Spam filter

1. corpus

1.1. tokenize

to tokenize the corpus, I used nltk. However, after running the program, I received the Unicode error in some instances, so I used this command to stop this error: encoding='utf-8',errors='ignore'.

1.2. Read file

one issue in reading every email was that I didn't want the list of words in the email to contain multiple instances of one word, because that would make my calculation in the training part harder, so after reading each email I converted every email to a dictionary to delete the multiples and then converted it back to a list.

1.3. read dataset

in this part, I designed the function in a way that it gets a general path to the data directory, and first combined it with the word "ham" and gets the Os to list everything in the ham and then went through it, and then did the same with the spam folder. I could also use os.list to list all the contents in the directory and then use a loop to add the words "ham" and "spam" to the path, but to prevent having yet another loop, I just added the word "ham" and "spam" as strings to the general path.

another issue that I had in this section, was that there were some hidden files in ham and spam folders that started with "." And I didn't want to read them. So I used this code to delete them from my list: if not i.startswith("."): i=os.path.join(ham_path, i)

2. nb

2.1. train

in train section, I created two separate dictionaries "spam_count" and "ham_count" to calculate the spam and ham probability of each word. Creating two dictionaries made more sense because at the end I were supposed to get both the spam probability and ham probability of each word.

2.2. Classification

For this part, I wanted the classifier to have access to the training data, so I saved the training data as self.spam-count and self.ham-count.

My main problem in this section was in calculating the spam score, because some numbers were very small and the spam score became so small that the computer couldn't calculate it and just assigned the 'nan' value to it. To prevent this problem, I used the logarithm method. I calculated the formula using logarithm but converted it back to the original probability scale by using the exponent method.

3. Main

This part interacts with the user. First it asks for the path to the train directory, then it asks the user if they want to classify just one email or a batch of emails, then asks for the path to them and then returns the result as a tuple of the label ‘spam’ or ‘ham’ and the spam score assigned to every email.

4. Extensions

Smoothing

One issue with my model was the problem of zero probabilities. To solve this issue, I applied Laplace smoothing both in train and in prediction. In train, I added one to both spam count and ham count of every word and I added 1 to both the number of total spam emails and the number of total ham emails. In prediction, Whenever the model encountered a word that it had never seen before, it assigned the account of one to it and then calculated its probability.