Singapore University of Technology and Design
Information Systems Technology and Design
50.007 Machine Learning

Design Project

Due 11 Dec 2014, 5pm

This project will be graded by Dong Fei (fei_dong@mymail.sutd.edu.sg), so please submit your work to him.

Please form groups for this project early, and start this project early.
Please send your group information to Dong Fei by Tuesday 18 Nov 2014

Twitter (`twitter.com`) is one of the most popular microblogging sites which contains a large amount of user-generated textual data. Many start-ups in Silicon Valley are working on developing applications that provide recommendation services based on sentiment analysis on tweets. To this end, it is very essential to build systems which can automatically analyse and comprehend the underlying semantics associated with such texts, where the first step is to build efficient tools that can perform certain basic syntactic analysis of the tweets.

In this design project, we would like to design our Part-of-Speech (POS) Tagger for tweets using the hidden Markov model (HMM) that we have learned in class. We hope that your POS tagging system for tweets can serve as the very first step towards building a more complex, intelligent natural language understanding system for Twitter.

The files for this project are in the file `project.zip`. We provide a labelled training set `train`, an unlabelled development set `dev.in`, and a labelled development set `dev.out`. The labeled data has the format of one token per line with token and tag separated by tab and a single empty line that separates sentences:

```
@USER_66d8d837 @
I O
like V
monkeys N
,,
but &
I O
still R
hate V
COSTCO ^
parking N
lots N
.. ,
```

```
RT ~
@USER_99544f94 @
: ~
wat O
muhfuckaz N
wearin V
4 P
the D
lingerie N
party N
????? ,
```

The task is to learn a POS tagger from such training data and then use the system to predict POS tag sequences for new sentences.

A few notes:

- **This is a group project. You are allowed to form groups in any way you like, but each group must consist of 2-3 people.**
- **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.**

1 Part 1 (20 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 $y_0 = *$ and $y_{n+1} = \texttt{STOP}$. Here q are transition probabilities, and e are emission parameters.

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

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

- (5 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):

$$\frac{1}{\texttt{Count}(y) + 1}$$

– If x appeared in the training set:

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

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.

Learn these parameters with `train`, and evaluate your system on the development set `dev.in`. Write your output to `dev.p1.out`. Compare your outputs and the gold-standard outputs in `dev.out` and report the accuracy score of such a baseline system. The accuracy score is defined as follows:

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

2 Part 2 (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 $q(y_1|*)$.

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

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

Run the Viterbi algorithm on the development set `dev.in`. Write your output to `dev.p2.out`. Report the accuracy score of this POS tagging system.

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

3 Part 3 – Challenge (10 points)

- (5 pts) Now, based on the training and development set, think of a better design for developing an improved POS tagger for tweets using the hidden Markov model. You are not allowed to use any external resources or packages. Note that you must design your new POS tagger within the framework of the hidden Markov model but you are free to perform any automatic preprocessing of the data. Please run your system on the development set `dev.in`. Write your outputs to `dev.p3.out`. Report the accuracy score of your new system.

- (5 pts) We will evaluate your system's performance on a held out test set `test.in`. The test set will only be released on 9 Dec 2014 at 5pm (48 hours before the deadline). Use your new system to generate the output POS tags. Write your output to `test.p3.out`.

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

Hints: Are there better ways to handle new words? Are there better ways to model the transition probabilities? Are there linguistic patterns associated with words? Can we cluster words into semantic classes? Any other ideas?