CSCI544: Homework Assignment No 2

Nehal Muthukumar
1677301672

Task 1: Vocabulary Creation

Threshold for unknown counts: 2
Vocabulary size: 23183
Unknown words count after replacement: 20011

1) The training data is parsed to obtain a list of words along with their corresponding tags.
2) The frequency of each word is counted and stored in a dictionary.
3) A vocabulary list is created by retaining only the words that have a count greater than or equal to the specified threshold (set to 2).
4) Words with a count less than the threshold are considered unknown and are mapped to the special ID <unk>.
5) Finally, the words are written to the vocab.txt file in sorted order.

Task 2: Model Learning

Number of emission parameters: 30303
Number of transition parameters: 1416 (included the starting and ending probabilities of a tag)

1) The code parses the training file line by line to obtain the emission and transmission count.
2) Transition count is obtained by counting all the (prev_state, state) in the train data. Emission count is obtained by counting all the (state, word) in the train data.
3) If a word is not present in the vocabulary, it is mapped to the "<unk>" token.
4) The "STARTING" and "ENDING" transition probabilities for all the tags in the tag set are also included, i.e., ("STARTING", TAG) and (TAG, "ENDING") for all TAG.
5) The transition and emission parameters are calculated as the ratio of the count of each event to the count of its corresponding state, as specified in the HW document.
6) Finally, the transition and emission probabilities are stored as dictionaries and saved to a JSON file.

Task 3: Greedy Decoding with HMM (30 points)

Accuracy on dev data : 0.9330945297796126
Prediction for test data is generated and stored as in train data. -greedy.out

1) The process of parsing the dev data line by line is carried out to obtain the words.
2) If a word is not present in the vocabulary, it is mapped to the token <unk>.
3) The tag with the highest transition and emission probability is determined and assigned to each word (in each step).
4) If the key is not found in the transmission or emission probabilities, a default value of 1e-7 is returned.
5) The accuracy is determined by comparing the predicted tags with the actual tags.

6) Finally, this procedure is repeated on the test data, resulting in the generation of the greedy.out file.

Task 4: Viterbi Decoding with HMM

Accuracy on dev data : 0.9475449901708526
Prediction for test data is generated and stored as in train data. - viterbi.out

.

1) Two dictionaries are initialized: "pi" to store the maximum probability of a tag sequence ending in a given tag at a given position, and "bp" to store the most likely previous tag for each tag at each position.
2) For each word in the input sequence and each tag in the tag set, the maximum probability of a tag sequence ending in the current tag is computed. For the first word, this is done by multiplying the transition probability from the starting state to the current tag and the emission probability of observing the current word given the current tag. The resulting value is stored in the "pi" dictionary.
3) For subsequent words, the maximum probability of a tag sequence ending in the current tag is computed as the product of the maximum probability of a tag sequence ending in each possible previous tag, the transition probability from the previous tag to the current tag, and the emission probability of observing the current word given the current tag. This value is stored in the "pi" dictionary, and the most likely previous tag is stored in the "bp" dictionary.
4) Once all words have been processed, the maximum probability of a tag sequence ending in a stopping state is computed by multiplying the maximum probability of a tag sequence ending in each tag and the transition probability from the tag to the stopping state. The tag with the maximum probability is stored in "max_end_tag".
5) The most likely tag sequence is then determined by tracing back from "max_end_tag" using the backpointers stored in the "bp" dictionary. The resulting tag sequence is returned as the output of the function.
6) Accuracy is determined by comparing the predicted tags to the actual tags. This procedure is repeated on the test data to generate an output file.

NOTE:
To handle the issue of vanishing 0s when processing long sequences of words, two methods were tried: adding log probabilities instead of multiplying probabilities and processing pos tagging sentence by sentence. The second method was found to produce more accurate results on dev, so it was adopted for prediction on the test.