Black) close to it. The performance of the model trained non-cumulatively on the non-adversarial datasets (Red) is poor and remains constant. The performance of the model trained cumulatively on the non-adversarial datasets (Blue) decreases gradually as it is unaware of the adversarial attacks made by the spammers. The performance of model trained non-cumulatively on the adversarial dataset (Green) depends on the total budget available for the spammer. When the total budget is less (=10), it performs poorly compared to (Green). But the performance gets better when the total budget is increased (=20). The overall performance of all the models decreases with the increase in total budget. Similar trends can be seen when these models are tested by keeping FPR=0.05 as shown in Figure 27, Figure 28 and Figure 29.

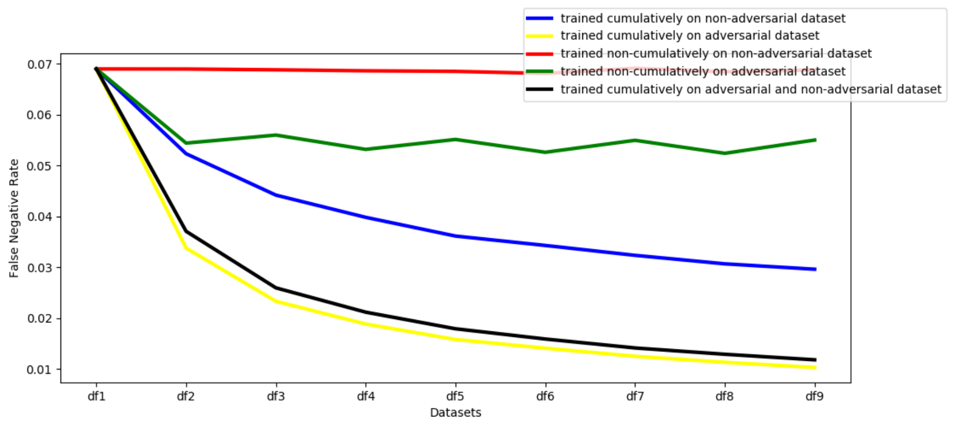

Figure 24. Performance of model with FPR=0.03 and Total Budget=10.

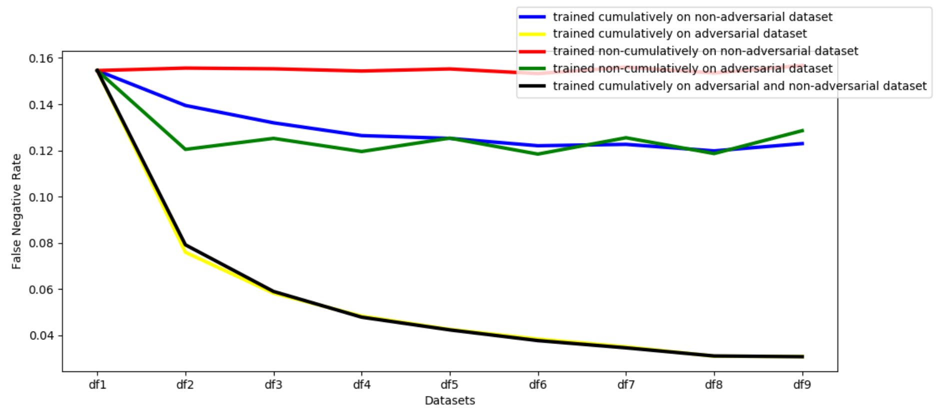

Figure 25. Performance of model with FPR=0.03 and Total Budget=15.

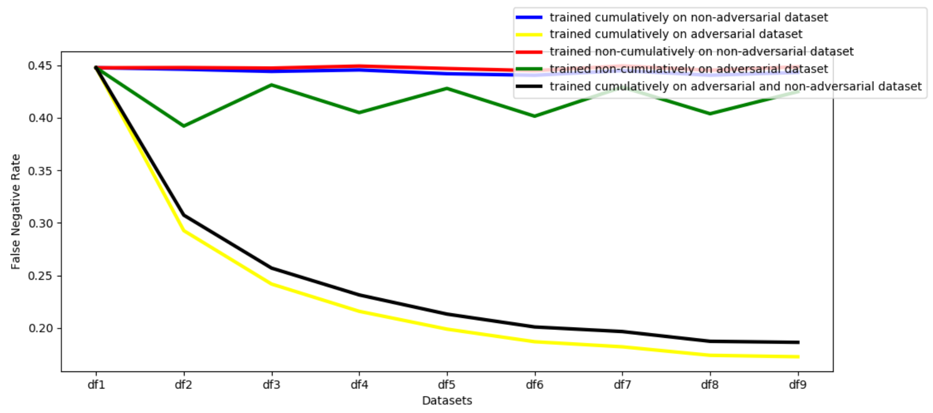

Figure 26. Performance of model with FPR=0.03 and Total Budget=20.

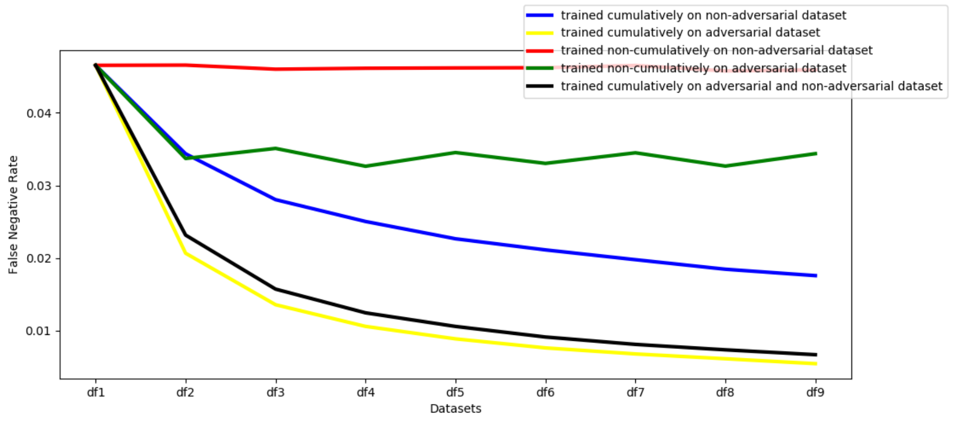

Figure 27. Performance of model with FPR=0.05 and Total Budget=10.

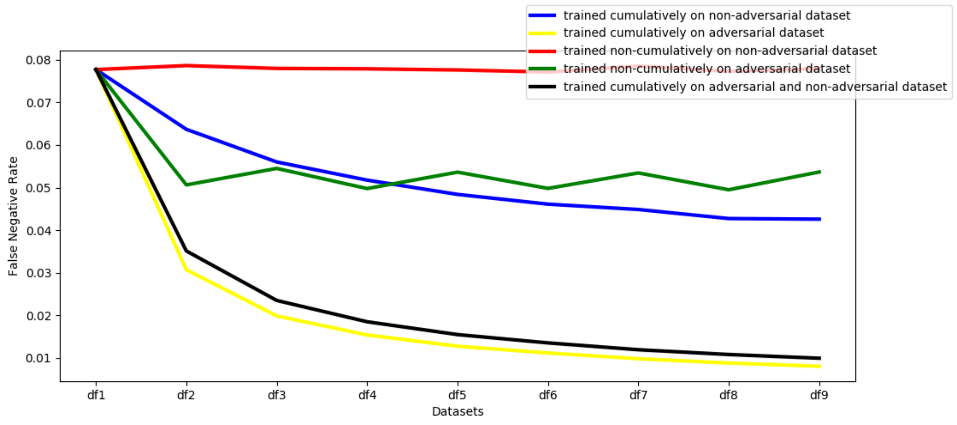

Figure 28. Performance of model with FPR=0.05 and Total Budget=15.

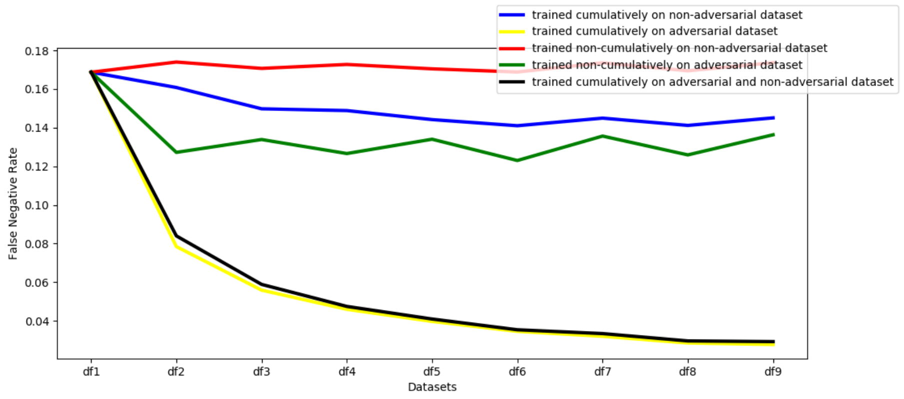

Figure 29. Performance of model with FPR=0.05 and Total Budget=20.

Further, Figure 30 and Figure 31 shows the performance of the models trained with the adversarial addition of features. As we can see all the three models (Blue, Yellow, Black) have similar performances. But the other two models (Red and Green) perform poorly.

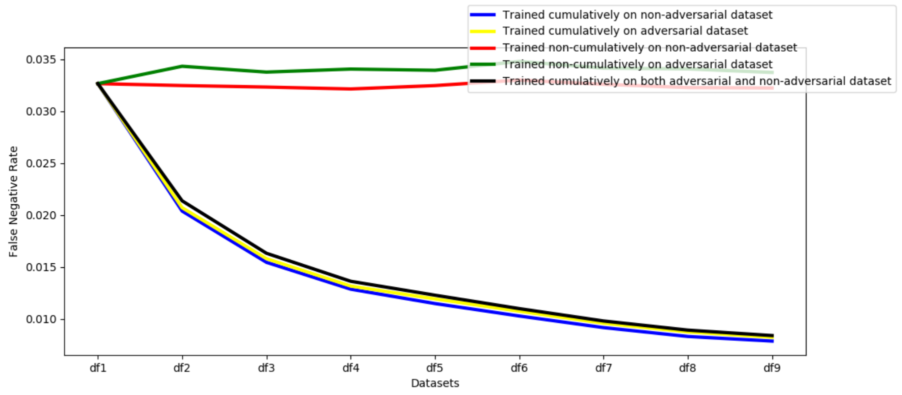

Figure 30. Performance of model (addition) with FPR=0.03 and Total Budget=10.

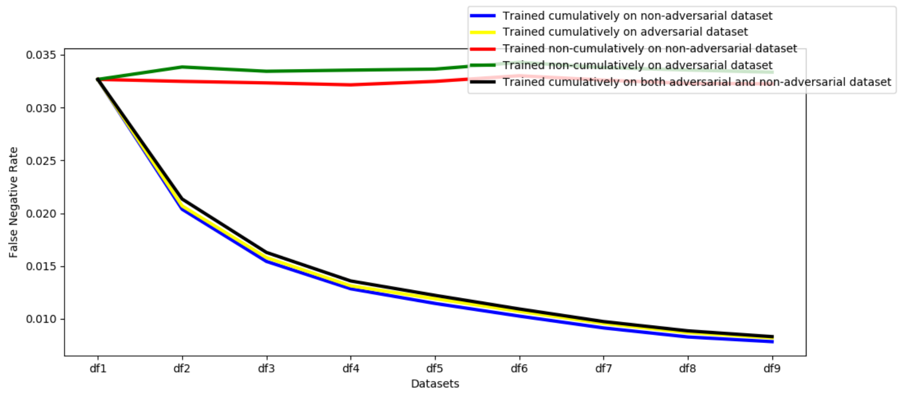

Figure 31. Performance of model (addition) with FPR=0.03 and Total Budget=100.

From the above experiments, we can see that the performance of the models trained on the adversarial datasets is better than those which are trained only on non-adversarial datasets. Although these experiments are tested using random cost vectors on synthetic data, the results are quite impressive. With sophisticated and well-estimated cost vectors, the models can perform better on real data. When the assumption of an adversary having complete knowledge of the classifier does not hold, the performance could be questionable. In that case, it appears that the model trained on both adversarial and non-adversarial datasets could perform well.

CHAPTER VI

CONCLUSION

We have explored various methods to analyze the trends of the features that depict the evolution of email spam. In the beginning, we discussed the performance variations of the model trained on a single year when tested on different years. We saw that the performance of the model trained on the dataset for the year 2000 decreased steeply which indicates a significant change in spam features over the years. Performances of the models trained on later years were quite high over a wide range of years which indicates the presence of common features. We also compared the accuracies of the model trained on all years which had better performance than any of the former models.

Further, we discussed how the weights of the features can be used to find the trends in the features over time. We saw that non-spam features in most of the models were the words used in usual business communications. We then observed a shift in the weights of the spam features over the time. The features which were significant in the early years were gradually replaced by new words. We also categorized the top features to see the categorical drift in the features over time. We also saw that vulgar and medical terms were common features in the early years, while the contents with new encodings were very common in the later years. This indicates spammers are trying to evade filters with new encodings.

Finally, we built a better classifier that will remain effective against the adaptive adversaries. The model assumes that the adversary has complete knowledge of the classifier and the adversary can attack the classifier by the addition or removal of features. When the model is trained in such adversarial settings, the performance is improved.

REFERENCES CITED

Biggio, B., Fumera, G., & Roli, F. (2009). Evade hard multiple classifier systems. *Applications of Supervised and Unsupervised Ensemble Methods*, 15-38.

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python.* O'Reilly Media Inc.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., . . . Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn. *CoRR*, *abs/1309.0238*, pp. 108--122. Retrieved from http://arxiv.org/abs/1309.0238

Chinavle, D., Kolari, P., Oates, T., & Finin, T. (2009). Ensembles in adversarial classification for spam. *Proceedings of the 18th ACM conference on Information and knowledge management*, (pp. 2015-2018).

Dalvi, N., Domingos, P., Sanghai, S., & Verma, D. (2004). Adversarial classification. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 99-108). ACM.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning, 29*, 131-163.

Guenter, B. (2018). *SPAM Archive*. Retrieved from untroubled.org: http://untroubled.org/spam/

Guerra, P. H., Guedes, D., Wagner Meira, J., Hoepers, C., Chaves, M., & Steding-Jessen, K. (2010). Exploring the spam arms race to characterize spam evolution. *Proceedings of the 7th Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS), Redmond, WA.*

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science Engineering, 9*, 90-95.

Jones, E., Oliphant, T., Peterson, P., & others. (2001). SciPy: Open Source Scientific Tools for Python. Retrieved from http://www.scipy.org/

Klimt, B., & Yang, Y. (2004). The enron corpus: A new dataset for email classification research. *European Conference on Machine Learning* (pp. 217-226). Springer.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. v. Millman (Ed.), *Proceedings of the 9th Python in Science Conference*, (pp. 51 - 56).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825--2830.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A Bayesian approach to filtering junk e-mail. *Learning for Text Categorization: Papers from the 1998 workshop*, *62,* pp. 98-105.

Wang, D., Irani, D., & Pu, C. (2013). A study on evolution of email spam over fifteen years. *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on* (pp. 1-10). IEEE.

Wikipedia contributors. (2018a, August 19). Spamming. Wikipedia, The Free Encyclopedia. Retrieved August 29, 2018, from https://en.wikipedia.org/w/index.php?title=Spamming&oldid=855654714

Wikipedia contributors. (2018b, August 19). *Logistic regression*. Retrieved August 25, 2018, from Wikipedia, The Free Encyclopedia: https://en.wikipedia.org/w/index.php?title=Logistic_regression&oldid=85562883
5