USC

CSCI 544 — Applied Natural Language Processing

Homework 5

Due: March 11, 2016, at 23:59 Pacific Time (11:59 PM)

Assignments may be turned in until March 19 without penalty.

Note: USC Spring Recess is March 12–19, and the due date of the assignment is before Spring Recess. Due to student requests, I am allowing submission until the end of Spring Recess. However, keep in mind that during Spring Recess there will be no office hours, and the instructional staff will spend less time than usual on Piazza. Therefore, students are **strongly encouraged** to complete the assignment before Spring Recess.

Overview

In this assignment you will write a Hidden Markov Model part-of-speech tagger for Catalan. The training data is provided tokenized and tagged; the test data will be provided tokenized, and your tagger will add the tags. The assignment will be graded based on the performance of your tagger, that is how well it performs on unseen test data compared to the performance of a reference tagger.

Data

A set of training and development data will be made available as a compressed ZIP archive on Blackboard. The uncompressed archive will have the following format:

- A file with tagged training data in the word/TAG format, with words separated by spaces and each sentence on a new line.
- A file with untagged development data, with words separated by spaces and each sentence on a new line.
- A file with tagged development data in the word/TAG format, with words separated by spaces and each sentence on a new line, to serve as an answer key.
- A readme/license file (which you won't need for the exercise).

The grading script will train your model on all of the tagged training and development data, and test the model on unseen data in a similar format.

Programs

You will write two programs: `hmmlearn.py` will learn a hidden Markov model from the training data, and `hmmdecode.py` will use the model to tag new data. If using Python 3, you will name your programs `hmmlearn3.py` and `hmmdecode3.py`. The learning program will be invoked in the following way:

```
> python hmmlearn.py /path/to/input
```

The argument is a single file containing the training data; the program will learn a hidden Markov model, and write the model parameters to a file called `hmmmodel.txt`. The format of the model is up to you, but it should contain sufficient information for `hmmdecode.py` to successfully tag new data.

The tagging program will be invoked in the following way:

```
> python hmmdecode.py /path/to/input
```

The argument is a single file containing the test data; the program will read the parameters of a hidden Markov model from the file `hmmmodel.txt`, tag each word in the test data, and write the results to a text file called `hmmoutput.txt` in the same format as the training data.

Grading

We will train your model, run your tagger on new test data, and compute the accuracy of your output compared to a reference annotation. Your grade will be the accuracy of your tagger, scaled to the performance of a reference HMM tagger developed by us. Since part-of-speech tagging can achieve high accuracy by using a baseline tagger that just gives the most common tag for each word, only the performance above the baseline will be scaled:

- If your accuracy <= baseline accuracy, your grade is your accuracy.
- If baseline accuracy < your accuracy < reference accuracy, your grade is baseline + (1 – baseline) × (yours – baseline) / (reference – baseline).
- If reference accuracy <= your accuracy, your grade is 100.

For example, if the baseline is 90%, the reference in 95%, and your accuracy is 93%, then your grade will be 0.9 + 0.1 × 0.03 / 0.05 = 96%.

Notes

- **Slash character.** The slash character '/' is the separator between words and tags, but it also appears within words in the text, so be very careful when separating words from tags. To make life easy, all tags in the data are exactly two characters long.
- **Smoothing and unseen words and transitions.** You should implement some method to handle unknown vocabulary and unseen transitions in the test data, otherwise your programs won't work. The unknown vocabulary problem is familiar from your naive Bayes classifier. The unseen transition problem is more subtle: you may find that the test data contains two adjacent unambiguous words (that is, words that can only have one part-of-speech tag), but the transition between these tags was never seen in the training data, so it has a probability of zero; in this case the Viterbi algorithm will have no way to proceed. The reference solution will use add-one smoothing on the transition probabilities and no smoothing on the emission probabilities; for unknown tokens in the test data it will ignore the emission probabilities and use the transition probabilities alone. You may use more sophisticated methods which you implement yourselves.

- **Runtime efficiency.** Vocareum imposes a limit on running times, and if a program takes too long, Vocareum will kill the process. Your program therefore needs to run efficiently. Run times for the reference solution are approximately 2 seconds for running `hmmlearn.py` on the training data and 5 seconds for running `hmmdecode.py` on the development data, running on a MacBook Pro from 2012.

Collaboration and external resources

- This is an individual assignment. You may not work in teams or collaborate with other students. You must be the sole author of 100% of the code you turn in.
- You may not look for solutions on the web, or use code you find online or anywhere else.
- You may not download the data from any source other than the files provided on Blackboard, and you may not attempt to locate the test data on the web or anywhere else.
- You may use external resources to learn basic functions of Python (such as reading and writing files, handling text strings, and basic math), but the extraction and computation of model parameters, as well as the use of these parameters for tagging, must be your own work.
- Failure to follow the above rules is considered a violation of academic integrity, and is grounds for failure of the assignment, or in serious cases failure of the course.
- Please discuss any issues you have on the Piazza discussion boards. Do not ask questions about the assignment by email; if we receive questions by email where the response could be helpful for the class, we will ask you to repost the question on the discussion boards.

Submission

All submissions will be completed through Vocareum; please consult the instructions for how to use Vocareum.

Multiple submissions are allowed, and your last submission will be graded. The submission script runs the program in a similar way to the grading script (but with different data), so you are encouraged to **submit early and often** in order to iron out any problems, especially issues with the format of the final output. The accuracy of you classifier will be measured automatically; failure to format your output correctly may result in very low scores, which will not be changed.

If you have any issues with Vocareum with regards to logging in, submission, code not executing properly, etc., please contact Siddharth.