LING 570: Hw6
Due on Nov 11

All the example files are under dropbox/09-10/570/hw6/examples/. For Hw6 and Hw7, we will use state-emission HMMs where the output symbols are produced by the to-states. The HMMs will have the following format (e.g., **hmm_ex1** and **hmm_ex2**):

```
state_num=nn        ## the number of states
sym_num=nn          ## the size of output symbol alphabet
init_line_num=nn    ## the number of lines for the initial probability
trans_line_num=nn   ## the number of lines for the transition probability
emiss_line_num=nn   ## the number of lines for the emission probability

\init
state  prob  lg_prob  ## prob=\pi(state), lg_prob=lg(prob)
...

\transition
from_state to_state prob lg_prob  ## prob=P(to_state | from_state)
...

\emission
state symbol prob lg_prob         ## prob=P(symbol | state)
...
```

Q1 (15 points): Write a script, **create_2gram_hmm.sh**, that takes the annotated training data as input and creates an HMM for a bigram POS tagger with NO smoothing.

- The format is: cat training_data | create_2gram_hmm.sh output_hmm
- The training data is of the format "w1/t1 .... wn/tn" (cf. **wsj_sec0.word_pos**)
- The output_hmm has the format specified at the beginning of this file.
- In your note file, explain what the states and the transition and emission probs in the output_hmm represent.

Q2 (30 points): Write a script, **create_3gram_hmm.sh**, that takes the annotated training data as input and creates an HMM for a trigram POS tagger WITH smoothing.

- The format is: cat training_data | create_3gram_hmm.sh output_hmm l1 l2 l3 unk_prob_file
- The training data is of the format "w1/t1 .... wn/tn" (cf. **wsj_sec0.word_pos**)
- The output_hmm has the format as in Q1.

- unk_prob_file has the format "tag prob" (see **unk_prob_sec22**): prob is $P(< unk >| tag)$. They are used to smooth $P(word \mid tag)$; that is, for a known word w, $P_{smooth}(w \mid tag) = P(w \mid tag)$ * $(1 - P(< unk >| tag))$, where $P(w \mid tag) = \frac{cnt(w,tag)}{cnt(tag)}$.
- l1, l2 and l3 are λ_1, λ_2, λ_3 used in interpolation: $P_{int}(w_3 \mid w_1, w_2) = \lambda_3 P_3(w_3 \mid w_1, w_2) + \lambda_2 P_2(w_3|w_2) + \lambda_1 P_1(w_3)$.
- In your note file, explain what the states and the transition and emission probs in the output_hmm represent.

Q3 (40 points): Write a script, **check_hmm.sh**, that reads in a state-emission HMM file, check its format, and output a warning file. The main purpose of this exercise is to learn to represent an HMM in an efficient data structure, as you will use this data structure for Hw7. You might want to read 11_09_hmm_p2.pdf when deciding what data structure you want to use to store hmm.

- The format is: check_hmm.sh input_hmm > warning_file
- Your code should check the three kinds of constraints for HMM. If the constraints are not satisfied, print out the warning messages to the warning_file (cf. **hmm_ex1.warning**).
- The real line numbers for initial/transition/emission prob etc. could be different from what are claimed in the header. In that case, print out the warning messages to the warning_file.

Q4 (15 points): Run the following commands and turn in the files generated by the commands:

```
cat wsj_sec0.word_pos | create_2gram_hmm.sh 2g_hmm
cat wsj_sec0.word_pos | create_3gram_hmm.sh 3g_hmm_0.1_0.1_0.8 0.1 0.1 0.8 unk_prob_sec22
cat wsj_sec0.word_pos | create_3gram_hmm.sh 3g_hmm_0.2_0.3_0.5 0.2 0.3 0.5 unk_prob_sec22
check_hmm.sh 2g_hmm > 2g_hmm.warning
check_hmm.sh 3g_hmm_0.1_0.1_0.8 > 3g_hmm_0.1_0.1_0.8.warning
check_hmm.sh 3g_hmm_0.2_0.3_0.5 > 3g_hmm_0.2_0.3_0.5.warning
```

The submission should include:

- The hw6 note file that includes answers to Q1 and Q2.
- The source and shell scripts in Q1, Q2, Q3: **create_2gram_hmm.sh**, **create_3gram_hmm.sh**, and **check_hmm.sh**, and any scripts called by them.
- The files created in Q4: 2g_hmm* and 3g_hmm*.