

LING 570: Hw7  
Due on Nov 18  
Total points: 100

All the example files are under `dropbox/21-22/570/hw7/examples/`:

- More info about hw7 is on the slides for hw7.
- For Hw7, we will use state-emission HMMs where the output symbols are produced by the to-states.
- **Format of HMM files in hw7:** 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
  2. the three distributions (**the `lg_prob` field is optional**).
- The two parts in an input HMM file might not be consistent; for instance, the header says 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 (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 above:

- Like in Hw6, no need to truncate or round *prob* and *lg\_prob*.
- For each probability distribution (initial, transition, and emission probability), the probability 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*.
- The example files on patas are not sorted and rounded, as they were created before, so those files are not meant to be gold standard.

**Q2 (25 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 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  $P(word | tag)$ ; that is, for a known word  $w$ ,  $P_{smooth}(w | tag) = P(w | tag) * (1 - P(< unk > | tag))$ , where  $P(w | tag) = \frac{cnt(w,tag)}{cnt(tag)}$ .
- `l1`, `l2` and `l3` are  $\lambda_1$ ,  $\lambda_2$ ,  $\lambda_3$  used in interpolation:  $P_{int}(t_3 | t_1, t_2) = \lambda_3 P_3(t_3 | t_1, t_2) + \lambda_2 P_2(t_3 | t_2) + \lambda_1 P_1(t_3)$ .
- When estimating  $P_3(t_3 | t_1, t_2)$ , 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 hw7, for the sake of simplicity, when  $t_1 t_2$  is unseen in the training data, let’s set  $P_3(t_3 | t_1, t_2)$  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).

**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 (Hw8). 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.

**Q4 (10 points):** Run the following commands and turn in the files generated by the commands:

```
cat wsj_sec0.word_pos | ./create_2gram_hmm.sh q4/2g_hmm
cat wsj_sec0.word_pos | ./create_3gram_hmm.sh q4/3g_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 q4/3g_hmm_0.2_0.3_0.5 0.2 0.3 0.5 unk_prob_sec22
./check_hmm.sh q4/2g_hmm > q4/2g_hmm.warning
./check_hmm.sh q4/3g_hmm_0.1_0.1_0.8 > q4/3g_hmm_0.1_0.1_0.8.warning
./check_hmm.sh q4/3g_hmm_0.2_0.3_0.5 > q4/3g_hmm_0.2_0.3_0.5.warning
```

The submission should include:

- The readme.[txt | pdf] file that includes your answer to Q3.
- hw.tar.gz that includes all the files specified in submit-file-list.