ELL884 Assignment 1

M Yagnesh
2023AIB2069

HMM (Hidden Markov Model):

In this part, I have used HMM for POS Tagging using PENN Dataset as the training data. The HMM code I have implemented is having HMMPOSTagger class which has 2 parts, one is for training the model and other is for generating the POS tags.

In training, the PENN dataset from the file(train.csv) has 2 columns in each row (one is untagged, and the other is tagged). Tagged sentences are used to get word-tag from the sentence and then calculate the emission probabilities and transition probabilities with the following equations.

$$\text{Transition\_probability } T_{i,j} = \frac{\#\ of occurances\ of\ tag\ i\ followed\ by\ tag\ j}{\#\ of\ occurances\ of\ i} = P\ (t_j \mid t_i)$$

$$\text{Emsission\_probability } E_{w,j} = \frac{\#\ of\ occurances\ of\ word\ i\ with\ tag\ j}{\#\ of\ occurances\ of\ word\ i} = P\ (w_j \mid t_j)$$

In assigning tags for the test data, I calculated the Viterbi matrix using emission and transition probabilities and maintained a backpointer matrix. Then I have backtrackted from the last column (word) of viterbi matrix and assigned tag with highest probability to the words in sentence.

$$\text{Viterbi } V_{iw} = \text{Transition}_{j,i} * \max(\text{Viterbi}_{pw}) * \text{Emission}_{wi}$$

j = prev_tag,i = current tag, w =word, pw = prev word

For each cell in the viterbi matrix, I have calculated the transition prob from all the prev tags, prev word viterbi values and multiplied them respectively which will generate a matrix of total count unique tags. Then from them I got the tag with maximum position and multiplied it with the emission of that tag for that word and stored it in the viterbi matrix. This helps in significantly reducing the time. In the backpointer matrix, I am storing the previous tags for which the present tag of the word has the highest probability prev tag so that it can be baktracked and then the tags can be assigned to the words.

After filling in the viterbi matrix, I iterated backwards to find the best tags. I have first got the position of the maximum value in the last column and then added it to the tag list and then checked the tag corresponding to the cell in the backpointer matrix and added it to tag_list and then

continued this until the first column in the viterbi matrix. Then I have just reversed the tag_list which has best tags for the corresponding words in the sentence.
There were few issues when assigning the tags to words in the test data.

- Assigning tags to words which were not present in the training data.

To solve this issue, I have counted the tag occurrences in the training data and tag having the highest count is given more probability when I have created an emission matrix for it. It works as most occurred tags might have a more chance of occurring again and hence is given majority probability.

- Dealing with values in transition probability 0

Transition from one tag to another might not have occurred in the training set, but the corresponding words might have more emission probabilities with that tag. To overcome this, the probabilities are smoothened, and a small value is added to transitions with 0 probability. It significantly improved the accuracy of the tags.

MEMM (Maximum-entropy Markov model):

In MEMM I have implemented the feature of the transitioning of a word from the previous tag. The main idea is to use the implementation of the probability of a word being assigned a tag given the prev tag and tried to use it. But the accuracies were worse.

The training and tagging steps in the MEMM are similar in HMM and instead of using transition matrix and emission matrix, I have implemented a probability matrix which can store the transition of a word given its previous tag.

Conclusion:

In theory, even though MEMM can perform better when compared to HMM, finding the right features for the MEMM can be difficult for POS tagging.

HMM: HMM Kaggle link | HMM Colab Link
MEMM: MEMM Kaggle Link | MEMM Colab Link