Digital Speech Processing HW1

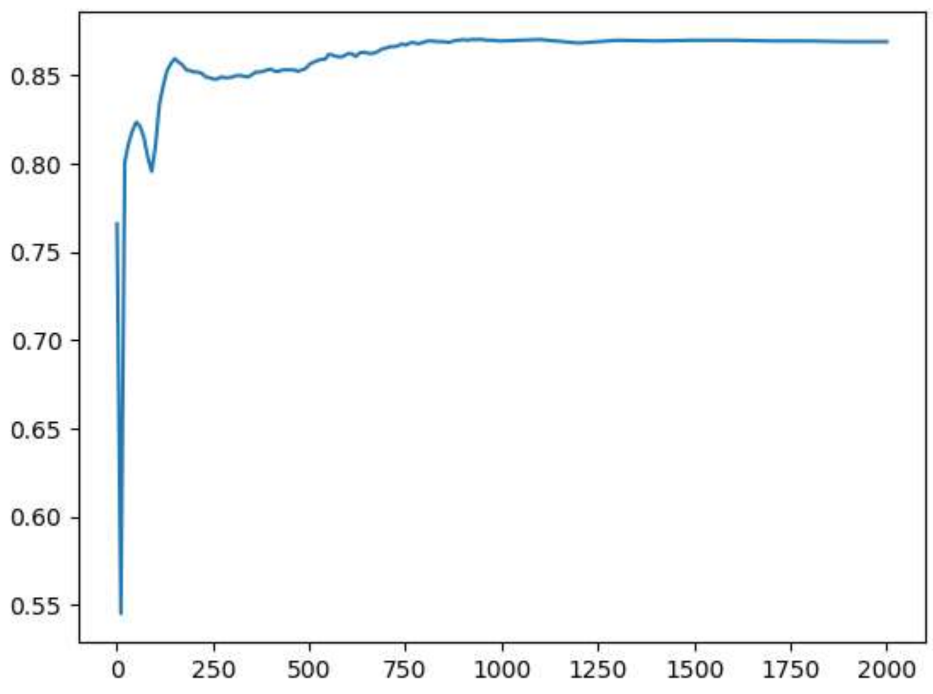
資工三　　沙佳哲　B05902030

1. Environment:
   Written on Windows 10, but also tested on CSIE workstation.

2. How to execute:
   Type make to compile.
   ./train [Iteration] [InitModelPath] [TrainDataPath] [OutputModelPath]
   ./test [ModelListPath] [TestDataPath] [OutputResultPath]

3. Summary:
   Training:
   First load in the initial model, then start training.
   In every training iteration, I read in the data line by line, then calculate alpha and beta with the HMM model and current training data, and then calculate and accumulate gamma and epsilon with the HMM model, alpha, beta and the current training data. After reading all data from the training data(when reached EOF), update the HMM model with the accumulated gamma and epsilon.
   Finally, after all iterations done, save the model.
   *Calculations.h is for the calculation of alpha, beta, gamma, epsilon and updating
   *utils.h contains a printModel function, and a State2Num function.
   Testing:
   First load the models on modellist, then test every data in testing data with Viterbi algorithm and save the best model with the highest probability and also it's probalility.
   *Viterbi.h contains the calculation of the Viterbi algorithm.

Results:



I first tested for 2000 iterations, and saved the model every 100 iterations, found out that the accuracy didn't improve at all after around 800 iterations, then I tested for 1000 iterations, and saved the model every 10 iterations to have a better resolution. I think the model might have memorized the training data in some sense, and this might actually cause overfitting at big iterations.

4. Others:
   Since the algorithm for training and testing are basically determined, everyone gets the same result with the same iteration if done exactly the same as the lecture, I wonder if there are some randomized algorithm that creates some diversity in model other than simply randomizing the data with techniques like bagging.