Corinne Bintz and Mira Chugh

November 2, 2018

CS451 - Professor Scharstein

HW 4: Spam Classification using SVMs

Building our first spam filter!

In training a support vector machine to classify email messages into spam and not spam, we started out by trying to extract better and longer word lists to build more powerful feature vectors. We first tried creating a list of the 3000 most frequent words from all emails. This ended up not improving the accuracy very much, so we decided to create a word list with the 3000 most frequent words found in emails classified as spam as well as a word list with the 3000 most frequent words found in emails classified as ham. To do this, we first made a dictionary with the the words in the corresponding set of emails as keys and the number of times they occured in the entire set as values. We then sorted the words by their counts and used the 3000 with the largest counts for our feature list. Between the two of these feature lists (top words found in spam vs. top words found in ham), when we trained our model using the ham word list, our model had a better accuracy. After trying these “frequency” word lists, we also created a word list with the top 3000 unique words for spam and top 3000 unique words for ham. We implemented this by taking the frequency dictionaries with the words as keys and counts as values (created earlier) and filtering out words that occurred in the other dictionary (spam if we were trying to create a unique ham dictionary and vice versa). These words lists, surprisingly, did not improve upon the accuracy of the model created using the non-unique, ham frequency word list.

After finalizing our word lists, we went on to do some parameter tuning. During this part of the process, we decided to mainly focus on finding an optimal value for C. To tune C, we trained our model using various values of C, starting at 1 and going up to 15, in increments of 1. We ended up finding that our model had the best accuracy with values of C equal to 1 and 3. From here, we used the same method used earlier, except we trained our model using values of C starting at .5, going up to 3.5, in increments of .25. Out of these C values, we found that our model had the best accuracy with a C value equal to .75.