HW6 CIS391 Artificial Intelligence

Tara Siegel and Joesph Trovato

Base code:

This project code expanded upon the Homework 5 code. The code that we started with implemented a simple naïve bayes classifier on the tokens returned by the nltk tokenizer. The tokens were all converted to lowercase and headers were kept in the data processed by the classifier. This basic model was tested on the dev set. And correctly classifed 200/200 ham emails and 153/200 spam emails. These preliminary results illuminated the fact that our original classifier was biased towards classifying mail as ham. Our approach was to improve on the tokenizer, so that we could give our classifier the best possible data and hopefully that would yield better results. We also wanted to add features for things such as the subject words, length of email, and capital letters in emails. The following tests are all using the hw6-spamham-data/dev set to judge performance. We assume 1-200 are ham and 201-400 are spam.

The tokenizer:

1. Stripped HTML tags – ham:194/200 spam:180/200
2. Stripped HTML content – ham:196/200 spam:186/200
3. Removed Punctuation – ham:196/200 spam:186/200
4. folded case to lowercase – ham:196/200 spam:186/200
5. excluded words <3 and >13 characters – ham:196/200 spam:186/200
6. stripped email header – ham:199/200 spam:186/200

From our first few alterations to the tokenizer we realized that small changes in how the emails are tokenized have big effects on their classification. We learned the there was lots of tokens in the header but they contained very little information.

The classifier:

1. Included email length - ham: 199/200 spam:187/200
2. Included the subject token in the classifier – ham: 199/200 spam:187/200

After making many alterations to the tokenizer that seemed like they were improving the performance, we realizer that our tokenizer was actually limiting our classifer so we went back and reduced the constraints we put on the tokenizer and saw a dramatic improvement.

The tokenizer (again):

1. include all tokens less than length (with punctuation) 13 – ham:197/200 spam:194/200
2. include all tokens less than length (without punctuation) 13 – ham:199/200 spam:194/200
3. including exclamation points – ham:197/200 spam:194/200
4. stopped folding case - ham:199/200 spam:194/200

Finally, after making these adjustment to the tokenizer, we tweaked the classifier once more to get the best results given the new data the the tokenizer was now revealing.

The Classifier (again):

1. included capital letters in features: ham:199/200 spam:197/200
2. applied all tokenizing techniques to subject: no change
3. included number of URLs as a feature – no change
4. included number of Recipients as a feature – no change

The final classifier was able to correctly classify 99% of the dev set 199 of the ham emails and 197 of the spam emails. The three spam emails that the classifiers had difficulty with were very difficult because of the content. One of the emails was not in English so its classification may have only been based of one or two data points. Another one of the emails was over 11,000 words. This email contained copied news articles and people's responses, and due to the formal nature of the new articles and colloquial nature of the response this look like ham text. Similarly, the one email that was misclassified ham, consisted of and address and a small sentence. The amount classification data available is very small. The content of the email was actually about spam, but we are not sure if this would affect the result. The current system is accurate to the point that we need more data in the dev and train sets in order to avoid simply adjusting the classifier to fit the outliers in the current set as opposed to actually predicting spam.

The final classifier was also tested on the training data which yielded 1 miss on ham and 1 miss on spam. The total correct was 1998/2000 or 99.9%. Similarly, on the dev set it missed only 4 of 400 emails yielding an accuracy of 99%.