Assignment 3 CS562, fall 2012 Prof. Knight TA: Hui Zhang
Due at the beginning of class, October 25.

1. EM using Carmel

Type in this source.wfsa, a bigram WFSA over symbols “c1” and “c2”, in Carmel format:

```
F
(0 (c1 *e* c1 0.5))
(0 (c2 *e* c2 0.5))
(c1 (c1 *e* c1 0.49))
(c1 (c2 *e* c2 0.49))
(c2 (c1 *e* c1 0.49))
(c2 (c2 *e* c2 0.49))
(c1 (F *e* *e* 0.02))
(c2 (F *e* *e* 0.02))
```

Create channel.wfst, a single-state WFST that transforms “c1/c2” sequences into letter sequences:

```
0
(0 (0 c1 a 0.03))
(0 (0 c1 b 0.03))
(0 (0 c1 c 0.03))
(0 (0 c1 d 0.03))
(0 (0 c1 e 0.03))
(0 (0 c1 f 0.03))
(0 (0 c1 g 0.03))
(0 (0 c1 h 0.03))
(0 (0 c1 i 0.03))
(0 (0 c1 j 0.03))
(0 (0 c1 k 0.03))
(0 (0 c1 l 0.03))
(0 (0 c1 m 0.03))
(0 (0 c1 n 0.03))
(0 (0 c1 o 0.03))
(0 (0 c1 p 0.03))
(0 (0 c1 q 0.03))
(0 (0 c1 r 0.03))
(0 (0 c1 s 0.03))
(0 (0 c1 t 0.03))
(0 (0 c1 u 0.03))
(0 (0 c1 v 0.03))
(0 (0 c1 w 0.03))
(0 (0 c1 x 0.03))
(0 (0 c1 y 0.03))
(0 (0 c1 z 0.03))
(0 (0 c2 a 0.03))
(0 (0 c2 b 0.03))
(0 (0 c2 c 0.03))
(0 (0 c2 d 0.03))
(0 (0 c2 e 0.03))
(0 (0 c2 f 0.03))
(0 (0 c2 g 0.03))
(0 (0 c2 h 0.03))
(0 (0 c2 i 0.03))
(0 (0 c2 j 0.03))
(0 (0 c2 k 0.03))
(0 (0 c2 l 0.03))
(0 (0 c2 m 0.03))
(0 (0 c2 n 0.03))
(0 (0 c2 o 0.03))
(0 (0 c2 p 0.03))
(0 (0 c2 q 0.03))
(0 (0 c2 r 0.03))
(0 (0 c2 s 0.03))
(0 (0 c2 t 0.03))
(0 (0 c2 u 0.03))
(0 (0 c2 v 0.03))
(0 (0 c2 w 0.03))
(0 (0 c2 x 0.03))
(0 (0 c2 y 0.03))
(0 (0 c2 z 0.03))
```

To ensure you have entered these correctly, type:

```
    % carmel -c source.wfsa
    % carmel -c channel.wfst
```

Next, take the TRAIN file from homework assignment 2, remove all underscores (“_”) and remove all but the first 3000 letters. Call this new file TRAIN1.

Next, do EM training to refine the probabilities of **both** finite-state machines above, by maximizing P(TRAIN1) = sum_t P(t) * P(TRAIN1 | t), where t ranges over all possible tag sequences:

```
    % carmel --train-cascade -HJ TRAIN1 source.wfsa channel.wfst
```

This will create source.wfsa.trained and chan.wfst.trained. See what chan.wfst.trained looks like. Improve your results by doing 10 random restarts:

```
    % carmel -! 10 --train-cascade -HJ TRAIN1 source.wfsa channel.wfst
```

You can also adjust the convergence/termination criterion with -X. The default is 0.999.

```
    % carmel -X 0.99999 -! 10 --train-cascade -HJ TRAIN1 source.wfsa channel.wfst
```

You can also set the maximum number of EM iterations with -M.

Experiment thoroughly with various settings, and observe the number of EM iterations, corpus perplexities, and resulting chan.wfst.trained models.

Turn in:

- Your learned channel probabilities. Only show transitions with probability > 0.01.
- A well-written report covering all the experiments you did, including qualitative and quantitative results.

2. EM implementation

Implement the forward-backward algorithm in order to obtain similar results without using Carmel. You will want to experiment with a shorter version of TRAIN1 when you begin, though final results should be on the full TRAIN1. Debug your implementation by making sure P(TRAIN1) gets better at every EM iteration.

Turn in:

- Your code.
- Your learned channel probabilities. Only show transitions with probability > 0.01.
- A Viterbi decoding of TRAIN1 (just the first 50 characters), using your learned models.
- A brief 5-page trace of your program running, organized and annotated so that we can clearly see it operating correctly. The trace should be automatically generated, not be hand-written.

3. A strange corpus

Download the file COPIALE from Blackboard. This corpus is also a sequence of characters, but the alphabet is much bigger, so every character has a nickname, such as “tri” or “ih” or “y..”. Note that character names are surrounded by quote marks, as some programs get confused by punctuation marks like periods and commas. First, count how many distinct characters there are in COPIALE. Next, **run EM on this corpus** using models like the ones above, obtaining the best P(COPIALE) you can.

Turn in:

- Your learned channel probabilities. Only show transitions with probability > 0.01.
- What does EM think about this corpus? You can look at http://stp.lingfil.uu.se/~bea/copiale to help determine whether your EM implementation is doing something interesting.