SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.007 Machine Learning, Spring 2025

# Design Project

Due 27 Apr 2025, 5:00pm

This project will be graded by Chengshun Wang
Please submit your work to eDimension.

Please form groups for this project early, and start this project early.

## Instructions

Please read the following instructions carefully before you start this project:

- This is a group project. You are allowed to form groups in any way you like, but each group must consist of either 2 or 3 people.
- You are strictly NOT allowed to use any external resources or machine learning packages. You will receive 0 for this project if you do so.
- Each group should submit code together with a report summarizing your work, and give clear instructions on how to run your code. Please also submit your system's outputs. Your output should be in the same column format as that of the training set.

## Project Summary

In an interview with Michael I. Jordan, the Pehong Chen Distinguished Professor in the Department of Electrical Engineering and Computer Science and the Department of Statistics at the University of California, Berkeley and a giant on Machine Learning, he was asked "If you got a billion dollars to spend on a huge research project that you get to lead, what would you like to do?". He answered: "I'd use the billion dollars to build a NASA-size program focusing on natural language processing".

Indeed, one of the most challenging problems within industry and academia is to design intelligent systems that are capable of comprehending human languages. Natural language processing (NLP) is regarded as one of the most challenging yet most promising directions within the field of artificial intelligence in the next 10 years. In this project, we will be working together on building some basic models for solving several simple NLP problems.

Many problems within the field of NLP are essentially structured prediction problems, among which sequence labeling is the simplest class of problems. The hidden Markov model (HMM) that we have learned in class is a simple structured prediction model. In this design project, we would like to design our sequence labelling model for informal texts using the HMM that we have learned in class. We hope that your sequence labelling system for informal texts can serve as the very first step towards building a more complex, intelligent language processing system. Specifically, we will focus on building an NLP systems – an *English phrase chunking* system.

The files for this project are in the files `EN.zip` (the English phrase chunking dataset). For each dataset, we provide a labelled training set `train`, an unlabelled development set `dev.in`, and a labelled development set `dev.out`. The labelled data has the format of one token per line with token and tag separated by tab and a single empty line that separates sentences.

The format for the dataset can be something like the following:

```
Cant B-VP
wait I-VP
for B-PP
the B-NP
ravens I-NP
game I-NP
tomorrow B-NP
.... O
go B-VP
ray B-NP
rice I-NP
!!!!!!! O
```

where labels such as `B-VP, I-VP` are used to indicate **B**eginning and the **I**nside of a **V**erb Phrase. Similarly for other tags such as `B-NP, I-NP` and `B-PP, I-PP`, which are used to indicate spans which are associated with noun phrases and propositional phrases etc. `O` is used to indicate the **O**utside of any phrase.

Specifically, in the above example, we have the following verb phrase chunks: "`Cant wait`", and "`go`", the following propositional phrase chunk: "`for`", and the following noun phrase chunks: "`the ravens game`", "`tomorrow`", and "`ray rice`".

Overall, our goal is to build a sequence labelling system from such training data and then use the system to predict tag sequences for new sentences. Specifically:

- We will be building one NLP system for English phrase chunking.

## 1 Part 1 *(30 points)*

Recall that the HMM discussed in class is defined as follows:

$$p(x_1, \ldots, x_n, y_1, \ldots, y_n) = \prod_{i=1}^{n+1} q(y_i|y_{i-1}) \cdot \prod_{i=1}^{n} e(x_i|y_i) \tag{1}$$

where $y_0 = \texttt{START}$ and $y_{n+1} = \texttt{STOP}$. Here $q$ are transition probabilities, and $e$ are emission parameters. In this project, $x$'s are the natural language words, and $y$'s are the tags (such as `O, B-NP`).

- Write a function that estimates the emission parameters from the training set using MLE (maximum likelihood estimation):

$$e(x|y) = \frac{\texttt{Count}(y \rightarrow x)}{\texttt{Count}(y)}$$

  *(10 points)*

- One problem with estimating the emission parameters is that some words that appear in the test set do not appear in the training set. One simple idea to handle this issue is as follows. First, replace those words that appear less than $k$ times in the training set with a special word token `#UNK#` before training. This leads to a "modified training set". We then use such a modified training set to train our model.

  During the testing phase, if a word does not appear in the "modified training set", we replace that word with `#UNK#` as well.

  This method is related to the idea called *smoothing*. The resulting emission parameters are called smoothed emission parameters. Let us assume such a smoothing method for the emission parameters is always used in the subsequent questions of this project (you may also do so in part 5 if you choose to).

  Set $k$ to 3, implement this fix into your function for computing the emission parameters.

  *(10 points)*

- Implement a simple system that produces the tag

  $$y^* = \arg\max_y e(x|y)$$

  for each word $x$ in the sequence.

  Learn these parameters with `train`, and evaluate your system on the development set `dev.in` for each of the dataset. Write your output to `dev.p2.out`. Compare your outputs and the gold-standard outputs in `dev.out` and report the precision, recall and F scores of such a baseline system.

  The precision score is defined as follows:

  $$\texttt{Precision} = \frac{\text{Total number of correctly predicted chunks}}{\text{Total number of predicted chunks}}$$

  The recall score is defined as follows:

  $$\texttt{Recall} = \frac{\text{Total number of correctly predicted chunks}}{\text{Total number of gold chunks}}$$

  where a gold entity is a true phrase chunk that is annotated in the reference output file, and a predicted phrase chunk is regarded as correct if and only if it matches exactly the gold phrase chunk (*i.e.*, both their *boundaries* and *chunk types* are exactly the same).

  Finally the F score is defined as follows:

  $$\texttt{F} = \frac{2}{1/\texttt{Precision} + 1/\texttt{Recall}}$$

  *Note: in some cases, you might have an output sequence that consists of a transition from* `O` *to* `I-NP` *(rather than* `B-NP`*). For example, "*`O I-NP I-NP O`*". In this case, the second and third words should be regarded as one chunk with type NP.*

  **You can use the evaluation script shared with you to calculate such scores.** However it is strongly encouraged that you understand how the scores are calculated.

  *(10 points)*

## 2 Part 2 *(25 points)*

- Write a function that estimates the transition parameters from the training set using MLE (maximum likelihood estimation):

  $$q(y_i|y_{i-1}) = \frac{\texttt{Count}(y_{i-1}, y_i)}{\texttt{Count}(y_{i-1})}$$

  Please make sure the following special cases are also considered: $q(\texttt{STOP}|y_n)$ and $q(y_1|\texttt{START})$.

  *(10 points)*

- Use the estimated transition and emission parameters, implement the Viterbi algorithm to compute the following (for a sentence with $n$ words):

  $$y_1^*, \ldots, y_n^* = \arg\max_{y_1,\cdots,y_n} p(x_1, \ldots, x_n, y_1, \ldots, y_n)$$

  Learn the model parameters with `train`. Run the Viterbi algorithm on the development set `dev.in` using the learned models, write your output to `dev.p2.out`. Report the precision, recall and F scores.

  *Note: in case you encounter potential numerical underflow issue, think of a way to address such an issue in your implementation.*

  *(15 points)*

## 3 Part 3 *(25 points)*

- Use the estimated transition and emission parameters, implement an algorithm to find the 4-th best output sequences. Clearly describe the steps of your algorithm in your report.

  Run the algorithm on the development sets `EN/dev.in`. Write the outputs to `EN/dev.p3.out`. Report the precision, recall and F scores.

  *Hint: find the top-4 best sequences using dynamic programming by modifying the original Viterbi algorithm.*

  *(25 points)*

## 4 Part 4 – Design Challenge *(20 points)*

- Now, based on the training and development set, think of a better design for developing systems for sequence labeling using any model you like. Please explain clearly the model/method that you used for designing the new system. We will check your code and may call you for an interview if we have questions about your code. Please run your system on the development set `EN/dev.in`. Write your outputs to `EN/dev.p4.out`. Report the precision, recall and F scores of your new systems.

  *(10 points)*

- We will evaluate your system's performance on two held out test sets `EN/test.in`. The test sets will only be released 48 hours before the deadline. Use your new system to generate the outputs. Write your outputs to `EN/test.p4.out`.

  The system that achieves the overall highest F score on the test sets will be announced as the winner.

  *(10 points)*

*Hints: Can we handle the new words in a better way? Are there better ways to model the transition and emission probabilities? Or can we use a discriminative approach instead of the generative approach? Perhaps using Perceptron?*[1]*. Any other creative ideas? Note that you are allowed to look into the scientific literature for ideas.*

## Items To Be Submitted

Upload to eDimension a single ZIP file containing the following: (Please make sure you have only one submission from each team only.)

- A report detailing the approaches and results
- Source code (.py files) with README (instructions on how to run the code)
- Output files
  - `EN/`
    1. `dev.p1.out`
    2. `dev.p2.out`
    3. `dev.p3.out`
    4. `dev.p4.out`
    5. `test.p4.out`

[1]http://www.aclweb.org/anthology/W02-1001