Answers for the Questions:

1. **What is the selected threshold for unknown words replacement?**

Ans. 2

2. **What is the total size of your vocabulary?**

Ans. 16920

3. **What is the total occurrences of the special token '<unk>' after replacement?**

Ans. 32537

4. **How many transition and emission parameters in your HMM?**

Ans. Transition Parameters: 2070

Emission Parameters: 761400

5. **What is the Accuracy on the dev data for Greedy Decoding?**

Ans. 92.98904752146093 %

6. **What is the Accuracy on the dev data for Viterbi Decoding?**

Ans. 94.37120021859416 %

Brief Explanation of the solution:

Task 1: Vocabulary Creation

- For the vocabulary creation task we first read the train dataset file and create a dataframe of it and replace all words with unknown tag for which the frequency of the particular word is below the threshold
- Secondly we use the frequency as the parameter to sort the words in the dataframe and then we extract all the unique words in the ascending order of the dataframe and then export it to the csv file having columns 'name', position of the word, 'frequency'

Task 2: Model Learning

- In this task we create transition matrix and emmission matrix and after creating both the matrices we then convert the matrix to a python dictionary for faster retrieval of data.
- transition matrix is of size: length of all_tags * length of all_tags
- row represents the tag at the previous state and the column represents the tag at the current state for which we want to calculate the transition probability
- Emmission matrix is of size: length of all_tags * length of vocabualry
- row represents the current tag at the previous state and the column represents the vocab word for which we want to calculate the emmission probability
- For the transition dictionary each key represents a tuple in which first value is the previous tag and the second value of the tuple is the current tag and the value of the key represents the transition probability from the previous tag to the current tag
- Likewise for the emmission dictionary each key represents a tuple in which first value is the current tag and the second value of the tuple is the vocab word from which the current tag is pointed to, and the value of the key represents the emmission probability from the current tag to the current vocab word

Task 3: Greedy Decoding with HMM

- In greedy decoding for each transition in the states we calculate the net probability score till the current tag every time and store the tag which is giving maximum score and the stored tag will help us get the final sequence and accuracy of the greedy decoding algorithm.
- To handle a situation where a vocab word is not present in the emmission matrix then we use the probability for the unknown words considering the not found word as unknown

Task 4: Viterbi Decoding with HMM

- For Viterbi Decoding , we maintain a list/array representing all the scores for the previous state/pos tags.
- We maintain a Memo or cache memory as a dictionary for storing all indices and pos tags as the key and value as the score or the probability, the cache will continue updating until we find a better transition mapping from one tag and to a vocab word.
- We calculated all the best scores for each sentence and stored it to a memo/cache using viterbi decoding, now we propagate backward to find the best pos tag sequence for the corresponding sentence.
- While propagating backwords we maintain the tag which gives the max score until now and find the next best tag (using the cache/memo) at the previous position, thats how we find the final sequence for that sentence.