Machine Learning Project Report

Xiong Yang 1004876
Wang Xilun 1004877
Gao Fancheng 1004879

April 21, 2023

1 Part 1 and 2

We managed to pack the functions and variables used for HMM into a HMM class. It takes paths of train data and test data as initialization parameters. Then the initializer reads data from the path and calculate emission matrix and transition matrix from the train set. Then we predict the label series using the built in function `eval_1` and `eval_2`.

2 Part 3

We failed to include part 3 in the newly created HMM class for part 1 and part 2. So we directly copied the original code from the notebook for developing. That's why it looks so strange, messy and different from part12.py.

For the Viterbi Algorithm used here, `pi[i][v][u]` represents the best score of position `i` when its label is `v` and its previous label is `u`. We give a min value to the `log_p` in the calculation to guarantee that there would be possible path to be found.

3 Part 4

For Part 4, we used Structured Perception. The Structured Perceptron is a linear sequence labeling algorithm. The Structured Perceptron can learn a linear boundary between multilple possible classes, where each class represents a possible output sequence. The Structured Perceptron algorithm works by iteratively updating a weight vector based on training examples. At each iteration, the algorithm makes a prediction for each training example and compares it to the true output. If the prediction is incorrect, the weight vector is updated to increase the score of the correct output sequence and decrease the score of the predicted output sequence. The score of an output sequence Y is defined as the dot product of the weight vector W and a feature vector $\Phi(X, Y)$, which represents the input sequence X and the output sequence Y.

$$score(x, y) = \sum_{i=1}^{T} w \cdot \phi(y, x)$$

The feature vector $\Phi(X, Y)$ is a vector of binary features that capture the relationship between the input sequence X and the output sequence Y. In our case, we included the information of two words before the current visited word and two words after the current visited word together with the current visited word to generate the features for each word in the input sentence. We set the initial weight to 0 and if the predicted Y' doesn't match with the true label Y, we penalize the weight of Y' by 1, and increase the weight of Y by 1.

$$w = w + \phi(x, y) - \phi(x, \hat{y}) \tag{1}$$

$$\phi_(y_t, x_t) = \begin{cases} 1 \\ 0 \end{cases} \tag{2}$$

To make a prediction for the output sequence Y', the algorithm selects the output sequence with the highest score.

$$\hat{y} = \arg\max_{y' \in Y} [w \cdot \phi(x, y')]$$

Since the weight needs to be updated after visiting each sentence in the input data X, we rewrite the functions for generating the transition matrix and emission matrix so that they generate these matrices from the updated weight value for the sentence we are visiting. We also rewrite the Viterbi function so that it generates predictions for each input sentence. When updating the weight, we iterate through each position in the input sequence x and its corresponding output label y. For each feature of the input at position i, update the weight vector for label y[i]. Then we iterate through each adjacent pair of labels y[i-1]:y[i] and update the weight vector for the transition between them. Compared with HMM model, the Structured Perception model allows for more flexibility in the choice of features and feature representations, as it can use any kind of features that can be extracted from the input sequence and the output sequence. In contrast, the HMM model is based on a fixed set of probabilistic models that define the emission and transition probabilities, and the features used by the HMM are typically hand-crafted and domain-specific

To notice, the Structured Perception has a high accuracy of 92%. But it misclassified some `B` and `I` words to `O` so it got a low sentiment F because `B`s and `I`s are rare in `FR` dataset. Though still better then HMM.

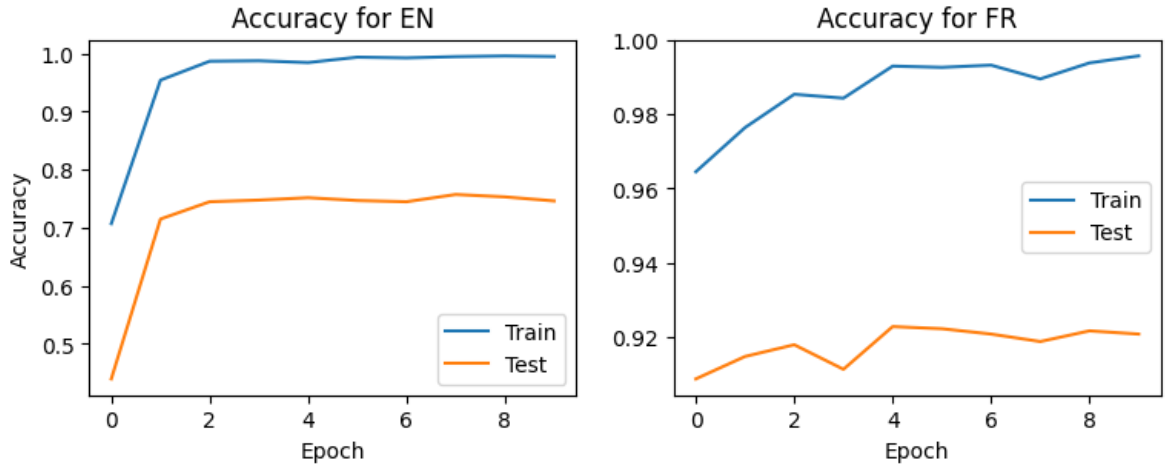

Figure 1: Training Result