Note: I didn't use command line in my code. So just run it directly, because I've set all the parameters in the code. If you want to change the parameters, please read the following instructions.

Also, in this question you only need to run Inference.py, because the other python files are used to generate intermediate variables (emission and transition probabilities) or the implementation of some functions called by Inference part. It may take a while to run the code, so you can also take a look at the results in 'Output_result', which I ran in advance.

Files:
q4_UD_English: Original datasets
test.tags, test.words, train.counts: from q4_UD_English
Output_result: Output tags using Viterbi algorithm, same format as test.tags (<word> <tag>)

There are three python files:

1. **HMM.py**: Read training data, and compute emission probability and transition probability.

a. Parameters: The parameter of this program is the path of training data, between line 8 to line 10.

b. Funstion process: First I read data from 'train.counts', and then change infrequent words to 'UNK' (here I replace words less than 2 times to be 'UNK'). After that, I compute emission probability and transition probability using function Emission_prob() and function Transition_prob().

   Specifically, the function observations_process() is to replace the UNK words in test data(here is observations).

c. Run the code: The only parameter in this program is the path of training data (line 8 – line 10) as I said. You can run the code directly without making any changes. The output of the program is the emission probability and transition probability which are used in Inference.py. But I print the first 10 elements in the program.

   Also, I print some process so that you can know which step it is during the running.

2. **viterbi.py**: Viterbi algorithm, which will return the POS tags of a given sentence. I use Wikipedia implementation as reference in this part.

a. Parameters:
   states: tuple of all tags
   start_p: dictionary of start probability

trans_p: transition probability, with format of dictionary
emit_p: emission probability, with format of dictionary

b. Function process: return a list of states(POS tags) of the given sentence.

3. **Inference.py**: Do inference on test data, and write the results in the file.

a. Parameters: The path of test set (line 10-12), and the output file name (line 53).

b. Function process: In this program, I firstly read data from test set, and call the observation_process() function to replace 'UNK' in the test data. After that, I use Viterbi algorithm do the inference and write the output in the file.

c. Run the code: I've set the parameters so that you can run this code directly, and when running finish, a new file will be created with the output. Also, you can just open 'Output_result' file to take a look at results which I ran, cause it may take a little long time to run the code.