Our initial approach to improving the spam classification algorithm was to increase the cross validation accuracy by growing the number of features we trained on. To compile a more robust word list (which in turn gave us more features), we began by querying the training data for the most common words in emails which were marked as spam. This was a good initial first step, but we quickly recognized that we could further improve cross validation accuracy by adding the most common words in emails which were marked as non-spam as well – the logic behind this being that if an email contained a word that was common amongst non-spam emails, then it decreased the likelihood that the email in question would be classified as a spam email. We were able to query for the most common words in spam and non-spam emails using the Counter function from the collections library, which allowed us to tag and extract the most popular words in the two sets of emails. We then went about combining these two sets of most common words. We wanted the most “polarizing” words between the two sets. To achieve this, when constructing our ultimate wordlist, we removed any word that was common across both lists. This helped to create a collection of words which were distinct to each category of email. Additionally, we executed several pre-processing steps when generating our word list. We capitalized all words across all emails so that comparison with our wordlist would not be case specific. We also initially excluded any words of length less than four from our list. This allowed us to remove words which might not only be common across spam and non-spam, but also are less descriptive and potentially detrimental to include in our wordlist. In the end, we found this approach to be unfavorable towards the performance of both our cross validation and training accuracy, so we removed any stipulations on word length.

After crafting our enhanced word list, we had to pick an ideal number of most popular words to search for in order to generate the list. We started by searching for just the 100 most common words in spam and 100 most common words in non-spam emails, but quickly decided that we could achieve better accuracy on the training and cross validation sets if we built our word list from a greater number of most common words. We pushed our system to take the 10,000 most common words on each of the two lists, but recognized that using this many features was extremely unnecessary. Not only did it fail to increase the cross validation accuracy, but it also failed to increase the training accuracy – the set which the most common words/features were being pulled from. We ended up settling on the 500 most common spam and non-spam words, which gave us a word list length of 430. This value maximized the cross validation accuracy while minimizing the variance between it and the training accuracy.

We also decided that it would be beneficial to start with features that had non-binary weights. This meant that instead of determining if an email contained one of the words in our wordlist, we wanted instead to determine the frequency with which the email contained that word. We anticipated this being beneficial as emails more heavily laden with spam words (i.e. ED medicine names) would be even more likely to be flagged as spam, while words which contained multiple mentions to non-spam words (i.e. Enron) would be even less likely to be flagged as spam. We later realized this had no effect on accuracy as it is unclear is SVMlight recognizes frequency versus appearance for a feature.

After making multiple adjustments to our word list, we were interested in “manually” studying some of the cases of miss-classification where spam was flagged as non-spam and non-spam was flagged as spam. To do this, we took the training and validation predictions and compared them against the labeled training and cross validation sets. We created a list of all the email files which we mis-classified. Using this list we iterated through and pull out the most common words from the mis-labeled emails. We added the words which made up the symmetric difference of the two sets to our initial words list. In total, this involved the addition of 9 extra words. The rationale behind this process was that there might
be several key words that are extremely specific to certain types of spam or non-spam, but not common across all emails of a giving type. Finally, we visually analyzed the mis-classified emails in the training set in an attempt to pull out any final words or misspelled words which we felt heavily indicated spam or non-spam. To verify that this step was beneficial, we ensured that the cross validation accuracy improves throughout these steps.

We also began testing variations in the hyper parameters of the SVM. Specifically, we were interested in adjusting the kernel of the SVM (which is linear by default). However, as part of some background research through the papers cited on the SVMLight webpages, we stumbled upon a report by Thorsten Joachims (the creator of SVMLight) titled “Text Categorization with Support Vector Machines: Learning with Many Relevant Features”. In this paper, Joachims noted that, “most text categorization problems are linearly separable,” and thus we determined that it would be optimal to maintain a linear kernel.