CSE5525 Lab 2: HMMs

Matias Grioni, Joshua Lan, Jayavardhan Peddamail

**Mandatory Part**

Using the most likely tag with the probability counts from the data, we achieved a word level accuracy of 92.5%. We used an epsilon probability of .000001 for unknown words belonging to the most frequent tag. For sentences where the alpha values went to zero prior to the end symbol (the probability of the observations given the model was 0), we took an arbitrary path through the rest of the tags.

Script to get word level accuracy with most probable word counts (takes around 20 minutes).

python3 run-trained-viterbi.py

**Extension 1**

After training with the forward backward implementation we were only able to achieve an accuracy of 50% after 5 iterations. We were not able to train the HMM until convergence as it was extremely slow, taking around 2 hours per EM step. Checking our algorithm against the Eisner spreadsheet, we were able to get the same results after 10 iterations as were displayed in the Eisner spreadsheet. This can be seen from the test\_eisner.py script. We were not able to determine why our implementation was so slow. Initially, we had been doing an E and M step for each observation sequence in the training data set, which was much too slow (~4 hours per epoch). We spoke with Dr. Fosler-Lussier and he told us to only perform the E step for each observation sequence and do a single M step for the entire dataset. After making several performance improvements by changing loops to store values on computation and using sums to use only one M step, the implementation was still too slow to run until convergence.

Script for training and eval (warning this may not terminate for several hours).

python3 train-test-model.py

Script for checking against Eisner spreadsheet (uses default Eisner values).

python3 test-eisner.py

**Extension 3**

Using lowercase words for training and test resulted in a word level accuracy of 91.5%, slightly lower than the model in the mandatory part. This may be due to the fact that the case of the words was important for determining the tag correctly. Things like proper nouns would be mistaken if the case was lost, i.e. the name Rich vs the adjective rich.

Script for using case insensitive data (takes around 20 minutes).

python3 run-case-insensitive.py