SVM Machine Learning Training Tactics

We took three main steps to train our SVM algorithm to distinguish between “spam” and “ham” emails. First we found a set of words that were representative of words found in spam and ham emails to use as features, then we created a set of custom features that we thought might further indicate the nature of the email, and then we identified the optimal value of C.

In our first stage of training our algorithm, we wanted to identify a set of words that were representative of spam and ham emails to use as features. We did this by identifying the 7,000 most common words in each of the spam and ham email sets, and storing these words in python sets. We then subtracted the spam set of words from the ham set of words, therefore eliminating any of the words that they had in common. We used the words that remained after the set subtraction as our initial set of features to train on. We then tested the algorithm and looked at the emails that were misclassified, and tried to identify words that we could add as features that would have helped the algorithm classify them correctly. We added these words, as well as words that we thought might be able to help, just based off of experience (words related to gambling, sex, investments, medication, etc.) to the list of words that would be used as features. This set of words makes up the majority of our features.

From there, we formulated a few custom features that we thought might be able to increase the accuracy of our algorithm. We decided to flag any email that had any of the following characteristics:
- If there are less than 15 words.
- If there is an unusually high ratio of exclamation points and question marks relative to periods.
- Of there are an abnormal number of “*” or “$” characters.
- If there is more than one person cc’d on the email.
- If there was a significant proportion of words that were misspelled or not in english.
- If the email is in response to another email (i.e., somewhere in the email there is a “re:”, referring to a previous email.

We added these custom features to the set of words we developed in the first step. The resulting set of words and custom features made our complete set of features for our SVM to train on.

The last step we took to train our algorithm was to tune the parameter C in order to identify the value that would optimize the cross validation accuracy. We did this by running the SVM on the training and validation on a range of values for C, beginning with 0.02 and going up to 2, increasing by an increment of .02, and making note of the value of C that resulted in the highest cross validation accuracy. It should be noted that each time we ran the SVM we divided up the
training set and cross validation set (70% training, 30% cross validation) randomly, so each run of the algorithm produced slightly different accuracies, even when we were not changing the value of C. When we looped through values of C multiple times, we found that a value of C that was somewhere between 0.28 and 0.32 rendered the highest cross validation accuracy. Thus, in our final output we used a 0.3 for our C parameter.

One thing we tried was to use every word in the english dictionary as a feature for the SVM to train on. This may have been a good idea, but we decided it might have resulted in overfitting, and the runtime was far too long. Therefore, we abandoned this tactic, and stuck with what we had before.