Assignment 3

In this assignment you are to implement an HMM-based approach to POS tagging. Specifically, you are to implement the Viterbi algorithm using a bigram tag/state model. As training data, I am providing a POS-tagged section of the BERP corpus. Your systems will be evaluated against an unseen test set drawn from the same corpus.

Sample sentence

```
i        PRP

'd       MD

like     VB

french   JJ

food     NN

.        .
```

Training

The training data consists of around 15,000 POS-tagged sentences from the BERP corpus. The sentences are arranged as one word-tag pair per line with a blank line between sentences, words and tags are tab-separated. Contractions are split out into separate tokens with separate tags. An example is shown above.

You should assume that the tags that appear in the training data constitute all the tags that exist (no new tags will appear in testing). On the other hand, new words may appear in the test set.

Decoding

For decoding, your system will read in sentences from a file with the same format minus the tags. That is one word per line ending with a period and blank line before the next sentence. As output you should emit an appropriate tag for each word in the same format as the training data.

Evaluation

To know if you're making progress on improving your system during development you need to have an evaluation metric. I will provide an evaluation script that reports a simple aggregate accuracy score. To use the script from the command line on a Unix system, type something like this:

```
evalPOS.py  berp-key.txt berp-out.txt
```

where berp-key.txt contains the **word <tab> tag** pairs for the gold standard tags and berp-out.txt contains the system output to be evaluated.

Assignment

To complete the assignment, you'll need to address the following problems:

1. Divide the training file into training and development files. An 80%-20% split is typical, or you could do 5-fold cross-validation.
2. Extract the required counts from the training data for the various probability estimates that are needed.
3. Deal with unknown words in some systematic way.
4. Do some form of smoothing (for the bigram transition probabilities).
5. Implement the Viterbi (Figure 5.17) algorithm.
6. Tune your system in some sensible manner. This may involve writing a smarter evaluation script than the one I'm giving you. In particular, an evaluation script that produced a per class error rate, or even better, a complete confusion matrix would be useful.

What to hand in and when

By 11:30 p.m., Friday, March 24, turn in code that covers steps 1-5 above. It should run without errors, but terrible results are fine.

By 11:30 p.m., Friday, April 7, turn in your revised code, along with a short report describing what you did and all the choices you made. You should include in your report how you assessed your system (training/dev) and how well you believe it works. In addition, I will provide a test set for you to run your system on shortly before the due date. Include the output of your system on this test data in the same form as the training data.