CSCE-489, Programming Assignment #2 Viterbi
Due: Sunday, October 14, 2018 by 11:00pm

For this assignment, you should write a program that implements the bigram Viterbi algorithm as applied to part-of-speech tagging. No starter code is provided this time since I expect you can handle file IO etc by now. As we discussed in class, the Viterbi algorithm is an efficient way to find the most likely part-of-speech tag sequence for a sentence. As input, your program will need two files: (1) a file of probabilities, and (2) a file of sentences to process. Your program should read the names of the 2 input files from the command line. The command-line arguments should be in the following order:

```
viterbi <probabilities_file> <sentences_file>
```

For example, we should be able to run your program by typing:

```
viterbi probs.txt sents.txt
```

1. The Probabilities File

This file will contain bigram and lexical generation probabilities. Each line will be formatted as:

X Y probability

which means that $P(X \mid Y) = probability$.

If both X and Y are part-of-speech tags, then it is a bigram probability. If X is a word and Y is a part-of-speech tag, then it is a lexical generation probability. To keep things simple for this assignment, you should assume that there are only 4 possible part-of-speech tags: *noun verb inf prep*, plus a special tag *phi* to denote the beginning of a sentence and a special tag *fin* to denote the end of a sentence. (If you see any of these six strings, you can assume that they are tags and not words.) A sample probabilities file might look like this:

bears noun .02
bears verb .02
fish verb .07
fish noun .08
noun phi .80
verb phi .10
fin verb .50
noun verb .77
etc.

Use a default value of .0001 for all bigram and lexical generation probabilities that are <u>not</u> present in the probabilities file. Your program should be case insensitive, so "bears", "Bears", and "BEARS" should all be treated as the same token.

2. The Sentences File

This file will contain a set of sentences that you should run your program on. Each sentence will be on a separate line. There will not be any punctuation marks in this file. A sample file might look like this:

> This is a sentence
> And here is another sentence

OUTPUT FORMATTING

The output produced by your program should consist of 4 items for each sentence that is processed:

1. The sentence being processed.
2. The probabilities associated with every node in the Viterbi network when the algorithm is finished.
3. The values associated with every node in the Backpointer network (i.e., the data structure that you use to reconstruct the final tag sequence) when the algorithm is finished. NOTE: this information could be stored in the Viterbi network itself or in a separate data structure. Either way, please print these values separately from the Viterbi probabilities.
4. The probability of the best tag sequence and the part-of-speech tag assignments for the best tag sequence.

IMPORTANT: When you print this information, please format it exactly like the example in Figure 1 on the next page! Since you'll be printing a lot of numbers, this is crucial to ensure that we know exactly what your numbers and results correspond to.

Please print the nodes as if you were reading the sentence left-to-right (so all nodes for the leftmost word in the sentence should appear first) and then print the nodes within a column ordered as "noun", "verb", "inf", and "prep". The trace files that we give you will follow this convention. When printing the probability values, please print exactly 10 digits to the right of the decimal point.

The final part-of-speech tag assignments should be printed starting with the last word in the sentence and ending with the first word. This order should be the easiest one for you to produce as you traverse the backpointer network.

```
PROCESSING SENTENCE: bears fish

FINAL VITERBI NETWORK
P(bears=noun) = 0.0160000000
P(bears=verb) = 0.0020000000
P(bears=inf) = 0.0000000100
P(bears=prep) = 0.0000000100
P(fish=noun) = 0.0001232000
P(fish=verb) = 0.0007280000
P(fish=inf) = 0.0000000440
P(fish=prep) = 0.0000004800

FINAL BACKPTR NETWORK
Backptr(fish=noun) = verb
Backptr(fish=verb) = noun
Backptr(fish=inf) = verb
Backptr(fish=prep) = noun

BEST TAG SEQUENCE HAS PROBABILITY = 0.0007280000
fish -> verb
bears -> noun
```

Figure 1: Sample Output for Viterbi Program

GRADING CRITERIA

Your program will be graded based on both given dev cases (50%) and new test cases (50%)! **So please test your program thoroughly to evaluate the generality and correctness of your code!** Even if your program works perfectly on the examples that we give you, that does not guarantee that it will work perfectly on additional test cases.

ELECTRONIC SUBMISSION INSTRUCTIONS
(a.k.a. "What to turn in and how to tun in")

You need to submit 2 things:

1. The source code files for your program. Be sure to include all files that we will need to compile and run your program!

2. A report file that includes the following information:

   - how to compile and run your code
   - results and analysis
   - any known bugs, problems, or limitations of your program

REMINDER: your program **must** compile and run. We will not consider code that cannot run.

How to turn in:

1. please include everything in a single .tar.gz (.tgz) package and email it to me at "huangrh@cse.tamu.edu".
2. please wrtie the subject line as: "PA #2 for CSCE489 - UIN - LastName", please replace the placeholders "UIN" and "LastName" with your own UIN and LastName.