CSCI 544 – Applied Natural Language Processing

Homework 2 – Vu Truong Si – 6031936649

Task 1: Vocabulary Creation

In order to create the vocabulary for this task, I used a dictionary and counted the occurrences of each word. I selected **3** for the threshold for unknown words replacement, so words that occurred less than 3 times will be encoded as "< unk >". Next, I sorted the data by the counts of each word in descending order.

The size of the vocabulary is **16920**. After replacement, the special token "< unk >" has a count of **32537**.

Task 2: Model Learning

To calculate transition probabilities, I counted the number of times one tag transited into another and also the occurrences of those tags, then applied the formula count(s->s') / count(s). If a word is the first word of a sentence, s = "< start >".

To calculate emission probabilities, I counted the number of times one tag produced a word and also the count of those tags, then applied the formula count(s->x) / count(s).

After creating the transition and emission dictionaries, I have **1392** transition parameters and **23373** emission parameters.

Task 3: Greedy decoding with HMM

To implement this algorithm. I first looped through the dataset and looked at each word. If that is the first word of the sentence, I used < start > as the conditional tag and calculated the transition probabilities for (<start> -> all tags) * (all tags -> word) and got the tag with the maximum probability out. If it is not the first word, I used the previous tag as the conditional tag and calculated (<previous tag> -> all tags) * (all tags -> word) and got the tag with the maximum probability out.

After the loop, I got a list of predicted tags that correspond to the words.

The accuracy of the Greedy algorithm on the dev set is: **0.9295124764738024**

Task 4: Viterbi decoding with HMM

For Viterbi, I created a dictionary (pi) and used it as a dynamic array to store the probabilities. I looped through the dataset and looked at each word, if that is the first word of the sentence, I calculated pi(0,s) for all s in the tags set. If it is not the first word, I used the previous rows' results to calculate the maximum probabilities for each tag, then took the tag with the highest probability.

After the loop, I also got a list of predicted tags that correspond to the words.

The accuracy of the Viterbi algorithm on the dev set is: **0.9312200230708518**