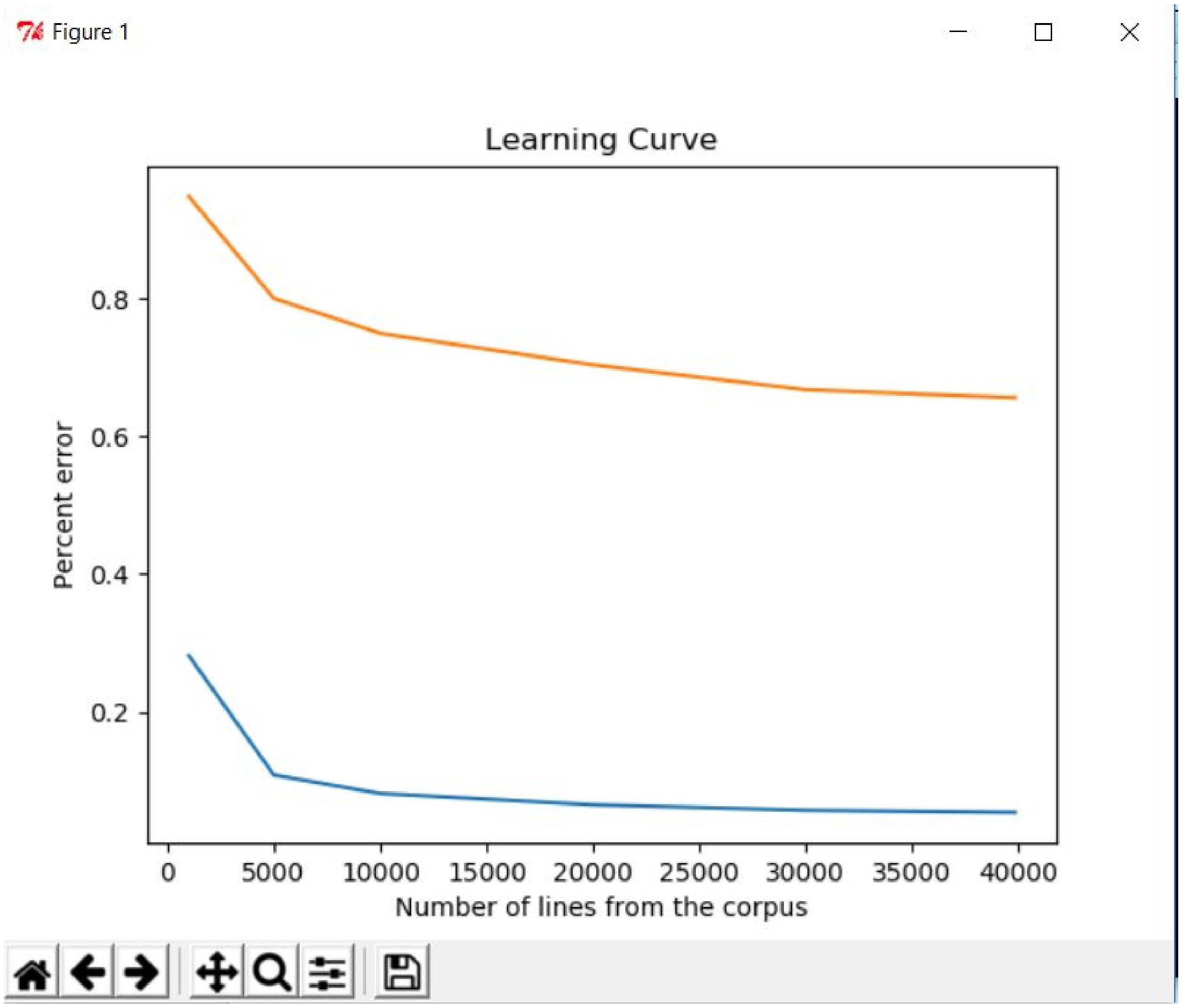
Sergei Chestakov
913914694
ECS 189G

Natural Language Processing HW #2

Part 1:

For the first part, I trained the model on the first 1000, 5000, 10000, 20000, and 30000 lines in the corpus as well as the entire corpus (39832 lines). I then aggregated all the data into one file (***results.txt***) and wrote a script (***plot.py***) to plot the error rate by word and sentence for each of the subsets using matplotlib. Below is the graph:



As you can see, the larger corpus always led to a more accurate model both for sentence error (orange line) and word error (blue line). It did begin to plateau as it approached the size of the corpus, however. In general, more POS tagged data would increase the accuracy of our model, especially if it were varied, but at a certain point it will plateau and benefits would be negligible.

Part 2:

Implementing the trigram HMM involved modifying the train_hmm file by changing the transitions dictionary to be 3 dimensional and the transitions_total dictionary to be 2 dimensional in order to remember transitions 2 words behind instead of just one like in bigrams. My modified version is in ***train_trigram.py***. The grunt of the work, however, was in modifying the viterbi algorithm to support the new trigram hmm. Again, the transitions, probability, and backpointer dictionaries would increase to 3 dimensions to account for two previous words. I also had to add 2 initial states before the first token, and update previous two states in training. Finally, the triple for loop through the set of tags caused it to run significantly slower. My implementation of the trigram viterbi is in ***viterbi.py***. My viterbi program outputs to ***my.out*** which achieved a 63% error rate by sentence and an 8.5% error rate by word (3407 errors out of 40117) which is a noticeable improvement over the original 65% error rate by sentence (1072 errors out of 1700). However, the error rate by word is slightly worse than the previous 5% (2170 errors out of 40117). I tried to implement deleted interpolation, but I found it actually decreased my accuracy for some reason, so I left it as is. I ran my model on the ptb.23.txt and outputed to ***PTB23.out***.

Part 3:

After training my trigram hmm on Bulgarian and Japanese, I found that the trigram model actually performed slightly worse for both. For Bulgarian I got 21.4% error rate by word and 78.6% error rate by sentence compared to 11.5% and 75.1% respectively for the bigram model. Likewise, for Japanese I received 11.3% error rate by word and 23.5% error rate by sentence compared to 6.2% and 13.6% respectively for the bigram. This decrease in accuracy for the trigram hmm model compared to the bigram for Japanese and Bulgarian seems to suggest that there is less long term dependencies in these languages and the previous tag is much more important than the previous two in dictating the current tag. My .hmm and .out files for both languages and both models are in this repository. It is interesting that for both the bigram and trigram models, Japanese had a smaller error rate than English, which had a smaller error rate than Bulgarian. This suggests a difference in predictability between the languages.