Passive Attacks on Statistical Spam Filters - Mid Term Project Report

Nandini Krupa Krishnamurthy
Clemson University
nandink@clemson.edu

I. Objective

This project is being conducted as a part of the course work for CPSC 8180 - Introduction to Artificial Intelligence. The main aim of this project is to implement the paper – Good word attacks on statistical spam filter [1]. This project, though based on the paper, mainly concentrates on implementing passive attacks and has some additives to it. Along with Naïve Bayes and Maxent filters, It also includes Support Vector Machine as a comparative study. The passive attack study is also included. The project along with dataset, methodology, approaches, and the result are explained in detail in this paper.

II. Introduction

The Internet is truly a bad place. There is no check for how it is being misused. As much as it is being used for constructive purposes, it is used for destructive works too in the same amount. One of them which goes unnoticed is the number of spam emails that are being generated every day. What's more, in many cases these go unnoticed. In other words, there go undetected by the spam filters. This is the exact reason behind the study in this paper. As the efficiency of spam filters grows, so does the efficiency of the spammer spamming the mails. Spam filters usually look at the words and their weights in the mail determined by their frequency and decide if the mail is a spam or a legitimate one. In many cases, the filters fail in their duty and the spam mail goes unnoticed usually due to something called "Good words", which makes it easier for the attacker to spam the mailbox. The technique for this attack is simple. An attacker already has a list of words that the spam filter considers as legitimate. So he simply inserts some of these words inside a mail which has spam content and the mail, though actual spam, is considered as a legitimate mail. This is called "Good Word Attack". These can be broadly classified into passive attacks and active attacks.

. This paper mainly concentrates on the passive attacks. These passive attacks can be broadly classified under three types – dictionary attacks, frequency word attacks, frequency ratio attacks. A dictionary attack is when an attacker chooses random words from a larger set of words such that they are more closely associated with legitimate mail and inserts them into the spam mail to make the spam mail look non-spam. Frequency word attack is a step above dictionary attacks. These use words that mostly appear normally in legitimate mails and uses them to start an attack. Frequency ratio attack considers a ratio of words that appear in legitimate mails and words that mostly appear in spam mails and uses this ratio to determine if the mail is spam or not. The next sections talk about the dataset and methodology implemented in this project. For the whole of project, code is written by self.

III. Dataset Description

This project uses the dataset from Ling-Spam Public corpus [2]. This is used in the evaluation of all the classifiers – NaiveBayes, Maxent and Support Vector Machine. It consists of 2893 emails in total. Out of these 481 are spam messages and 2412 are non-spam messages. Spam messages are shows by the files starting with "spm". This corpus consists of 4 folders in total containing bare messages without any pre-processing done, lemmatized messages, messages with stop-words removed and messages which are lemmatized and also with stop words removed. For this project, messages from the "bare" folder are used and pre-processing is dealt in a separate way.

The main disadvantage of this dataset is that the dataset may seem very small with spam making only 16.6% of the whole corpus, meaning, we don't have an equal division of spam and legitimate emails. But considering it is widespread availability and ease of use, this project used this dataset for analysis.

This project also contains another dataset from reuters [3]. This is mainly used to test the effectiveness of an attack.

A. Dataset Pre-processing

Preprocessing of the dataset was done in 3 different ways to check the accuracy of the 3 filters based on various types of words. The processes which were used are :

1) **Lower-casing:** The entire email is converted to lowercase so that all the characters are treated as equal. For example, "Congratulations" is treated as "congratulations".
2) **HTML tags removal:** HTML tags, if present, are removed so that only content remains.
3) **URLs removal:** URLs, if present, are replaced with "httpaddr".
4) **Email addresses removal:** All the email addresses are replaced with "emailaddr".
5) **Numbers normalization:** All the digits are replaced with text "numbers".
6) **Dollar sign normalization:** All dollar signs ("$") are replaced with the text "dollar".

7) **Extra character removal:** The data is searched for non-alphabetic characters such as tabs, extra spaces and commas and are replaced with a single space
8) **Stop-words removal:** NLTK provides a list of English stop words such as “an”, “a”, “etc”. These are removed.
9) **Word stemming:** Words are reduced to their stem form. For example, “want”, “wanted” and “wanting” are all replaced with “want”.
10) **Word Lemmatization:** Words are reduced to their lemmas. Words stemming and word lemmatization may seem to be similar concepts but there is difference. Stemming gives the pseudo-stem, meaning, it tries to take off the suffixes or prefixes and gives the actual word, but lemmatization gives the proper root of the word or the base dictionary of the word. For example, if there is a word such as “better”, stemming gives “better”, whereas lemmatization gives “good”. Similarly for a word such as “meeting”, stemming may give “meet”, but lemmatization looks up for the context i.e., if it’s a verb or a noun and gives the resultant word accordingly.

B. Dataset Variation

Depending on the processes chosen, there are three sets of datasets for this project. For all the 3 variations, effects of word addition/deletion on the filters are studied.

1) Preprocessed data without lemmatization and removal of stop words (“data_clean”). This process is represented by the python file “data_clean.py”.
2) Preprocessed data with lemmatization (“data_clean_lemmatize”). The code is in “data_clean_lemmatize.py”
3) Preprocessed data with all the processes in section 2.1 applied (“data_clean_stop_words”). The file “data_clean_stop_words.py” contains the code for this process.

C. Dataset Variation

As mentioned before, the dataset is inconsistent in the sense that there is no equal division of spam and legitimate emails, spam emails consist only 16.6% of the who corpus. Therefore, efforts are made to keep data consistent with an ideal train and test ratio of 8:2 respectively. Therefore the dataset used in this project has training and test set in the ratio of 864:108, i.e., we have 768 mails in training set with an equal number of spam and non-spam emails (384) and similarly, we have 192 mails in test dataset with an equal number of spam and non-spam emails (96). The dataset ratio is same for all the three dataset variations mentioned in section 2.2. The experiment is conducted on all three datasets, meaning, all three filters are tested with three types of dataset for all three types of passive attacks. So it means, in the end we will have a comparative study of 27 results.

IV. Methodology

As mentioned before, since this project concentrates on passive attacks, all three types of prepared datasets are used to evaluate efficiency on all three types of passive attacks. Every dataset has three python codes, each having three filters. These python codes are named after the type of attack it is evaluating.

1) create_dictionary.py: Used to create a dictionary from the reuters dataset [3]. To keep the consistency common, only the first 3000 words ranked on the basis of frequency are considered from the dictionary
2) train_test_data_clean_Dictionary.py : This is used to test for dictionary attack.
3) train_test_data_clean_Frequency.py: This is used to test frequency attack.
4) train_test_data_clean_Frequency_Ratio.py: This is used to test frequency ratio attack.

A. Libraries Used

1) sys
2) json
3) os
4) re
5) collections
6) nltk
7) matplotlib
8) numpy
9) sklearn

B. Steps Followed

1) Read train data: Since we have three types of training data, they are shown in separate folders as mentioned above. Data is read to train the model.

2) Read-dictionary created using $create_dictionary.py$: The dictionary generated from create_dictionary.py for reuters dataset is read in this method

3) Generate features: Feature generation is important as its result is needed as inputs to the filters we are using.

4) Generate labels: Generating labels is also a part of the generating feature process. Here every message with spam content is labeled as 1 and that without spam content is labeled as 0.

5) Train the machine learning model: Now that we have everything to train a model – feature matrix and labels from the train data – the next step is to train the classifiers. These algorithms we are considering – Naïve Bayes, Maxent and Support Vector Machine – are used to classify the data into classes. Now that training is done, we will have to test the model for the test set.

6) Test the machine learning model: This is the final step. Here we test the model for how well it classifies an unknown input data. We have a dataset prepared for it as mentioned above.

7) Generate/Plot results: The results are generated and plotted for easier understanding.

C. Differences in Approaches

1) Dictionary attack: For the dictionary attack, the weights from the dictionary are averaged so that the random

principle of dictionary attack holds true. This is shown in the text file "dictionary_averaged.txt"

2) Frequency attack: To just measure the frequency of the most common type of attack, the first 1000 words ranked on the basis of weights are considered. This is shown in the text file "dictionary_averaged_frequency.txt".
3) Frequency Ratio Attack: Here the weights of the train data set are measured. The assumption is, it is easier to cause an attack if the words are from the legitimate mails rather than the spam mails. Note that the algorithm, if nothing is specified, used this method to predict the output class.

V. Results

Since there is a large number of images related to results, all of them are zipped into the folder and only the summary of results is presented here. The table below talks about the AUC scores obtained from all the tests.

Classifier	**AUC Score**
data_clean	
Dictionary Attack, NB	96.88
Dictionary Attack, Max	93.75
Dictionary Attack, SVC	91.15
Frequency Attack, NB	94.27
Frequency Attack, Max	90.10
Frequency Attack, SVC	89.58
Frequency Ratio Attack, NB	98.96
Frequency Ratio Attack, Max	96.88
Frequency Ratio Attack, SVC	95.83
data_clean_lemmatize	
Dictionary Attack, NB	96.88
Dictionary Attack, Max	93.75
Dictionary Attack, SVC	90.62
Frequency Attack, NB	94.27
Frequency Attack, Max	91.67
Frequency Attack, SVC	90.10
Frequency Ratio Attack, NB	98.96
Frequency Ratio Attack, Max	96.88
Frequency Ratio Attack, SVC	95.83
data_clean_lemmatize	
Dictionary Attack, NB	96.88
Dictionary Attack, Max	93.75
Dictionary Attack, SVC	93.75
Frequency Attack, NB	93.75
Frequency Attack, Max	92.71
Frequency Attack, SVC	90.62
Frequency Ratio Attack, NB	98.44
Frequency Ratio Attack, Max	96.35
Frequency Ratio Attack, SVC	96.35

From the results, we see a great variation in the working of classifiers. In all the datasets we see that highest accuracy is achieved by Naive Bayes classifier when the stop words are added. This can be see in all the categories. Plus we also see that even in the sub-categories, Naive-Bayes performs the best.

The second observation is that as the words get added, as seen in random dictionary and frequency attacks, the accuracy of detecting spam by the classifiers decrease which uphold our hypothesis that in reality, good words do affect the ability of the classifiers to detect spam.

VI. drawbacks

1) This projects only takes into account passive attacks. There are also active attacks which might be more difficult to detect.
2) Dataset size taken is small. Bigger datasets with better training may produce better results.

VII. Scope for Improvement

In the final submission, all the drawbacks are subject to be fulfilled.

1) Active attacks have to be implemented as a add-on.
2) Methods to block spams instead of combating them have to be studied.

References

[1] Lowd, Daniel, and Christopher Meek. "Good Word Attacks on Statistical Spam Filters." CEAS (2005).
[2] See http://www.aueb.gr/users/ion/ publications.html
[3] The Reuters-21578 text categorization test collection, Distribution 1.0, is freely available for research use at http://www.daviddlewis.com/resources/testcollections/-reuters21578.