CIS 410/510: Natural Language Processing
Fall 2019

due 11:55pm Monday, Nov. 4, 2019

1 Problem 1

[30 points]

This is the first programming assignment for the NLP class.

The minimum requirement for this assignment is to implement and train a part-of-speech tagger using the Hidden Markov Model and the Viterbi decoding algorithm, as described in class and in section 5.5 of the text (i.e., SLP2).

We provide you with a copy of a standard corpus used for testing POS taggers. This data is made available to you as a resource through Canvas in the form of a zip file, POS_CORPUS_FOR_STUDENTS.zip. This zip file contains:

- POS_train.pos: words and tags for training corpus
- POS_dev.word: words for development corpus
- POS_dev.pos: words and tags for development corpus
- POS_test.word: words for test corpus
- scorer.py: the scorer file provided to evaluate the output files (need Python 3)

Files with a "word" extension have the document text, one word per line; for those with a "pos" extension, each line consists of a word, a tab character, and a part-of-speech. In both formats a sentence boundary is indicated by a empty line.

You should train your tagger on the training corpus and evaluate its performance on the development corpus. We provide a simple Python scoring program for this purpose (we will use this scorer to score your output on the test set as well). Please read the scorer file to know how to run it (basically "python scorer.py keyFile responseFile"). Having a development corpus allows you to evaluate alternatives in designing the tagger – for example, how to treat unknown words. When you are satisfied with your tagger you should run it on the test data and submit the result. You have the option of training on the union of the training and development corpora before making this final run. Note that we do not provide the key (pos file) for test data; you should not train or do development using the test data.

The minimal requirement involves Viterbi decoding with a uniform probability for unknown words (i.e., if you see an unknown word in test data, you assume the same probability for every POS tag in the emission probabilities). If you are more ambitious, you may want to make additional enhancements to improve performance through better treatment of unknown words or through trigram models, as discussed in section 5.8 of the text (i.e., SLP2). Some extra credit will be given to assignments which incorporate such improvements.

The assignment you submit through Canvas should include three attachments (all put in a single zip file):

- a write-up (in the PDF format named "write-up.pdf"). If doing the minimal assignment this will be very brief, essentially reporting the score on the development corpus. If you have implemented additional features which have yielded improved performance, you should describe them here. Finally, all the instructions to running the code or so should be put here in the end of this file.

- your code (preferably put in a separate directory).

- the output of your code when run on the test data; this should have the file name "wsj_23.pos" and the same format as the other "pos" files.

We anticipate giving 30 points out for meeting the minimal requirement (i.e., full credit) and up to 5 additional points (bonus) for more elaborate submissions (i.e., trying more features/methods to improve the HMM model that can lead to better performance on the development set based on your experiments). The submissions with top performance on the test set will be rewarded 5 additional points/bonus (so the maximal point is 40). In particular, we will take your submitted output file for the test set, run the scorer that compares your submitted output file with the golden annotation file we have for test data, and obtain the score for your output file. Please make sure your output file follows the format of the "pos" files so our scorer can run correctly. Once you submitted your files, we won't be responsible for the wrong format of your output file or any other issues. If our scorer crashes with your output file, your score would be zero. In order to ensure the right format for your output, you are encouraged to use the provided scorer to see if it would fail with your output file format (e.g., in the development data).

We expect that most students will submit assignments in Python. If you wish to submit this assignment in another language, please make sure you include an clear instruction on how to install the requirements and run your code.

You may discuss methods with fellow students, but the code you submit must be your own.