# CS4248 Assignment 2

Wei Wenbo

A0105683E

## Introduction

In this assignment, I use a Java Program to perform part-of-speech (POS) bigram tagging with Penn Treebank tag set on a hidden Markov Model. Specifically, we are using Viterbi Algorithm, a dynamic algorithm to find the optimal sequence of most probable POS tags.

In particular, we are going to build a black box, where once the inputs, untagged sentences are thrown in, we will have output sentences properly tagged.

## Methodology

## Collecting training set data for statistical analysis

The training set we have been using is *sents.train,* as well as Penn Treebank tag set. For example, the information we want from this training data set is below:

|  |  |
| --- | --- |
| Terms | Sample structures |
| The count for the tags | {NNP=300,JJ=234, …, </s> =1296} |
| The probability of the transitions between tags | {  NNP={NNP=0.0001,JJ=0.0003,…},  JJ={NNP=0.009,JJ=0.0087,…},  …} |
| Vocabulary | {I, am, collecting, …} |
| The list of words and their counts under a certain tags | {NNP={Internet, US, POS, …}, …} |

## Using Viterbi Algorithm to make predictions

Based on the Hidden Markov Model, without dynamic programming, the time complexity should be O(T \* N^T), where T is number of words and N is number of POS tags.

With Viterbi Algorithm, we can reduce the time complexity to O(T \* N^2). Viterbi Algorithm will return a list of states that indicate the optimal prediction for one sentence.

## Written Bells Smoothing

In this assignment, we are using written-bells method to deal with unknown words and smoothing. In particular, we need to do smoothing for both tag-tag and tag-word.

For tag-tag smoothing:

Similarly, we have for tag-word smoothing

## Implementation Details

**3.1 A model for storing and reading statistical information**

We are using a serializable Java class Model to perform a role for storing and reading the statistical information of a training set. In particular, every time the user execute the command,

*java build\_tagger sents.train sents.dev model\_file*

we will store the information we have obtained from processing sents.train into model\_file.

And after that, if we execute the run\_tagger

java run\_tagger sents.test model\_file sents.out

we will read the model\_file which we have created previously, reform an object out of it and start to do the prediction process and save the results into sents.out.

**3.2 Using logarithm for Viterbi Algorithm to calculate the probability instead of multiplication**

The probability ranges from 0 and 1, and it causes some dangerous situations because the number is so small and it causes overflow.

Situation 1: the transition probability between states can be very small

Situation 2: the accumulated probability for a certain path is more likely to be very small, because of multi-multiplication of several decimals.

To avoid this problem, we are going to use logarithm. On one hand, it can solve the overflow problem; on the other, it makes the calculation much faster, since we are changing the multiplication into addition.

## Evaluation and Discussion

After running the program and comparing the results with standard answers, for the given test set, we can get the correction rate is more than 95%, with the help of HMM and written-bell smoothing.

And we also have some observations:

1. The program has difficulty in distinguishing VBD and VBN, especially when the VBD and VBN form of a verb is the same.

2. The program does not handle the proper noun well. One of the reason I thought it is because some of the proper noun are phrase, for example, Internet/Net/NNP, happens in the training set.