LING 570: Hw3
Due date: 11:45pm on Oct 21

Goal: Build a unigram POS tagger using Carmel. All the example files mentioned below are under **hw3/examples**/.

Q1 (20 points): Use Carmel to build a FST acceptor, **fst_acceptor.sh**.

- The format of the command line is: fst_acceptor.sh fst_file input_file > output_file
- fst_file is an FST in the Carmel format (e.g., "examples/fst1")
- Each line in the input file is a string (e.g., "examples/ex")
- Each line in the output_file has the format "x => y prob" (e.g., "ex.fst1"), where
  - x is the string from the input file
  - y is the output string if x is accepted by the FST, or *none* if x is not accepted by the FST. If there are multiple paths for an input string x, y is the output string of the path with the highest probability (for paths with the same probabilities, Carmel breaks the tie somehow)
  - prob is the probability of the chosen path.

Q2 (20 points): Manually create FSTs for the following regular relations and save the FSTs in Carmel format as files "fst1", …, "fst4".

- fst1 for $\{(a^n, b^n) \mid n >= 0\}$
- fst2 for $\{(a^n, b^{2n}c) \mid n >= 0\}$
- fst3 for$\{(a^n d^*, (bc)^n g) \mid n>=0\}$
- fst4 for $\{(a^m b^n c^p, d^{m+1} e^{2n} f^q g^r) \mid m,n,p,q,r>=0 \text{ and } q+r=p\}$

Run your fst_acceptor.sh from Q1 with those FSTs and hw3/examples/ex as input file, save the output files in ex.fst1, …, and ex1.fst4, respectively.

Q3 (60 points): To create a unigram POS tagger which consists of two parts: trainer.sh and decoder.sh. You might need to do something special for quotation marks appearing in the training and test data.

(a) (15 points): Build a trainer that takes a training_file as input and creates a FST from that.

- The command line has the format: **trainer.sh** training_file fst
- The training_file has the format "w1/tag1 w2/tag2 … w_n/tag_n" (e.g., "hw3/examples/training_data")
- The fst is an FST in the Carmel format.

(b) (15 points): Build a decoder that takes a test_file and an FST as input and output the tagged sentences as output.

- The command line has the format: **decoder.sh** fst test_file output_file
- The fst is the FST created by the trainer and it is in the Carmel format
- The test_file has the format "w1 w2 … w_n" (e.g., "examples/test_data")
- The output_file has the format "w1/tag1 w2/tag2 … w_n/tag_n   prob", where tag_i is the most common tag for the word w_i in the training data as indicated by the fst, and prob is P(tag1, .., tag_n | w1, …, w_i), which is calculated as the product of P(tag_i | w_i) for all the i.
- Note: decoder.sh could be very similar to fst_acceptor.sh.

(c) (10 points) Test your code on WSJ data: run the following commands:

- cd hw3/examples/
- trainer.sh wsj_sec0.word_pos your_hw3/wsj_sec0.fst
- decoder.sh your_hw3/wsj_sec0.fst wsj_sec0.word your_hw3/wsj_sec0.sys
- ./calc_tagging_acc.pl wsj_sec0.word_pos your_hw3/wsj_sec0.sys
- Here, your_hw3 is your hw3 dir and calc_tagging_acc.pl is a perl code that I wrote for calculating tagging accuracy.

(d) (20 points) In your hw3 note file, briefly answer the following questions:

- What does the FST created by the trainer look like?
- How do your trainer and decoder work?
- What tagging accuracy do you get when running the commands in (c)?
- In (c), the test data used with decoder.sh is the same as the training data (once the tags are removed). What kind of problems will this unigram tagger encounter if the test data is not the same as the training data?
- Have you done something special for the quotation mark? If so, what's that and why is it necessary?

The submission should include:

- The hw3 note file that includes answers to Q3(d).

- The source and shell scripts for Q1, Q3(a), Q3(b), and any scripts called by them.

- The four FSAs for Q2 (fsa1, fsa2, fsa3, and fsa4) and the output files created by running the following commands for fsa1-fsa4:

```
fst_acceptor.sh   fst1     hw3/examples/ex > ex.fst1 2>ex.fst1.log
…
fst_acceptor.sh   fst4     hw3/examples/ex > ex.fst4 2>ex.fst4.log
```

- The files created in Q3(c): wsj_sec0.sys