# Hidden Markov models

Nikhil Jangamreddy (2018csm1011), Aakash Saini (2018csm1001)

May 7, 2019

## 1 Loading and Writing the model files:

To implement the model we have written class **HMM** which denotes the Hidden Markov Model. For this class transition and emission probabilities are used as the parameters which are specified in the files like models/two_english.trans and models/two_english.emit. In .emit file, each row is having  symbol which denotes the start state.

Function HMM.load is used to load the parameters from the model files in which some of the transitions may be omitted. Such parameters maintains value 0. If conditional probabilities are not specified in model files then constructor will initialize them randomly. Inverse method **HMM.dump** takes basename of output file model's parameter are written to the .trans and .emit files.

## 2 HMM Supervised learning :

For supervised learning method **HMM.learn_supervised** is defines which takes which takes observation's list with know state sequences from the file *browntags.obs*. This method also estimate the parameters of HMM using the maximum likelihood estimator.

## 3 Generating random observations :

We have defined method **HMM.generate()** which generates the sampling from the HMM. This function takes an integer $n$ as input parameter and returns a random observation having length n.


**Figure 1: Random 20 Observations**

# 4 Finding best State sequence using Viterbi algorithm:

For finding the best State sequence using Viterbi Algorithm, we have defined a method named **HMM.viterbi()** for implementing the Viterbi algorithm. This method assigns the observation's state sequence with Viterbi state sequence.

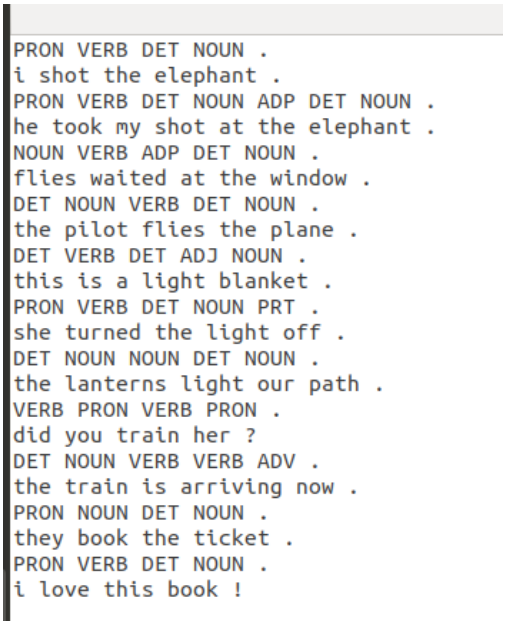



**Figure 2: Output file in Viterbi algorithm**

# 5 Implementing Forward and Backward Algorithms :

Forward and Backward algorithms are implemented in the functions **forward** and **backward()** respectively, both these methods take output sequence as input.

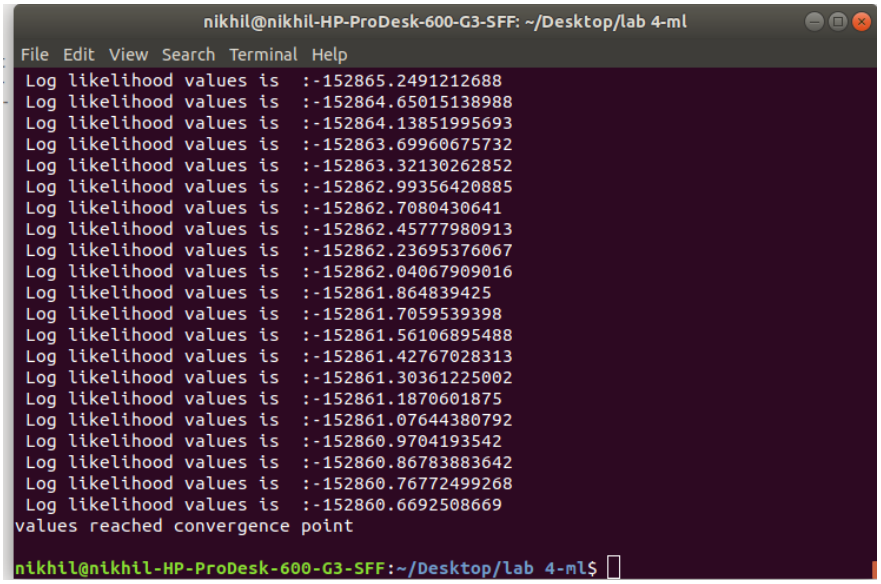
We have also defined the methods **forward_probability()** and **backward_probability()** for calculating the probability for a given output sequence using the values returned from *forward* and *backward* functions.

The probability estimated by the forward_probability function from the file ambiguous_sents.obs is written to the file ambiguous_sents.forwardprob.

The probability estimated by the backward_probability function from the file ambiguous_sents.backwardprob is written to the file ambiguous_sents.prob.

# 6 Supervised and Unsupervised Learning with Baum Welch(EM) algorithm:

Function **learn_unsupervised()** is defined for estimating the parameter of model from the beginning of current model and log-likelihood for the trained model. This method takes a list of of observations for which state sequence is unknown, convergence threshold, booleans for specification of transition and emission parameters, and the number of random restarts. Baum Welch EM algorithm is used by this algorithm by drawing upon the backward and forward algorithm.




Output of final log likelihood