Good Word Attacks on Statistical Spam Filters - Final 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 8100 - Introduction to Artificial Intelligence. The main aim of this project is to implement the paper – Good word attacks on statistical spam filter [1]. Part of the project – Passive attacks were implemented for the mid-term submission. The final report concentrates on implementing one type of active attacks – First-N attack. First-N attacks were tested against three common types of spam filters - Naïve Bayes filter, Maxent filters and Support Vector Machine. The active attack study is also included. The project along with dataset, methodology, approaches, and the result is explained in detail in this paper. Along with the final term work, some parts of mid-term submission and the results from the mid-term are also included just to give a background and to obtain continuity to this work. The report begins with explaining the 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.

In the mid-term, passive attacks were implemented. 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.

For the final-term report, active attacks are considered. A special type of active attack called First-N attack is implemented on three of the classifiers – Naïve Bayes, Maxent and SVM and results are compared and analyzed. A detailed explanation is provided in the upcoming sections.

III. Dataset Description

Unlike the mid-term submission, the dataset considered here is a pair of spam and non-spam emails. The spam mail has 1463 words and non-spam(legitimate) mail has 329 words. The two mails were randomly chosen.

A. Dataset Pre-processing

The processes carried out for dataset preprocessing are described below:

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.

IV. Methodology

The final part of the submission is based on the result given by the midterm submission (shown in Table 1 for reference). It was concluded in the submission that we see a great variation in the working of classifiers. In all the datasets we see that the highest accuracy is achieved by Naive Bayes classifier when the stop words are added. This can be seen in all the categories. Plus we also see that even in the sub-categories, Naive-Bayes performs the best. The second observation was that as the words get added, as seen in the random dictionary and frequency attacks, the accuracy of detecting spam by the classifiers decrease which upholds our hypothesis that in reality, good words do affect the ability of the classifiers to detect spam.

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

TABLE I
RESULTS OF THE MID-TERM SUBMISSION

A. Active Attacks

After seeing the performance of each of the classifiers for passive attacks, it was necessary to understand how these classifiers perform in cases of active attacks. So a method was devised to perform an active attack. This is called First-N attack. This process begins by calling a function called FindWitness. FindWitness works on the dataset we use - a pair of spam and legitimate emails. It follows a process wherein for each word from the legitimate mail is checked to see if that occurs in spam mail. If it is present in the spam mail, it is ignored, else it is deleted. Similarly every word in spam mail is checked to see if it is present in legitimate mail if it is present, then it is ignored and if it is not present, then it is added to the legitimate mail. What happens in this process is that at each step the legitimate email is dram closer to being a spam mail. The algorithm stops when the legitimate email transits to spam email by the addition of the last word. These mails are called as barely-legitimate mails in this project.

This process is carried out on all the three classifiers we are considering in this project – Naïve Bayes, Maxent and SVM. And at every step, the results are recorded.

The next step is to add every word from dictionary to the barely legitimate email (the one which is spam now from the previous step on the addition of words and differs from a legitimate email just by a single word) and then this mail is passed on to the classifier to classify as legitimate or spam. It is interesting to know what happens when each word from the dictionary is added to the nearly-legitimate mail. On the addition of some words, the classifier classifies the mail as spam and others as non-spam. These words, in addition to which the classifier considers the barely legitimate mails as non-spam mails called good words.

So, in other words, when "good words" are added to a spam mail which is at the boundary of being of spam or a son-spam, it can become a legitimate mail and pass through the classifiers. Classifiers cannot detect such mails. These so-called "good words" hence become the backbone of the active attacks.

Therefore, our next step is to identify these good words from our dictionary created in the previous steps. This is done by the next algorithm which adds each word from the dictionary to see if that makes the barely-legitimate mail as a non-spam mail and passes it through the filter. On every iteration, a new word from the dictionary is added to the barely-legitimate mail and this mail is passed through the filter. The filter evaluates it and gives the result. If it gives a value of "1", then the good word considered failed to make the mail non-spam, else if the result is "0", the mail has now become spam. This whole process is tested on all the filters and results are shown below.

B. Libraries Used

1) sys
2) json

3) os
4) re
5) collections
6) nltk
7) matplotlib
8) numpy
9) sklearn

C. Files Description

The codes used for this project and the result from this project are stored in the folder AI_FinalProject.

1) **find_witness_first.py**
   This file contains an algorithm which does the following.
   a) Finds the barely legitimate email.
   b) Finds the number of words added to the legitimate email to convert it into barely-legitimate.
   c) Calculates the time taken to get a spam mail pass through a filter.
   d) The result of this file is stored in added_findwitness_classifier-name.txt
2) **first.py**
   a) Finds the number of good words from a dictionary which when added can turn the bare-legitimate email to non-spam email.
   b) the result of this file is stored in good-words_classifier-name.txt

V. Results

The results of the project are summarized in Table 2 and Figure 1.

Table 2 - Results of First-N Attack			
	NB	Maxent	SVM
trials	158	155	200
words	102	101	109
goodwords	2658	2683	2912
time (s)	562.204	547.109	689.982

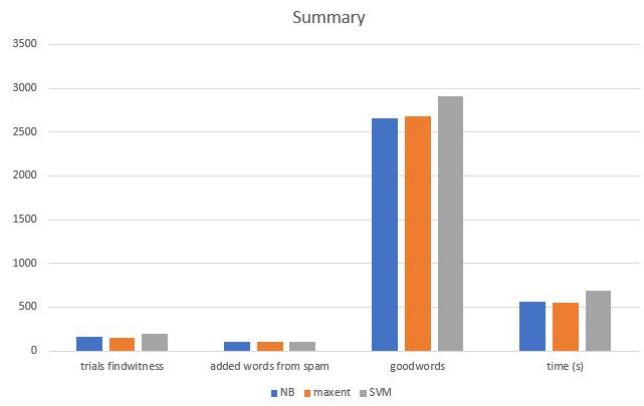

Fig. 1. Summary of results visualized

The first row talks about how many trials were needed for each classifier to convert a legitimate email to a spam email. These are 158, 155 and 200 for Naïve Bayes, Maxent and SVM respectively. The second row tells how many words from the spam email were needed to be added to the legitimate email to convert it to spam. The values are 102, 101 and 109 for Naïve Bayes, Maxent and SVM respectively. The third row gives the count of "good words" i.e., the number of words which when used can convert a spam email to a non-spam and hence help pass through the filter. These are 2658 for Naïve Bayes, 2683 for Maxent and 2912 for SVM. Finally, the last row is all about time efficiency (inefficiency in literal terms) of a filter to convert spam to nonspam. This is in seconds. The time taken by Naïve Bayes is 562.2 seconds, Maxent takes 547.1 seconds and SVM takes 689.9 seconds. The summary of my analysis is as follows.

1) SVM is best at efficiency in terms of trials needed to fool it and the number of words needed to fool it. In either way, it takes the largest number of trials and a larger number of words to let the spam message pass through it. Similarly Maxent performs the least in both the conditions.
2) When it comes to the detection of words, Naïve Bayes performs the best as out of 3000 spam words, it allows the least number of words to pass through it. In this case, SVM allows most emails (2912) and is the worst performer.
3) When time efficiency is considered, Maxent is the best performer and SVM is the lowest performer.

Considering all the above facts, we can safely conclude that when all the above measures are considered, Naïve Bayes is the safest performer. It should also be noted that once trained, detecting spam words is the most important task of a classifier. Naïve Bayes does this the best.

The result from the individual criteria is shown in the figures below. They visualize the following:

- Fig 2 : Trials needed to convert a non-spam email to spam email
- Fig 3 : Words needed to convert a non-spam email to spam email
- Fig 4 : Good words obtained by each classifier
- Fig 5 : Time efficiency of each classifier

VI. Drawbacks

1) The classifiers trained in this project allow more than 80% (more than 2400 out of 3000 dictionary words) of the dictionary words when added to spam mail to pass through the filters. This means the training has to be done over a larger dataset to reduce the number of words allowed and to increase variance
2) The project is static. When implemented in real-time, it might not show performances as it is showing now since thee training and implementation is now enough for the real-time tasks.
3) The dataset used for the second part of the project is just a pair of spam/non-spam emails to compare accuracies

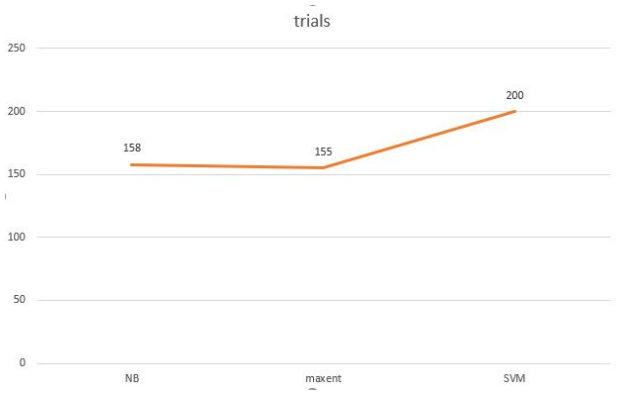

Fig. 2. Trials needed to convert a non-spam email to spam email

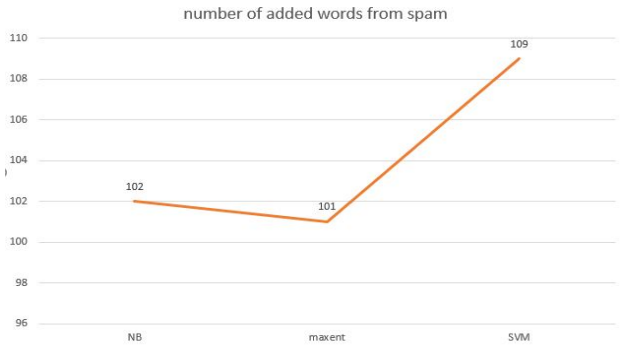

Fig. 3. Words needed to convert a non-spam email to spam email

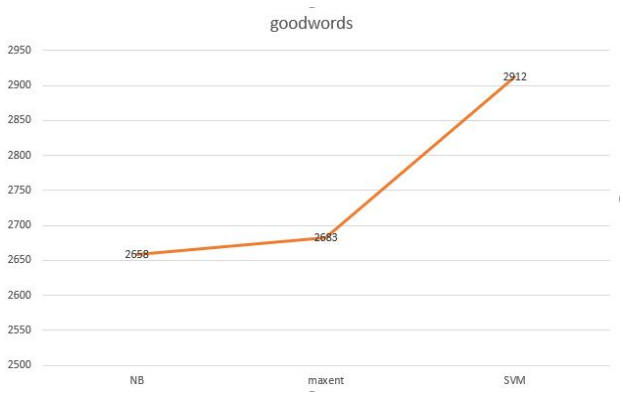

Fig. 4. Good words obtained by each classifier

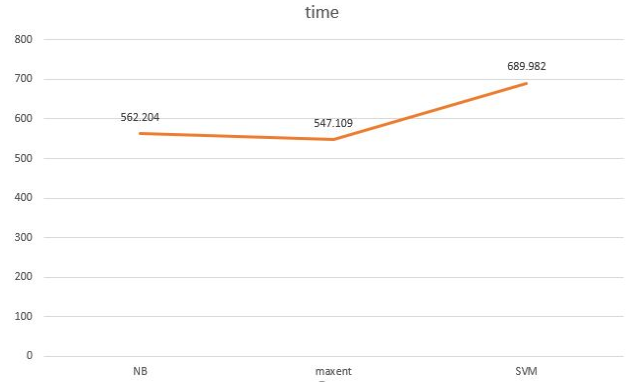

Fig. 5. Time efficiency of each classifier

of the three filters on the same set of spam/non-spam email combinations. This creates a bias. Probably if a larger dataset is used, the performance may be better.

VII. Conclusion

In this project we saw some of the types of attacks on spam filters. These attacks can be broadly classified as passive and active attacks. Dictionary attacks, frequency attacks and frequency ratio attacks are some of the attacks we saw in passive attacks and we spoke about First-N attack in active attacks. We also devised methods to implement the above-said attacks. We also understood the concept of "good words" and how using good words an attack can be caused. The paper – "GoodWord Attacks on Statistical Spam Filters" has been implemented in this project. I sincerely thank Dr.Luo for providing me with an opportunity to learn and understand the spam filters in-depth and use machine learning algorithms to implement them and also implement and understand the attacks on them. I also our T.A., Nushrat, for guiding me with the project as happily as ever, as and when required. Thank you!

NOTE: The images on the next page show some snippets from the results after the codes are run.

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.
[4] Zhang, L., & Yao, T. (2003). Filtering junk mail with a maximum entropy model. Proceedings of the 20th International Conference on Computer Processing of Oriental Languages (ICCPOL 2003) (pp. 446–453). ShenYang, China.
[5] Graham-Cumming, J. (2004). How to beat an adaptive spam filter. MIT Spam Conference. Cambridge, MA.
[6] See https://appliedmachinelearning.blog/2017/01/23/email-spam-filter-python-scikit-learn/

```
155


[0. 1.]
[0. 1.]
**********
156
ADDED TO THE LIST
102


[0. 1.]
[0. 1.]
**********
157


[0. 1.]
[1. 1.]
**********
--- 562.2047770023346 seconds ---
PS C:\Users\krupa krishnamurthy\NB\NB_MT\dataset\bare\p2\NB_2>
```

Fig. 6. FindWitness for NB

```
count of good words      2655
**********
RESULT
[1. 1.]


count of good words      2656
**********
RESULT
[1. 1.]


count of good words      2657
**********
RESULT
[1. 1.]


count of good words      2658
**********
PS C:\Users\krupa krishnamurthy\NB\NB_MT\dataset\bare\p2\NB>
```

Fig. 7. Count of Good words found by NB filter

```
152


[0. 1.]
[0. 1.]
**********
153


[0. 1.]
[0. 1.]
**********
154
ADDED TO THE LIST
101


[0. 1.]
[1. 1.]
**********
--- 547.1092908382416 seconds ---
PS C:\Users\krupa krishnamurthy\NB\NB_MT\dataset\bare\p2\maxent_2>
```

Fig. 8. FindWitness for Maxent

```
RESULT
[1. 1.]


count of good words      2682
**********
RESULT
[1. 1.]


count of good words      2683
**********
PS C:\Users\krupa krishnamurthy\NB\NB_MT\dataset\bare\p2\maxent>
```

Fig. 9. Count of Good words found by Maxent filter

```
198
ADDED TO THE LIST
108


[0. 1.]
[0. 1.]
**********
199
ADDED TO THE LIST
109


[0. 1.]
[0. 1.]
**********
200


[0. 1.]
[1. 1.]
**********
--- 689.9825859069824 seconds ---
PS C:\Users\krupa krishnamurthy\NB\NB_MT\dataset\bare\p2\SVC_2>
```

Fig. 10. FindWitness for SVM

```
RESULT
[1. 1.]


count of good words      2911
**********
RESULT
[1. 1.]


count of good words      2912
**********
PS C:\Users\krupa krishnamurthy\NB\NB_MT\dataset\bare\p2\SVC>
```

Fig. 11. Count of Good words found by SVM filter