CS 410/510: Natural Language Processing
Spring 2024

due 11:55pm May 3, 2024

1 Problem 1

[110 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 (HMM) and the Viterbi decoding algorithm, as described in class and in Section 8.4 of the text (i.e., SLP3: `https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf`).

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.

To handle unknown words, the minimal requirement for the Viterbi decoding is to use a uniform probability: if you see an unknown word in test data, you assume the same probability for every POS tag in the emission probabilities. We anticipate giving 100 points out for meeting the minimal requirement (full credit). You are also encouraged to explore different methods to improve the performance of your HMM model for our dataset. We will provide up to 10 additional points (extra credits) for submissions with top performance on the test set (so the maximal point is 110). Some potential techniques to improve HMMs include using better approaches to handle unknown words or exploring trigram models (i.e., the second-order Markov assumption).

For test set performance, we will run the scorer to compare your submitted output against the golden annotation file we have for test data. We will use the accuracy returned by the scorer for your output file. Please make sure your output file follows the format of the ".pos" files so the scorer can run correctly. If the scorer crashes with your submitted output file due to any issues (e.g., wrong format), your test set performance would be zero. To ensure correct format for the output file, you can use the provided scorer to check if it would fail with your output file format (e.g., in the development data).

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 expect that most students will use Python for the submissions. If you wish to do 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.