SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.007 Machine Learning, Fall 2015
Design Project

Due 9 Dec 2015, 5pm

This project will be graded by Yang Jie.
Please submit your work to *jie_yang@mymail.sutd.edu.sg*.

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

- **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. Please send your group information to Yang Jie by Tuesday 23 Oct 2015.**
- **Your should always implement all the algorithms on your own. You are strictly NOT allowed to use any external resources or machine learning / natural language processing / optimisation packages. You will receive 0 for this project if you do so.**
- **Part 1 deadline is Friday 30 Oct 2015. Annotated training and development set will be shared with you all on 6 Nov 2015.**
- **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 in a single zip file. Your output should be in the same column format as that of the training set.**

Several start-up companies in Singapore are interested in developing apps for mobile users that can perform customized product recommendation. The idea is to analyze the natural language texts typed, shared and read by users through services such as Twitter, WhatsApp, WeChat and SMS and analyze such texts to infer the users' interests. Such texts are different from standard texts that appear, for example, on news articles. They are often very informal, and can be very noisy. To this end, it is very essential to build systems which can automatically analyse and comprehend the underlying semantics associated with such informal texts, where the first step is to build efficient tools that can perform certain basic syntactic and semantic analysis of the texts. mobile apps

In this design project, we would like to design our sequence labelling model for informal texts using the hidden Markov model (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 natural language understanding system for mobile texts.

The files for this project are in the file `POS.zip` and `NPC.zip` (available on 6 Nov 2015, after we all have finished part 1). 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 a space and a single empty line that separates sentences.

For the `POS` dataset, the format can be something like the following:

```
@MiSS_SOTO USR
I PRP
think VBP
that DT
's VBZ
when WRB
I PRP
'm VBP
gonna VBG
be VB
there RB

On IN
Thanksgiving NNP
after IN
you PRP
done VBN
eating VBG
its PRP
#TimeToGetOut HT
unless IN
you PRP
wanna VBP
help VB
with IN
the DT
dishes NNS
```

where tokens such as `DT`, `VBZ` are part-of-speech (POS) tags.

For the `NPC` dataset which is to be released on 6 Nov 2015, the format can be something like the following:

```
Orh O
. O
Anyway O
are O
you B-NP
free O
tmr B-NP
afternoon I-NP
? O

Some B-NP
scavenger I-NP
hunt I-NP
thingy I-NP
```

```
my I-NP
fren I-NP
thought I-NP
up I-NP
. O
```

where `B-NP` and `I-NP` refer to the beginning and inner part of a noun phrase, and `O` means the current word is not part of any noun phrase.

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. There are two tasks involved here.

- We will be building a noun phrase chunking system from scratch, using our own annotations.
- We will be building a POS tag sequence prediction system using existing annotations.

1 Part 1 (15 points, due 30 Oct 2015 at 5pm)

The first and most important step towards building a supervised machine learning system is to get annotated data. This is also often one of the most difficult and most challenging steps in building a practical machine learning system, as we will see in this project. To allow each of us to have a full end-to-end experience on how challenging it is to build a practical supervised machine learning system, in the first part of this project. we will work together to get annotated data for performing noun phrase chunking from local SMS data. You will receive 10 points if you complete the annotation. An additional 5 points will typically be awarded to you too unless we found the quality of your annotation is unacceptable. We will use an automatic approach to assess the quality of your annotations. Your annotations may be shared with your fellow students in order to complete Part 2 and Part 3 of this project.

Please visit the following site for the annotation interface:

`http://brat.statnlp.com/main/#/sms_corpus/students/**YOUR_ID_HERE**/sms_corpus`

For example, if your student ID is 1001900, then you should visit the following URL:

`http://brat.statnlp.com/main/#/sms_corpus/students/1001900/sms_corpus`

We also provide a sample collection of annotated data, which is available here:

`http://brat.statnlp.com/main/#/sms_corpus/example`

Essentially, we are interested in annotating all noun phrases (NP). Detailed instructions on the annotation can be found at

`http://brat.statnlp.com/NP%20Annotation%20Guideline.pdf`

The annotation interface is straightforward to use, but if you have questions there is a manual here:

`http://brat.nlplab.org/manual.html`

Disclaimer: to grant us the right to re-distribute your annotated data to your other fellow classmates and for potential future usage, by submitting your annotations online, you agree that your annotated data will be in public domain unless otherwise stated. Please contact Yang Jie if you have questions or doubts on this.

2 Part 2 (25 points)

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

$$p(x_1, \ldots, x_n, y_0, y_1, \ldots, y_n, y_{n+1}) = \prod_{i=1}^{n+1} q(y_i|y_{i-1}) \cdot \prod_{i=1}^{n} e(x_i|y_i) \tag{1}$$

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 (POS tags or noun phrase tags).

- (5 pts) 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 pts) 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 is to assign a fixed probability to all such new words. For example, we can simply do the following:

  - If x is a new word (that did not appear in the training set):

$$e(x|y) = \frac{1}{\texttt{Count}(y) + 1}$$

  - If x appeared in the training set:

$$e(x|y) = \frac{\texttt{Count}(y \rightarrow x)}{\texttt{Count}(y) + 1}$$

  for any x and y. Implement this fix into your function for computing the emission parameters.

- (10 pts) Implement a simple POS tagger that produces the tag

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

  for each word x in the sequence.

  For both `POS` and `NPC` datasets, learn these parameters with `train`, and evaluate your system on the development set `dev.in` for `POS` and `NPC` dataset respectively. Write your output to `dev.p2.out` for the two datasets respectively. Compare your outputs and the gold-standard outputs in `dev.out` and report the accuracy score of such a baseline system for `POS` and `NPC` dataset respectively.

  The accuracy score is defined as follows:

$$\texttt{Accuracy} = \frac{\text{Total number of correctly predicted tags}}{\text{Total number of predicted tags}}$$

3 Part 3 (20 points)

- (5 pts) 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 $q(y_1|\texttt{START})$.

- (15 pts) Use the estimated transition and emission parameters, implement the Viterbi algorithm to compute the following:

$$y_1^*, \ldots, y_n^* = \arg\max_{y_1,\ldots,y_n} p(x_1, \ldots, x_n, y_0, y_1, \ldots, y_n, y_{n+1})$$

  For both `POS` and `NPC` datasets, 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.p3.out` for the two datasets respectively. Report the accuracy scores of both tagging systems.

  Note: you might encounter potential numerical underflow issue. Think of a way to address such an issue in your implementation.

4 Part 4 (20 points)

- (20 pts) Use the estimated transition and emission parameters, implement an algorithm to find the *10th-best* POS tag sequence. Clearly describe your algorithm in your report.

  Run the algorithm on the development set `POS/dev.in`. Write your output to `POS/dev.p4.out`. Report the accuracy score of this POS tagging system only.

  Hint: find the top-10 best POS tag sequences using dynamic programming by modifying the original Viterbi algorithm.

5 Part 5 – Challenge (20 points)

- (10 pts) Now, based on the training and development set, think of a better design for developing an improved POS tagger for tweets using any model you like. Please explain clearly the method that you used for designing the new POS tagger. 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 `dev.in`. Write your outputs to `POS/dev.p5.out`. Report the accuracy score of your new system.

- (10 pts) We will evaluate your system's performance on a held out test set `POS/test.in`. The test set will only be released on 7 Dec 2015 at 5pm (48 hours before the deadline). Use your new system to generate the output POS tags. Write your output to `POS/test.p5.out`.

  The system that achieves the highest accuracy on the test set will be announced as the winner.

Hints: Can we handle the new words in a better way? Are there better ways to model the transition probabilities? Or can we use a discriminative approach instead of the generative approach? Any other ideas?

Deliverables

1. A report describing your algorithms and results
2. A single zip file containing the following two folders:
   - under folder `POS`:
     (a) `dev.p2.out`
     (b) `dev.p3.out`
     (c) `dev.p4.out`
     (d) `dev.p5.out`
     (e) `test.p5.out`
   - under folder `NPC`:
     (a) `dev.p2.out`
     (b) `dev.p3.out`