Hidden Markov Models

Yang Liu

Training Part:

In this part, I used four collections.Counter to count the appearance of each states and observations in order to compute probabilities. One for observation o and state j, one for state j, one for transition i to j, and one for first state in each sentence. These four sets of numbers give enough parameters to compute each probability.

Modify in test_hmm.py:

Since the ice-cream part's training (only need to transit the parameters) is different the tag part, so I implemented another method to deal with the ice-cream problem. So in the test_hmm.py file, the training method in test of the ice-cream is changed to train_ice_cream.
And also I added a test to compute the likelihood of a give test set.

Unknown Word:

If a word appeared in training data when classifying, I will add it to the codebook and thus have an index point at this word that could be used to smooth its probability.

Smoothing:

For each emission probability, I added 1 to the numerator and 2 to the denominator. If a combination of observed data and state is not in training data, I will pretend it appeared once. It is similar for transition and initial probabilities. So when classifying, if there is some words that not appeared in the training data, I will pretended it appeared once same as before, but only for the emission probability because only this one need the observed data. What's more, for each new word when classifying, there must be a "for loop" to ensure that emission probability is added for each state.

Experiments & Performance:

1. Original
   With train rate 0.95 and 3500 words, the classifier gives a accuracy of 92% with in 12 seconds.
2. Collapse the tag-set space
   When only remains the first letter of each tag, the accuracy became 93% and the time consumed reduced to 6.1 seconds.
   And when we do not lower each word, the accuracy became 93%.
3. Size of the corpus and the train/test split

rate	words	accurucy(%)
0.95	4000	92
0.9	4000	91
0.95	3500	92
0.85	3500	90
0.95	3000	90
0.95	2500	89
0.9	2500	88
0.95	2000	89
0.9	2000	87
0.95	1500	89

Unsupervised HMM:

I implemented the unsupervised HMM in class UnsupervisedHMM. I give the equal probabilities as initial, and ones as final transmission probabilities (a_iF). After testing, the parameters are correctly computed (such as alpha, beta and gamma), but the numbers were too small so that, in the NLTK data, some of the beta will become zero. And I implemented the E-step and M-step as it said in the textbook.