The first way we tried to optimize our SVM was to improve and increase the size of our word list. Our first idea was to create a dictionary of every single word in every email, ham or spam. Our dictionary also kept track of the frequency of each word, not just a binary indicator of whether or not it existed. We sorted our dictionary by frequency and created a new word list by taking the 2000 most frequent words. We also experimented with the size of our wordlist and settled at 3000. The way we split the emails also ended up putting a lot of different punctuation into our word list which figured would just be given really insignificant weights and not be an issue. We also didn’t worry about punctuation because it may also predict spam or ham, for instance if ham tends to include a lot more punctuation marks. Extracting our features using this method hugely improved our test accuracy.

Next we tried to identify whether or not an email had misspellings in it. We figured that spam emails might have misspelled words more often. We used the enchant library to check for every word in every email, if it was a correct spelling of an English word. We only looked as misspellings for words with a length longer than 1 in order to avoid picking up symbols and punctuation. We added a feature to our feature vector for every email that was the amount of misspellings in an email. When we implemented this optimization technique our training accuracy actually slightly decreased. We also experimented with the length of words to check spellings for (2 to 5), but it did not improve accuracy. We then added it as a binary feature indicating as whether or not there were more than 5 misspelled words in the email. This seemed to help a bit.
We tried to optimize our variable c by writing a ‘for’ loop to try different values. We wanted to be able to keep track of which value of c generated the most accurate model. First we tried a range of .01 to 1 with a step size of .02. In that test we found that .07 easily looked like the best choice. Then we ran the same test from .06 to .08 with a step size of .005 to better converge on the optimal c value. We found that a c of .07 and .075 both consistently return an accuracy of 98.62%. We chose to go with a c value of .075 because it had a higher precision with the same recall as .07.

The next thing we tried was taking a look at the examples that we misclassified to better understand where our model was failing. We compared the predictions in the output file and the original labels for the validation data and printed the filenames that didn't match up.

After analyzing the misclassified emails, we added a couple more features to our model. First, we saw that some of spams had their entire content in the subject and only a couple words in the body. So we split the email into subject and body and if the length of subject is greater than the length of the body, we added a binary feature. While adding these new features, we made sure the feature names were unique by using a constant N for the number of words in the word list and adding 1 to it each time we add a new feature.

We also noticed that a lot of spams did not use capitalized words after periods. So we went through every word after a period and checked if they were capitalized or not. We added a binary feature indicating whether or not at least half the words after periods are capitalized.