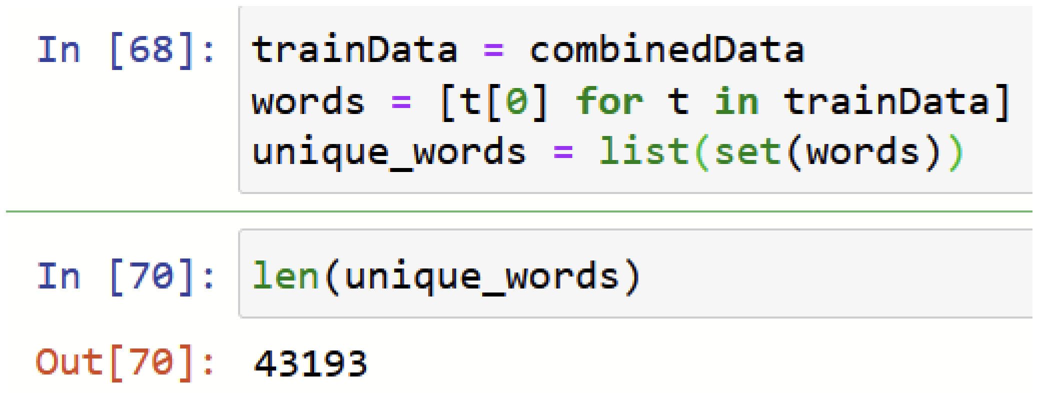
HW2

2995359247
Kevin Lee

Task 1
To create the vocabulary, I first got rid of any tags that wasn't part of the UPenn Tagset. Although there wasn't much in the dataset, this part was necessary as the probability will utilize default instead of unk probability. Here I also played around with the threshold, seeing what would be the threshold for unique words (such as Sally) to become unk. I've tried 0, 1, 3, 5, 10. The accuracy decreased with the 3 (~92%)/ 5 (~87%) /10 (~10%!!!) thresholds, and the best threshold I found is 1.

In total, there are 43193 unique words in the training dataset, and with the 46 tags (45 tags from UPenn and 1 tag for unk), having to go through each word and their tags to get the probability for emission and transition probabilities took the longest time. Thankfully, I found out that I could utilize Counter from Collections python package to speed this greatly.

I also pickled the probabilities (as *_6.obj, but there are other pickled variables if you want to check them out) so that the graders didn't need to face this pain.

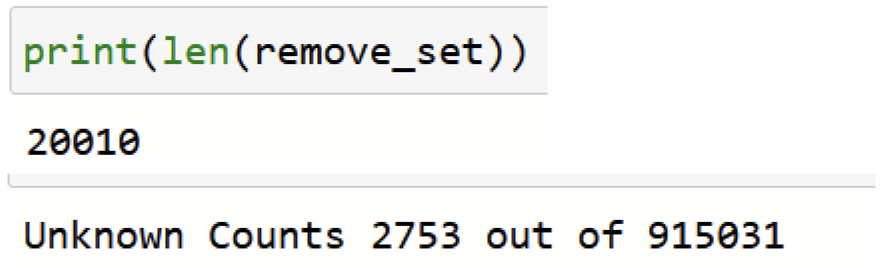


```
In [68]: trainData = combinedData
         words = [t[0] for t in trainData]
         unique_words = list(set(words))

In [70]: len(unique_words)

Out[70]: 43193
```

There were initially 2753 words that had a tag not part of the UPenn tagset, which I converted to unk. With the threshold of 1, there were in total 20010 words that were all unique. So in total, there were 22763 unique words that I converted tags to become unk.



```
print(len(remove_set))

20010

Unknown Counts 2753 out of 915031
```

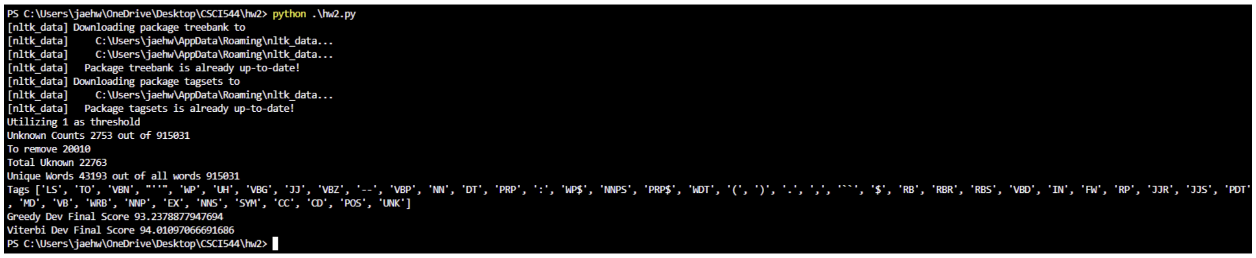
Task 2
The length of the emission probability is 43193 for all the unique words multiplied by the number of tags I was going through, giving it the size of 1986878 different floats

The length of the transitional probabilities is a lot more handleable, as we only had to utilize 46 * 46 tags.

For the HMM.json creation, since you can’t have tuple as key, I utilized string that appended first element of tuple and second element of tuple with “-”.

Task 3
Utilizing threshold 1, I got 93.24% accuracy. But with threshold 3, I got 92.02%, and with threshold 5, I got 84.02% accuracy. But with threshold 0, I got 92.01% accuracy!

Task 4
Utilizing threshold 1, I got 94.01%.Viterbi was more sensity to the threshold, as having threshold 5 got me 64.996% accuracy, a lot worse than Greedy.



```
PS C:\Users\jaehw\OneDrive\Desktop\CSCI544\hw2> python .\hw2.py
[nltk_data] Downloading package treebank to
[nltk_data]     C:\Users\jaehw\AppData\Roaming\nltk_data...
[nltk_data]     C:\Users\jaehw\AppData\Roaming\nltk_data...
[nltk_data]   Package treebank is already up-to-date!
[nltk_data] Downloading package tagsets to
[nltk_data]     C:\Users\jaehw\AppData\Roaming\nltk_data...
[nltk_data]   Package tagsets is already up-to-date!
Utilizing 1 as threshold
Unknown Counts 2753 out of 915031
To remove 20010
Total Uknown 22763
Unique Words 43193 out of all words 915031
Tags ['LS', 'TO', 'VBN', "''", 'WP', 'UH', 'VBG', 'JJ', 'VBZ', '--', 'VBP', 'NN', 'DT', 'PRP', ':', 'WP$', 'NNPS', 'PRP$', 'WDT', '(', ')', '.', ',', '``', '$', 'RB', 'RBR', 'RBS', 'VBD', 'IN', 'FW', 'RP', 'JJR', 'JJS', 'PDT'
, 'MD', 'VB', 'WRB', 'NNP', 'EX', 'NNS', 'SYM', 'CC', 'CD', 'POS', 'UNK']
Greedy Dev Final Score 93.2378877947694
Viterbi Dev Final Score 94.01097066691686
PS C:\Users\jaehw\OneDrive\Desktop\CSCI544\hw2>
```