A Naive Bayes Email Spam Filter

CS 480 Project Final Report

Chris To, Valerie Free, Nathan Wikle

Project Summary: As email has become omnipresent in the twenty-first century, unsolicited bulk email, or "spam" messages, remain a common problem encountered by users. According to a 2013 estimate, an average of 97.4 billion spam messages were sent each day, meaning approximately 78 percent of all email volume is spam (Commtouch). Thus, a fundamental component of all email providers is the performance of their spam filter. It must catch the vast majority of spam messages, but more importantly, it should minimize classifying legitimate emails as spam. As a result, a variety of spam filtering methods have been developed, with varying levels of complexity and success. For our project, we chose to implement one of the most popular spam email filtering techniques, a Naive Bayes spam filter.

The Naive Bayes spam filter is based on the principle of Bayes' Theorem. At its most basic level, a Naive Bayes classifier is trained on a large set of emails, consisting of both spam emails and legitimate, or "ham", emails. It then uses the frequencies of words seen in the spam and ham training emails as estimates of conditional probabilities, and then uses these conditional probabilities when classifying a new email. More specifically, we used multinomial Naive Bayes classifier with Boolean attributes. The formal classifier can be defined as follows (Note, the following notation is borrowed heavily from Metsis et al.):

Let each message d be represented as a vector $\vec{x} = \langle x_1, \cdots, x_m \rangle$, where x_i is the value of attribute (word) X_i. These attributes are Boolean: $X_i = 1$ if the message contains the word; $X_i = 0$ otherwise. We will consider two categories: c_s, spam, and c_h, ham. Then, the criterion that a message $\vec{x}$ is classified as spam is

$$\frac{p(c_s)\prod_{i=1}^{m} p(x_i \mid c_s)^{x_i}}{\sum_{c\in\{c_s,c_h\}} p(c)\prod_{i=1}^{m} p(x_i, c)^{x_i}} > T, \tag{1}$$

where $p(c_s)$ is the proportion of training emails that are spam, and each $p(x_i \mid c)$ is given using a Laplacian prior:

$$p(x_i \mid c) = \frac{1 + N_{x_i,c}}{m + N_c}.$$

$N_{x_i,c}$ is defined as the number of training messages of category c that contain word x_i, and $N_c = \sum_{i=1}^{m} N_{x_i,c}$, the sum of all $N_{x_i,c}$ for each word x_i in category c. A Laplacian prior is used to account for words that were not contained in the training emails. T is the cutoff value for classification as spam. The default is $T = 0.5$.

Implementation: We implemented our multinomial Naive Bayes classifier in Python from scratch, and used two existing email corpora for classification and test purposes. For the classification process, we relied on the EnronSpam and LingSpam email sets, consisting of thousands of emails, containing both spam and ham. A training set of emails was created from the corpora, and each email was classified as spam or ham. The Naive Bayes filter was then trained and tested using the following algorithm.

For each word appearing in a document, the algorithm counts how many ham emails and how many spam emails in which the given word appears. The Laplacian prior is then calculated using the counts. We also keep track of the percentage of spam messages contained in the training email set. To validate our filter, we test the classifier on the test email set. For each message in the set, we calculate the value in the numerator of (1) for c_s and c_h. We then classify the message as spam if $c_s > c_h$, and ham if $c_s < c_h$. In doing this, we omit the cutoff value T. However, it would be relatively simple to implement and test for the optimality of this value in the future. We print out the number of messages successfully classified as spam and ham in order to obtain success rates for our filter.

Results:

Team Contributions: This project was in many ways a team effort, with multiple people involved in almost every step. However, certain steps can be more attributed to specific individuals than others. Valerie helped write the initial code for the spam filter, in particular implementing the dictionary of words found in the training email sets and calculating many of the required counts necessary for the classification calculations. She was directly responsible for creating the power point slides used in our presentation, and she helped write the first two project reports and found references. Nathan wrote the initial classification code for the spam filter. He determined which spam filter to use, read literature on the best version of the Naive Bayes classifier, and specified how the multinomial Naive Bayes classifier with Boolean attributes should be implemented. He wrote the final project report, and helped edit previous reports. Chris rewrote and cleaned up much of the initial code, giving the classifier a class structure, adding much documentation, and speeding up the performance of the classifier. He found the training and test corpora that we used, provided the input files needed to test our code, and reported the results. He also helped write the progress reports and found some of the references. However, despite the partition in work, everyone was extremely cooperative and willing to work on whatever task was needed of them.

References

Commtouch. Internet Threats Trend Report. (2013) *Commtouch Software Ltd.*

Metsis, V., Androutsopoulos, I., Paliouras, G. Spam Filtering with Naive Bayes - Which Naive Bayes? (2006) *CEAS 2006 - Third Conference on Email and Anti-Spam.* Mountain View, California USA.