# Hw6

LING 570

# Format of HMM

The header

```
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                   ## initial probability
state  prob   lg_prob   ## prob=\pi(state), lg_prob=lg(prob)
  ...                   ## All the lg_prob in this file are base-10 and optional
```

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

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

➔ sort the probability lines alphabetically by the first field first, and then by the 2nd field.

➔ for prob and lg_prob, keep 10 digits after the decimal point.

# Q1: HMM for a bigram tagger

- cat  training_data | create_2gram_hmm.sh output_hmm

- training data: w1/t1 … wn/tn
  - If a word contains a slash, use backslash before that.
  - Ex:  2/3 as a CD is written as 2\/3/CD

- No smoothing

- For bigram tagger, each state corresponds to a POS tag, BOS, or EOS.

# Q2: HMM for a trigram tagger

- cat training_data | create_3gram_hmm.sh output_hmm  l1  l2  l3  unk_prob_file

- unk_prob_file is an input file. Its format is “tag prob”, which means P(<unk>|tag)=prob

- If w is a known word

$$\text{then } P_{smooth}(w \mid tag) = P(w \mid tag) * (1 - P(< unk >| \; tag))$$

$$\text{else } P_{smooth}(w \mid tag) = P(< unk >| \; tag))$$

# Q2: HMM for a trigram tagger

- Smooth P(t3 | t1, t2) with interpolation:

$$P(t_3 \mid t_1, t_2) = \lambda_3 P_3(t_3 \mid t_1, t_2) + \lambda_2 P_2(t_3|t_2) + \lambda_1 P_1(t_3)$$

- $P_1()$, $P_2()$, $P_3()$ are probability distributions without smoothing.

- If the bigram (t1, t2) is unseen in the training data, what should $P_3$(t3 | t1, t2) be?

Let $P_3$(t3 | t1, t2) = 0 if t3 is BOS
= 1/(|T|+1) otherwise

Here, |T| is the size of the tagset.

- P(t3 | t1 t2) = prob

```
t1_t2   t2_t3   prob  lg_prob
```

- P(table | N) = prob

```
adj_N  table  prob  lg_prob
xx_N    table  prob  lg_prob
…
```

# Trigram tagger and HMM

- Trigram tagger: P(t3 | t1, t2) and P(w | t)

- HMM: $a_{ij} = P(s_j \mid s_i)$ and $b_{jk} = P(w_k \mid s_j)$

- $s_i = (t1, t2)$ $s_j = (t2', t3)$:

  $P(s_j \mid s_i) = P(t3 \mid t1, t2)$ if $t2 = t2'$

  $= 0$ otherwise

- $s_j = (t1, t2)$:

  $b_{jk} = P(w_k \mid s_j) = P(w_k \mid t2)$

# Q3: read and check HMM

- check_hmm.sh input_hmm > warning_file

- Check
  - whether the header matches the distributions, and
  - whether the constraints are satisfied:

$$\sum_{i=1}^{N} \pi_i = 1$$

$$\forall i \quad \sum_{j=1}^{N} a_{ij} = 1$$

$$\forall i \quad \sum_{k=1}^{M} b_{ik} = 1$$

Let Sum is a real number, how to check whether Sum is equal to 1?

# Q3: read and check HMM

- check_hmm.sh  input_hmm > warning_file

```
state_num=6
sym_num=11
warning: different numbers of init_line_num: claimed=2, real=1
   ## "claimed" is what is in the header,  "real" is what is in the distributions
warning: different numbers of trans_line_num: claimed=13, real=15
warning: different numbers of emission_line_num: claimed=11, real=12
warning: the trans_prob_sum for state N is 0.9
warning: the trans_prob_sum for state V is 1.1
warning: the emiss_prob_sum for state BOS is 0
warning: the emiss_prob_sum for state N is 0.5
warning: the emiss_prob_sum for state V is 0.85
warning: the emiss_prob_sum for state Adv is 0
```

# Implementation issue: storing HMM

Approach #1: use hash tables

- $\pi_i$: pi {state_str}
- $a_{ij}$: a {from_state_str} {to_state_str}
- $b_{jk}$: b {state_str} {symbol}

Approach #2: map a string to an index first

- state2idx{state_str} = state_idx
- symbol2idx{symbol_str} = symbol_idx

- $\pi_i$: pi [state_idx] = prob
- $a_{ij}$: a [from_state_idx] [to_state_idx] = prob
- $b_{jk}$: b [state_idx] [symbol_idx] = prob

- idx2state[state_idx] = state_str
- Idx2symbol[symbol_idx] = symbol_str

# Storing HMM: sparse matrix

- Two-dimensional array:
  - $a_{ij}$: a [i] [j] = prob
  - $b_{jk}$: b [j] [k] = prob

- One-dimensional array:
  - $a_{ij}$: a[i] = “j1 p1 j2 p2 …”, or
  - $a_{ij}$: a[j] = “i1 p1 i2 p2 …”

  - $b_{jk}$: b[j] = “k1 p1 k2 p2 ….”, or
  - $b_{jk}$: b[k] = “j1 p1 j2 p2 …”

# Other implementation issues

- Index starts from 0 in programming, but often starts from 1 in algorithms
- The sum of lgprob is used in practice to replace the product of prob.
- Check constraints and print out warning if the constraints are not met.