LING 570: Hw4
Due on October 23
Total points: 100

Assignment goals

- Gain experience with part-of-speech tagging
- Investigate n-gram HMM tagging models
- Implement parameter learning and smoothing models
- Increase understanding of model constraints

Assignment resources & notes

All the example files are under /dropbox/25-26/570/hw4/examples/:

- More info about this assignment is in the class slide deck for Oct 16..
- For this assignment, we will use state-emission HMMs where the output symbols are produced by the to-states.
- In Q3, your code will read an HMM and check its format. The function(s) you write for this will be needed for your next assignment as well.

HMM file structure

- An HMM file (e.g., **hmm_ex1** and **hmm_ex2**) has two parts:
  1. A *header* that shows the numbers of states, output symbols, and lines for the three probability distributions (initial, transition, and emission)
  2. the three distributions: **the lg_prob field is optional** and *lg* means the base is 10.
- **Note:** The two parts in an input HMM file might not be consistent; for instance, the header may say that there are 10 states, but the distributions show that there are more than 10 states. In Q3 below, you will write a script that checks whether two parts are consistent, etc.

```
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 (20 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 above:
  - No need to truncate or round *prob* and *lg_prob*.
  - For each probability distribution (initial, transition, and emission probabilty), the probabilty lines should be sorted alphabetically on the 1st field (*state* or *from_state*) first, and then for lines with the same 1st field, sort on the second field. For instance, the emission probability lines are sorted by *state* first. For the lines with the same *state*, sort the lines by *symbol*.
  - **Note:** The example files on patas are not sorted, as they were created before, so those files are not meant to be gold standard.
- Run the following command and submit the output file, 2g_hmm:

  cat wsj_sec0.word_pos | ./create_2gram_hmm.sh 2g_hmm

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**.

You will implement smoothing for both transition and emission probabilities.

- For emission probabilities, you will model unseen words as observations.
- For transition probabilities, you will apply interpolation to estimate transition probabilities for all legal state-state transition, i.e. all possible trigrams.
- The format is: cat training_data | create_3gram_hmm.sh output_hmm lambda1 lambda2 lambda3 unk_prob_file
- The training data is of the format "w1/t1 .... wn/tn" (cf. **wsj_sec0.word_pos**)
- The output_hmm has the same format as in Q1.

- unk_prob_file is an **input** file (not an output file). That is, the file is given to you and you do not need to estimate it from the training data. The file's format is "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 $P(w \mid tag) = \frac{cnt(w,tag)}{cnt(tag)}$.
- lambda1, lambda2 and lambda3 are λ_1, λ_2, λ_3 used in interpolation: $P_{int}(t_3 \mid t_1, t_2) = \lambda_3 P_3(t_3 \mid t_1, t_2) + \lambda_2 P_2(t_3|t_2) + \lambda_1 P_1(t_3)$.
- When estimating $P_3(t_3 \mid t_1, t_2)$, if the bigram $t_1 t_2$ never appears in the training data, both $count(t_1, t_2, t_3)$ and $count(t_1, t_2)$ will be zeros. The value of dividing zero by zero is undefined. For this assignment, for the sake of simplicity, when $t_1 t_2$ is unseen in the training data, we will set $P_3(t_3 \mid t_1, t_2)$ to be $1/(|T|+1)$ when t_3 is a POS tag or EOS, and to zero when t_3 is BOS. Here, $|T|$ is the size of the POS tagset (which excludes BOS and EOS).
- Run the following command and submit the output files:

  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

Q3 (25 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 read in an HMM file and store it in an efficient data structure, as you will need to use this data structure for the next assignment. Think about what data structure you want to use to store hmm.

- The format is: check_hmm.sh input_hmm > warning_file
- Your code should check
  - whether the two parts of the HMM file are consistent (e.g., the number of states in the header matches that in the distributions), and
  - whether the three kinds of constraints for HMM (see slide #13 in day11-hmm-part1.pdf) are met.
- If the two parts are not consistent and/or the constraints are not satisfied, print out the warning messages to the warning_file (cf. **hmm_ex1.warning**).
- **In the note file, explain what data structure you use to store the HMM.**
- Run the following commands and turn in the files generated by the commands:

  ./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 readme.[txt | pdf] file that includes your answer to Q3. You should also discuss any challenges that you encountered and how you resolved them (or not).
- hw.tar.gz that includes all the files specified in submit-file-list:
  - create_2gram_hmm.sh
  - create_3gram_hmm.sh
  - check_hmm.sh
  - 2g_hmm
  - 3g_hmm_0.1_0.1_0.8
  - 3g_hmm_0.2_0.3_0.5
  - 2g_hmm.warning
  - 3g_hmm_0.1_0.1_0.8.warning
  - 3g_hmm_0.2_0.3_0.5.warning