Spam Filtering by Noah Portnoy

4/22/14

Today I will be talking about spam filtering! I can see how excited you guys are*.* But you should be excited. Spam filtering is an important part of our electronic lives. 80-85% of email sent around the world is spam. Spam also appears in instant messaging, forums, text messaging, social networking, online games, and blogs. According to surveys and analysts, spam costs Internet users anywhere from $20 billion to $50 billion or more. It’s clear that spam has a significant impact, and we should try to avoid it as much as possible.

Today I will focus on inbound email spam filtering; that is, filtering spam at the recipient’s end. How do we do it? Here’s a sample spam email. What do you think? Any ideas for how we could filter out messages like these?

Let’s see: spam messages might often begin with “Dear Friend”, like this one. Maybe we should try to filter out messages that begin in this way. Or maybe we filter out messages that have the word “click”? That could be common. Or if they have the words, “Bank of Nigeria”, like this message.

It turns out there’s a better way, and it doesn’t involve human effort to pick out specific spam-like phrasing. The technique is called Bayesian spam filtering, and it used in almost all spam filtering programs today.

Let’s break down how a Bayesian spam filter works. First, we have to “train” our filter, so it knows it what is and isn’t spam. We start by giving the program both a spam and a non-spam corpus of emails, and indicate to the program which is which. For each corpus, the program parses all of its messages and breaks them down into words, where words are composed of letters, digits, dashes, apostrophes, and dollar signs. We’ll tell the filter to ignore the case of words as well. The program also keeps track of how often a word appears in each corpus.

So what do we have so far? We have the words that appear in the spam and non-spam corpuses, and how often they appear in each. The program next has to determine a spam ranking for each word, and this is where the Bayesian component comes into play. Here’s a simplified version of Bayes theorem:

This equation says that the spam ranking of a word is equal to the frequency of the word appearing in the spam corpus, divided by the sum of the word’s frequency in both corpuses. Essentially, what percentage of the word’s occurrences appear in spam messages? We can do this operation for each word that we’ve found in the corpuses, with some added restrictions, and create a hash table mapping each unique word from the corpuses to a spam ranking.

Now comes the fun part: filtering! Let’s feed a new message into our hypothetical spam filter. Now what should the program do? Let’s have it look up the spam rankings of the words that appear in the message. For words in the new message that we do not have a spam ranking for, we’ll give them a ranking of 0.4, because it is more likely than not that if the filter hasn’t come across a word before, it is an unusual word in the given language. Now the program will narrow down to consider only the 15 most “interesting” words. “Interesting” words are defined by how far their spam ranking is from the neutral 0.5 (0 and 1 are the lowest and highest spam rankings, respectively).

Here’s an example of the 15 most interesting words in a spam message.

So the spam filter has the 15 most interesting words and their rankings. Now all the program has to do is take those spam rankings for the 15 words, and classify the message as spam or ham.

Here are a few methods. One method is the majority. Set a threshold, and see whether more rankings are above the threshold or below. If a majority of the rankings is above the threshold, classify the message as spam. If not, classify it as ham. The majority method turns out to be fairly weak. If some probabilities were to change among the 15 words, you could imagine how that might not affect the classification outcome.

A better way to classify the message from the 15 most interesting words is by using the mean average of the spam rankings. Add them all up and divide by the number of words, 15, and then compare to a threshold, just as before. The mean method works significantly better than the majority method, but we can do even better.

The naïve Bayes classifier method is called naïve because it assumes the spam word rankings are purely independent events. In reality, this is not exactly true, as nouns often follow adjectives, for example, but we still get very useful results. We can combine the spam rankings of the 15 words as follows, following Bayes theorem:

are the spam rankings for each of the interesting words

This method using a combination of probabilities proves to be the most accurate method for detecting spam and reducing the number of false positives. False positives in this case are ham messages that were improperly classified as spam.