CSCI 544 – Homework 2 – Gautami Langarkande

What is the selected threshold for unknown word replacement?
Ans: 3

What is the total size of your vocabulary?
Ans: 16920

what are the total occurrences of the special token '< unk >' after replacement?
Ans: 32537

How many transition and emission parameters are in your HMM?
Ans: 45 x 45 transition parameters
45 × 16920 emission parameters
Number of non-zero transition probabilities: 1378
Number of non-zero emission probabilities: 23373

What is the accuracy of the dev data (for greedy)?
Ans: 93.08%

What is the accuracy of the dev data? (for Viterbi)
Ans: 94.28%

Task 1: Vocabulary Creation:

> For this task, I first loaded the dataset into a data frame called df. Then I check the value_counts of the unique words in the data frame. I have set the threshold as three. I replaced the words with threshold less than three as "<unk>" . I have not performed any preprocessing on the dataset. Then I stored this data frame in the desired format as a text file called "vocab.txt"

What is the selected threshold for unknown word replacement?
Ans: 3

What is the total size of your vocabulary?
Ans: 16920

What are the total occurrences of the special token '< unk >' after replacement?
Ans: 32537

Task 2: HMM Model training

I calculated the transition probability using the following formula:

t(s'|s)=count(s->s')/count(s)

and then converted it into a matrix and saved these values in a data frame called transition_df.

The emission probabilities were also calculated using the following formula:

e(x|s)=count(s->x)/count(s)

and just like transition probabilities, this was converted into a matrix and stored in a data frame called emission_df

How many transition and emission parameters are in your HMM?

Ans: 45 x 45 = 2,025 transition parameters and 45 × 16920 = 761,400 emission parameters

Task 3: Greedy Decoding Model

I created a list of initial probabilities using the following formula:

Number of POS tags in the data frame/length of the data frame

For this task, first, I imported the dev data and saved this in the correct format for processing. I iterate over the words in the dev dataset. For calculating the probability of the first word, I used the initial probability and the transition probability, and I multiplied them. Then I consider the POS tag, which gives the maximum probability out of these products. For the words after the first word in the sentence, I use the product transmission probability and the emission probability and then find the max probability compared to the after multiplying it with the probability of the previous word, which continues till a full stop is encountered. The final output is all predicted POS tags of each word in list form.

What is the accuracy of the dev data (for greedy)?

Ans: 93.08%

Task 4: Viterbi Algorithm

This algorithm has a more complicated approach. The logic to find the POS tag for the first word of the sentence is the same as the greedy approach. For the words following the first word, I stored the prior probabilities in a list and kept track of the current probabilities; I also have maintained a list with the "nodes" or POS tags of the previous word; I update the

probabilities as I keep iterating over the next words in the sentence. Once we encounter a full stop, we trace back to check the max probability we get from multiplying all the probabilities of each POS tag of each word from that sentence, and we store this list of best predicted POS tags. The process restarts after the full stop. The final list consists of all the POS tags in the data frame. Then the tags were generated for the test data set

What is the accuracy of the dev data? (for Viterbi)

Ans: 94.28%