**EMAIL SPAM FILTER**

Nisarg Hareshbhai Shah

Sugandha Kher

College of Computer and Information Science

Northeastern University

**Abstract**

With increasing security measures in network services, exploiting the security vulnerability without any prior access to the vulnerable system is getting harder. As a result, attackers focus on more reliable attack vectors like emails. Victims are infected using either malicious attachments or links leading to malicious websites. Therefore, efficient filtering and blocking methods for spam messages are needed.

According to a study by Radicatti Research Group Inc., spam costs businesses $20.5 billion annually in decreased productivity as well as in technical expenses. In this paper, we introduce a more reactive approach rather than a proactive approach for spam identification. Based on the collected information, we plan to develop a spam identification engine that would employ Naïve Bayes’ classifier to identify spam.

**Introduction**

In the recent years, we observe a shift how attackers proceed to compromise system on a larger scale: instead of using random scanning and remote exploits against common Windows network services, more and more attacks use email messages as propagation vector. These spam messages either contain a malicious attachment or a link to a malicious web page to compromise victims by exploiting client side applications, like the victim’s browser.

Current approaches to deal with email spam are reactive. In our paper, we have attempted to follow a reactive approach towards finding email spam. The idea is: Given a large collection of email messages, collected at the end of user’s mailboxes or dedicated mailboxes (so called spamtraps), the algorithm extracts features of all messages that can

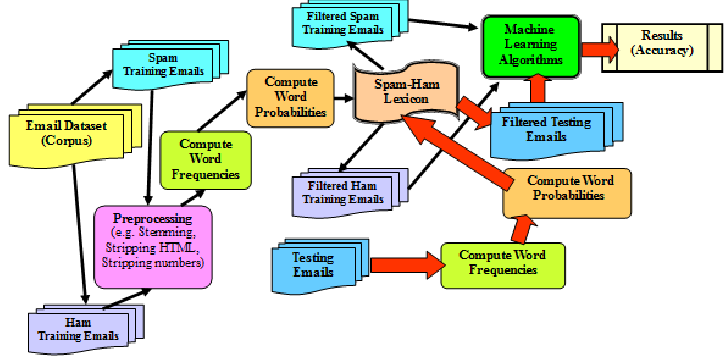
be used to distinguish pam from ham messages. For instance, by using Bayesian network or any other

Machine Learning techniques, email spam classifier can be developed.

A complementary approach is to generate a blacklist of IP addresses that are known to be related to spam. Such blacklists can, for example be constructed by extracting frequently appearing sender IP addresses from spam email headers [6]. Another example are URIBLs (Uniform Resource Identifier Blacklists) that list domain names that appear in URIs such as web sites U. Flegel and D. Bruschi (Eds.): DIMVA 2009, LNCS 5587, pp. 38–47, 2009. c Springer-Verlag Berlin Heidelberg 2009 Towards Proactive Spam Filtering 39 mentioned in messages more than a given threshold of times [7].

In our experiment, following a Bayesian approach which Graham and Robinson described and developed in early 2000s, we build a Bayesian classifier that accurately categorizes emails in ‘Spam’ or ‘Ham’. To fulfill this task, we took data from CSMining group which contained email data set in 10 folders.

The contribution in this project is three-fold. After we classified the data from our data set into training and test data set dividing in ratio of 70:30. Firstly, we processed the data extracted from the emails for stemming, removing stop words and punctuations and lemmatizing. Secondly, we applied the Bayes’ formula and calculated probability of spam email given the email and probability of ham email given email. Thirdly, we evaluated our classifier against already implemented library nltk against accuracy, precision and recall.



**Methodology**

In this paper, we discuss our Machine Learning algorithms ability to successfully recognize and classify the email as being a spam or ham. Our goal is to predict correct classification for a given email as being spam or ham. For this we train our data set to classify the emails into spam or ham. After running the training set, we run our classifier on test set to predict accuracy.

**Data Source**

For this project, we used the data set from CSMining group. The data was contained in 10 folders. To create an algorithm for spam detection, we need to teach our program what email spam looks like. For this we divided our data set into training and testing set: taking 7 folders containing emails as our bareTesting set and 3 folders as our bareTraining set. For training data, we have total of 2024 emails out of which 336 emails are spam and 1688 are ham emails. Spam emails are characterized by the prelabel in the file name as “spmsg”. For the Test set we have total of 2893 emails over which we test our classifier.

**Libraries Used**

* *nltk:* For our evaluation, we compared our accuracy with accuracy through using nltk library.

**Techniques**

According to Bayes’ theorem:

P(A|B) = P(B|A) P(A)P(B)

where

* A and B are events, P(B) 0
* P(A) and P(B) are probabilities of observing A and B without regards to each other.
* P(A|B) is conditional probability, probability of A given B is true.
* P(B|A) is conditional probability of B given A is true.

Using the Bayes’ theorem in our classifier, we consider set A as email being spam and set B as the content of email.

So, we calculate,

P(Spam | EmailContent) = P(Email | Spam) P(Spam)P(Email)

and

P(Ham | EmailContent) = P(Email | Ham) P(Ham)P(Email)

If P(Spam | Email) > P(Ham | Email), we classify email as spam otherwise a Ham.

Since we equate both equations the denominator would get cancelled out. So we are just concerned about getting probability of spam and ham emails and the conditional probability of email given it is spam and probability of email given it is ham.

**Implementation**

The first fold of our project was to extract the bare data from training set and pre-process the data for stemming, removing stop words, punctuation.

Since, in the data set spam email are pre-labeled, we get the count of number of spam emails in the training set and thus calculating independent probability of spam emails and probability of ham emails.

For our second step, since for our classifier to work we are implementing Bayes’ algorithm and comparing, probability of Spam given Email content and probability of Ham given Email content.

P(Spam | EmailContent) = P(Email | Spam) P(Spam)P(Email)

and

P(Ham | EmailContent) = P(Email | Ham) P(Ham)P(Email)

For our comparison,

if P(Spam | Email) > P(Ham | Email), the email is classified as a spam.

Therefore, from the formula we get, email is spam if:

P(Email | Spam) P(Spam) > P(Email | Ham) P(Ham).

We already calculated the independent probabilities of Spam and Ham. Hence, we already have P(Spam) and P(Ham).

To calculate conditional probability of email given Spam or Ham, we considered two models.

*Bag-of-words model*

First, we calculated the conditional probability using bag-of-words model. Bag-of-words model treats each piece of word as a bag of individual word not considering their ordering. For each word in email we calculate the frequency with which each word shows up in spam email or a non-spam email. For instance, for a word “free”, we calculate P(‘free’ | Spam), by counting the frequency with which word ‘free’ occurs in all Spam emails combined and then divide by the total number of words in all Spam emails combined.

For the implementation of bag of words, we maintain two dictionaries, train\_positive and

train\_negative. train\_positive maintains the frequency of word as obtained in the Spam emails and train\_negative counts the frequency of word in non-spam email.

*tf-idf model*

Tf-idf model is a simple twist in bag-of-words model. Instead of calculating the frequency of all the words, tf-idf calculates the normalized frequency of each word where each word is divided by number of emails where the word occurred.

The tf-idf is the product of two statistics, term-frequency and inverse document frequency.

In case of term frequency, we use raw count of the word in a document. Denoting raw count by f(t,d), the simplest tf scheme would be: tf(t,d) = f(t,d). Other possibilities include:

* term frequency adjusted for document length: f(t,d) / (number f words in d)
* logarithmic scaled frequency: tf(t,d) = log(1 + f(t,d))
* augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the raw frequency of the most occurring term in the document: tf(t,d) = 0.5 + (0.5\*f(t,d) /max{f(t’,d), t’ d})

The inverse document frequency, measure of how much information the word provides i.e., whether the term is common or rare across all documents. idf is logarithmic scaled inverse fraction of the document that contained the word, obtained by dividing total number of documents by the number of documents containing the term.

idf(t,d) = log N|{d D: t d}|,

where,

N: total number of documents

|{d D: t D}|: number of documents where term t appears. Since if a term does not appear in documents, it will lead to division by 0, hence we usually adjust denominator to 1 + |{d D: t D}|.

For calculating conditional probability through tf-idf, we maintain two dictionaries, positive\_index and negative\_index. Both the indexes store the documents and the frequency of the word that occurs in that document.

from both the indexes, we count the total frequency of the word occurring in all the document. Then multiplying the total of spam or ham count and dividing by the number of documents where the word is occurring in both spam and ham emails.

**Evaluation**

For the third fold of our project, we need to evaluate our classifier. The aim of this stage is to evaluate the classification process conducted.

An ideal spam filter would autonomously, immediately, and perfectly identify spam as spam and non-spam as non-spam. To evaluate a spam filter, we must somehow measure how closely it approximates this ideal. Furthermore, whatever measurement we use should reflect the suitability of the filter for its intended purpose.

For our evaluation, we considered following factors while evaluating the filter:

* Precision
* Recall

The following definitions are explained with respect to the problem domain:

* False Positive
* False Negative
* True Positive
* True Negative

It is not obvious how to measure any of these dimensions separately, nor how to combine these measurements into a single one for the purpose of comparing filters.  Nevertheless, reasonable standard measures are useful to facilitate comparison, provided that the goal of optimizing them does not replace that of finding the most suitable filter for the purpose of spam filtering.

Failures to identify non-spam and spam messages have materially different consequences. Misclassified non-spam messages are likely to be rejected, discarded or placed in isolation.

Any of these actions eventually increase the risk of loss of information contained in that message or at least information getting delayed substantially. We cannot predict exactly how much risk and delay are incurred, as are the consequences, that depend on the nature of the message.

Some messages are simply more important than others, while others are more likely to be missed, or delivered by separate channels, if they go astray. For instance, advertising from a frequent flier program is less important than an electronic ticket receipt, but the latter is certain to be missed and retrieved, either from quarantine or from a different medium. On the other hand, failure to deliver immediately a message from one’s spouse to “come home right away” could have serious consequences.

For these reasons, one must be cautious about characterizing failures to deliver non-spam in terms of a simple proportion, as such failures are rare events with causes and consequences that defeat statistical inference.

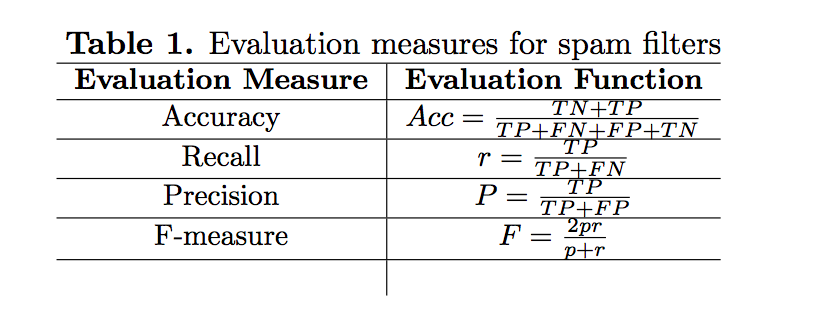
With this warning, **false positive** rate — the proportion of non-spam messages identified as spam is a reasonable first-order measure of failures to identify non-spam.

Failures to identify spam also vary in importance, but are generally less important than failures to identify non-spam. Viruses, worms, and phishing messages may be an exception, as they pose significant risks to the user. Other spam messages have impact in proportion to their volume; so, **false negative** rate the proportion of spam identified as non-spam is also an apt measure.

The overall efficacy of a email spam classifier may be characterized by the pair False positive and False Negative. A classifier with lower False Positive rate and False Negative rate than another is superior. Whether a classifier with a lower False Positive rate and higher False Negative rate is superior or inferior depends on the user’s sensitivity to each kind of error.

For our spam filter, the evaluation metrics used to evaluate the performance of email classifier based on our data set is chosen. The simplest measure is filtering accuracy namely percentage of messages classified correctly.

The following table shows evaluation measures for spam filter:



where accuracy, recall, precision, F-measure, FP, FN, TP and TN are defined as follows:

i.) Accuracy: Percentage of correctly identified spam and not spam message.

ii.) Recall:  Fraction of relevant instances that have been retrieved over the total number of relevant instances.

iii.) Precision: Percentage of correct message for spam e-mail.

iv.) F-measure: Weighted average of precision and recall.

v.) False Positive Rate (FP): The number of misclassified non-spam emails.

vi.) False Negative Rate (FN): The number of misclassified spam emails.

vii.) True Positive (TP): The number of spam messages are correctly classified as spam.

viii.) True Negative (TN): The number of non-spam email that is correctly classified as non-spam.

**Experiments & Results**

During development of our final version of email spam filter, we experimented with a lot of results and datasets.

Initially for our classifier, we used email data set of CSMining group that were already classified into training and test sets. The problem with the test set was that we could have tested our filter over the test set but since in the test set, there was no classification of spam or ham, we were unsure whether our filter was giving correct results for that test set. For this reason, we decided to change the test set and took another data set from CSMining group, in which the test set given was already classified into spam and non-spam. Over the new data set we could easily test the accuracy and precision of our spam filter.

Also, while we were using the initial data set, we planned to evaluate our filter in a different way.

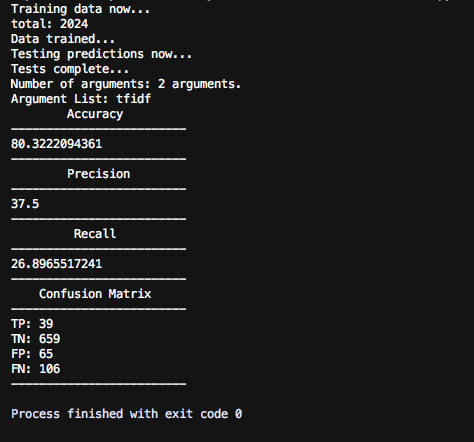
Our idea was to use the nltk library and the compare the results of nltk and our filter over the test set and then determine the accuracy and precision.

But instead of using another library for evaluation, we changed our data set to determine accuracy and precision of our own filter.

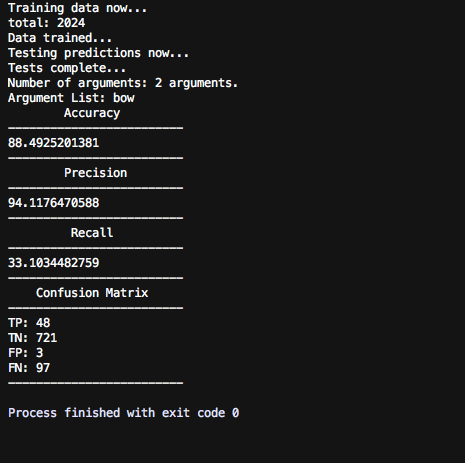
While in our second stage, where we were applying Naive Bayes’ algorithm to find out the conditional probability, we applied two models as explained in the previous section.

While we were expecting the tf-idf model to give more accurate results in terms of accuracy and precision, it was actually unexpected to see that our bag of model was giving much higher accuracy and precision than the tf-idf model.

The following was the result from a tf-idf model:



The following was the result from bag-of-words model:



While the results were unexpected, we found the answer to such mismatch.

The tf-idf model can stretch the word count as well as compress it. In other words, it makes some counts bigger, and others close to zero. Therefore, tf-idf could altogether eliminate uninformative words.

Another different set of results we found during tf-idf model was by eliminating the new words encountered during the test data set.

<Explain>

<Also attach screenshot>

<or table>

**Conclusion**

E-mail spam filtering is an important issue in the network security and machine learning techniques; Naive Bayes classifier that we used has a very important role in this process of filtering email spam. The quality of performance Naive Bayes classifier is also based on datasets that used. As we can see, dataset that have fewer instances of e-mails and attributes can give good performance for Naive Bayes classifier. Naive Bayes classifier also can get highest precision that give highest percentage spam message manage to block if the dataset collect from single email accounts.

We have the accuracy and running times of all the algorithms and experiments performed.

|  |  |  |
| --- | --- | --- |
| Algorithm/  Approach | Accuracy | Execution Time |
| Naive Bayes’ (First approach) |  |  |
| Naive Bayes’ (Second approach) |  |  |
| Using tf-idf |  |  |
| Using bag-of-words |  |  |
| Eliminating new words from Test set |  |  |
| Including new words from Test data set |  |  |
|  |  |  |

The best results would be the algorithm where accuracy as well as execution time are optimized. As we can see above the Naive Bayes’ algorithm gives maximum accuracy and best execution time when run with bag-of-words model.

While there is a stark difference in the performance of the three algorithms, these could vary with different data sets. The ratio of ham/spam messages, the number of data points in the training set and the number of data points in the testing set can have an effect on the performance of the algorithms.

**References**

1. <https://hackernoon.com/how-to-build-a-simple-spam-detecting-machine-learning-classifier-4471fe6b816e>
2. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.178.4052&rep=rep1&type=pdf>
3. <https://www.safaribooksonline.com/library/view/feature-engineering-for/9781491953235/ch04.html>
4. <https://aip.scitation.org/doi/pdf/10.1063/1.4958497>
5. <https://amor.cms.hu-berlin.de/~jaehnicp/project/spam-filter/bayesian_report.pdf>
6. <https://link.springer.com/chapter/10.1007/978-3-642-02918-9_3>
7. <https://pdfs.semanticscholar.org/7be9/7249e88516058f34ad3836c16096413bd02b.pdf>
8. <http://sce.uhcl.edu/hayesivj/projects/spamFilter.pdf>
9. <https://plg.uwaterloo.ca/~gvcormac/spam/>
10. <https://appliedmachinelearning.wordpress.com/2017/01/23/email-spam-filter-python-scikit-learn/>
11. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.5562&rep=rep1&type=pdf>