CS4248
AY 2022/23 Semester 1
Assignment 1

Due Date

Friday 16 September 2022 at 9am. Late assignments will not be accepted and will receive ZERO mark.

Objective

In this assignment, you are to write a program in Python 3.8 to perform part-of-speech (POS) tagging. Given a sentence, the task is to assign a POS tag to each word in the sentence. We will adopt the Penn Treebank tag set in this assignment.

For example, given the following input sentence:

```
He also is a consensus manager .
```

The POS tagger should output the following:

```
He/PRP also/RB is/VBZ a/DT consensus/NN manager/NN ./.
```

Note that each input sentence has been tokenized (punctuation symbols are separated from words).

You are to implement a POS bigram tagger based on the hidden Markov model, in which the probability of the POS tag of the current word is conditioned on the POS tag of the previous word. The Viterbi algorithm will be used to find the optimal sequence of most probable POS tags.

Training and Testing

You are provided with a training set of POS-tagged sentences (`sents.train`). You will train your POS tagger using this training set. The command for training the POS tagger is:

```
python3.8 buildtagger.py sents.train model-file
```

The file `model-file` is the output of the training process and contains the statistics gathered from training, which include the POS tag transition probabilities and the word emission probabilities.

The test file `sents.test` consists of a list of sentences (without POS tags), one sentence per line. The command to test on this test file and generate an output file `sents.out` is:

```
python3.8 runtagger.py sents.test model-file sents.out
```

Your output file `sents.out` must have the same format as the POS-tagged training file `sents.train`.

The following command computes the accuracy of POS tagging, where `sents.out` is the output file of your code and `sents.answer` is the file containing the human-annotated POS tags:

```
python3.8 eval.py sents.out sents.answer
```

You are provided with the scoring code `eval.py`.

Deliverables

The commands `buildtagger.py` and `runtagger.py` as shown above will be executed to evaluate your POS tagger. Grading will be done after the submission deadline, by testing your POS tagger on a set of new, blind test sentences.

You will need to submit your files `buildtagger.py` and `runtagger.py` via CodeCrunch. Please do not change the file names and do not submit any files other than `buildtagger.py` and `runtagger.py`. Use the skeleton code for `buildtagger.py` and `runtagger.py` released to you to add your code.

The URL of CodeCrunch is as follows:

https://codecrunch.comp.nus.edu.sg/index.php

Login to CodeCrunch with your NUSNET ID and choose the task 'CS4248 (22/23 Sem 1) A1' under the CS4248 module. Submit your two Python files there.

After submission, you can find the accuracy of your POS tagger on the 'My Submissions' tab after execution is completed. The test file used in CodeCrunch is `sents.test`. We will run your code on the SoC cluster nodes xgpg0 to xgpg2. You can test your code on these nodes before submitting in CodeCrunch. You can login to these cluster nodes using your SoC Unix ID. Details of these computing nodes can be found at the following URL:

https://dochub.comp.nus.edu.sg/cf/guides/compute-cluster/hardware

Some Points to Note:

1. Your `buildtagger.py` should not take more than 2 minutes to complete execution on CodeCrunch. The same applies to `runtagger.py`. Your code will be terminated by CodeCrunch if it takes more than 2 minutes to execute.

2. Arguments to the Python files are absolute paths, not relative paths. They will be passed as arguments to the Python files, so please do not hard code the paths in your code.

3. We will use python3.8 to run your code. As such, please only use python3.8 when testing your code.

4. You can make a maximum of 99 submissions to CodeCrunch for this assignment.

Grading

The marks awarded in this assignment will be based on the accuracy of your POS tagger on a blind test set of sentences.