Homework 2

This assignment is about POS tagging. You should be aware that POS tagging is widely used, and that it's widely seen in the NLP community as a solved problem (or at least well enough solved that most people don't put much effort into improving POS tagging for its own sake). It's a good problem for learning about NLP, though, and that's what we are aiming for in this exercise. For this assignment, you may only use the datasets provided. We realize that you could probably track down the test data we will evaluate your models on, but that would be cheating – so don't.

Download and unpack the Homework2.zip. There is a `README` file, three Perl scripts, one Python script, and some data. Using what we've given you, you can build a baseline bigram HMM POS tagger on the standard training set from the Penn Treebank (sections 2-21, `ptb.2-21.*`) and evaluate it on the development set (section 22, `ptb.22.*`). (Actually, we already did that, but you should take two minutes to follow the `README` and do it yourself to make sure you understand the tools.) Note that we have not given you the standard test set output.

Task 1 (7 pts):

A learning curve is a useful tool that lets you visualize how a model's performance depends on the amount of training data. You can vary the amount of training data by creating smaller sub-corpora from the original full corpus. The learning curve plots a performance measure evaluated on a fixed test set (y-axis) against the training dataset size (x-axis). Generate a learning curve for the bigram HMM as we've provided it, using section 22 to evaluate. What are your thoughts about getting more POS-tagged data and how that would affect your system?

Task 2 (7 pts):

Come up with a way to improve the model. Again, you can only train on the training data (no external resources are allowed). Some relatively easy ideas: implement a trigram HMM, or try to smooth the probability estimates. It may be helpful to know that many tagging schemes encode tags in such a way that if you replaced every tag by its first letter only, the tags would still be meaningful, only more coarse (this holds for all of the datasets used in this assignment). You are constrained to using an HMM: you are building a replacement for the `train_hmm.{py,pl}` script. You can use our `viterbi.pl` script for tagging (**clarification: you may modify this script**) or write your own Viterbi algorithm in Python, though you may need a post-processing script that runs after the viterbi.pl. Describe your approach clearly. How well does your model perform on the development data? Run your tagger on `ptb.23.txt` (the test data) and turn in the output so we can evaluate it using `tag_acc.pl`.

Task 3 (3 pts):

Train the baseline model and your model on the Japanese (`jv.*`) and Bulgarian (`btb.*`) training datasets, and compare them on the test sets. Report the performance in a readable way, and discuss. You should look at the data (obviously!) and give some analysis – what factors lead to the differences among performance on English, Japanese, and Bulgarian? What about the baseline and your model makes them relatively better or worse on the data in these other two languages?

Submission

Please submit a zip archive containing the following items. Note that you are not submitting code for this assignment; we rely on you to describe your approach well.

- Your modified code (you can submit the entire folder, compressed)
- the output of your new model on `ptb.23.txt` (we will evaluate it against gold-standard POS tags),
- a text or pdf file containing your answers to all questions, including the plot for task 1. **Whether or not you collaborated with other individuals or employed outside resources, you must include a section in this file documenting your consultation (or non-consultation) of other persons and resources.**