Attached file: hw3.ipynb

I read the training data set as pairs (word, tag) from each line of the Train.txt file and put it in the train_tagged list. I will calculate the parameters of the training model on this list.

The first step is to separate the tags and know what tags are used in the training data. The total set of tags has 23 elements and includes the following:

```
tags = {V, AR, DET, PRO, MS, MQUA, P, OH, INT, DEFAULT,
PS, CON, IF, DELM, ADJ, OHH, SPEC, QUA, NP, ADV, MORP,
N, PP }
```

For the Viterbi algorithm, two calculations are needed from the training data. One is the probability of two given singles in a row and the other is the probability of one word per given tag.

For the first calculation, it means p(tag1|tag2) for each tag1, tag2 that is a member of tags. These probabilities are used in every step of the Viterbi algorithm regardless of the current and previous labels, but their value is determined based on the training data and does not change during the work. These values can be stored in a 23x23 matrix by counting all consecutive labels in training pairs. The calculations of the matrices of this matrix are done in the p_t2_given_t1 function. I create the created matrix as a pandas.DataFrame called tags_df.

For the second calculation, I need to calculate p(word|tag) based on the words of the test set and the tags on it in each step of Viterbi. Before running the algorithm, these values are calculated for each word and tag pair in the test set from the training set with the word_given_tag function. This function counts two values on the training data, the occurrences of the given word-label pair and the occurrences of the label alone. This information is stored on a pandas.DataFrame called test_words_df.

The production of the two matrices above are actually part of the model training. For a hidden Markov model, the two matrices above are actually matrix A and part of matrix B. The production of these two or the counting of pairs on the set takes the most time of the execution of this code.

The Viterbi algorithm implemented in the Viterbi function calculates a path step for each word of the training set. It selects and stores more possibilities until it reaches the end of the path (end of words). Then it selects and outputs the path that gives the highest probability. This sequence of states is actually the result of tagging input words.

I read the test data line by line from the Test.txt file in the form of two lists. One is test_untagged, which contains the sequence of untagged test words. The second one is test_tagged, which is similar to the list of training data composed of word-tag pairs.

I give test_untagged as input to Viterbi. It takes some time to generate the sequence of output tags and they are finally stored in tagged_sequence. To compare the real tags with the

generated tags, I compare the two lists tagged_sequence and test_tagged and count the correct results. The accuracy criterion obtained on all test data is 82%.