

# POS tagging (2)

LING 570

Fei Xia

# Unit 2 (so far)

- LM
  - Ngram model
  - Smoothing: interpolation, etc.
- POS tagging
  - N-gram model
  - HMM (I): definition

# Outline for today

- Using HMM for n-gram taggers:
  - Bigram tagger
  - Trigram tagger
- Smoothing:
  - Unseen tag sequences
  - Unknown words

# Using HMM for ngram taggers

# HMM

- HMM:
  - States:  $\{s_1, s_2, \dots, s_N\}$
  - Output symbols:  $\{w_1, w_2, \dots, w_m\}$
  - Initial prob:  $\pi_i$
  - Transition:  $a_{ij}$
  - Emission:  $b_{jk}$
- How to use HMM to build a n-gram tagger?

# N-gram POS tagger

$$\operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

$$\approx \operatorname{argmax}_{t_1^n} \prod_i P(w_i | t_i) P(t_i | t_{i-1}^{i-1})$$

Bigram model:  $\prod_i P(w_i | t_i) P(t_i | t_{i-1})$

Trigram model:  $\prod_i P(w_i | t_i) P(t_i | t_{i-2}, t_{i-1})$

# The bigram tagger

- States: POS tags, BOS, EOS
- Output symbols: words,  $\langle s \rangle$ ,  $\langle /s \rangle$
- Initial probability:  $\pi(\text{BOS}) = 1$ .
- Transition probability:  $a_{ij} = P(s_j \mid s_i)$
- Emission probability:  $b_{jk} = P(w_k \mid s_j)$

# The bigram tagger (cont)

$$O_1^n = w_1^n$$

$$X_1^{n+1} : X_1 = BOS, X_2 = t_1, \dots, X_{n+1} = t_n$$

$$P(O_1^n, X_1^{n+1})$$

$$= \pi(X_1) \prod_{i=1}^n P(O_i | X_{i+1}) P(X_{i+1} | X_i)$$

$$= \pi(BOS) \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$

$$= \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$



# The trigram tagger

- States: a tag pair, a tag is a POS tag or BOS or EOS
- Output symbols: words,  $\langle s \rangle$ ,  $\langle /s \rangle$
- Initial probability:  $\pi(\text{BOS\_BOS}) = 1$ .
- Transition probability:  
$$a_{ij} = P(t_3 | t_1, t_2), \text{ where } s_i = (t_1, t_2), \text{ and } s_j = (t_2, t_3)$$
$$= 0, \text{ where } s_i = (t_1, t_2), \text{ and } s_j = (t_2', t_3), \text{ and } t_2 \neq t_2'$$
- Emission probability:  
$$b_{jk} = P(w_k | t), \text{ where } s_j = (t', t) \text{ for any } t'$$

# The trigram tagger (cont)

$$O_1^n = w_1^n$$

$$X_1^{n+1} : X_1 = (BOS, BOS), X_2 = (BOS, t_1), \dots, X_{n+1} = (t_{n-1}, t_n)$$

$$P(O_1^n, X_1^{n+1})$$

$$= \pi(X_1) \prod_{i=1}^n P(O_i | X_{i+1}) P(X_{i+1} | X_i)$$

$$= \pi(BOS) \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-2}, t_{i-1})$$

$$= \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-2}, t_{i-1})$$

# Smoothing

# Why smoothing?

- To handle unseen tag sequences  
→ to smooth the transition prob
- To handle unknown words  
→ to smooth the emission prob
- To handle unseen (word, tag) pairs, where both word and tag are known (?)

# Handling unseen tag sequences

- Ex: To smooth  $P(t_3 | t_1, t_2)$  for a trigram tagger.
- How about interpolation?

$$P(t_3 | t_1, t_2) =$$

$$\lambda_1 P1(t_3) + \lambda_2 P2(t_3 | t_2) + \lambda_3 P3(t_3 | t_1, t_2)$$

# How about unknown words?

- Introduce a new output symbol: <unk>
- Estimate  $P(<unk> | t)$  for each tag  $t$ :
  - Ex: split training data into two sets: create the voc from set1 and estimate  $P(<unk>|t)$  from set2.
- Add  $P(<unk> | t)$  to the emission prob and renormalize so that  $\sum_w P(w|t) = 1$ 
  - Ex: Keep  $P(<unk>|t)$  the same, and make

$$\sum_{w \neq <unk>} P(w|t) = 1 - P(<unk> | t)$$

# Evaluation

- Tagging accuracy:
  - Overall: accuracy on all the words
  - Accuracy on unknown words

# Error Analysis

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	—	.2			.7		
JJ	.2	—	3.3	2.1	1.7	.2	2.7
NN		8.7	—				.2
NNP	.2	3.3	4.1	—	.2		
RB	2.2	2.0	.5		—		
VBD		.3	.5			—	4.4
VBN		2.8				2.6	—

- Confusion matrix (contingency table)
- Identify primary contributors to error rate
  - Noun (NN) vs Proper Noun (NNP) vs Adj (JJ)
  - Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)



# Summary so far

- We can use HMM to build ngram taggers.
- The best state sequence corresponds to the best tag sequence.
  - ➔ We can use the Viterbi algorithm to find the best tag sequence.
- Accuracy on PTB:
  - Unigram tagger: 91%
  - Trigram tagger: 93%

# Remaining issues

- Viterbi algorithm
  - ➔ next session
- Other algorithms
  - ➔ ling572
- How to exploit unlabeled data?
  - ➔ semi- and unsupervised learning

# Cues for predicting POS tags for unknown words

- Affixes: unforgettable: un-, -able → JJ
- Capitalization: Hyderabad → NNP
- Word shapes: 123,456 → CD
- The previous word: San \_ → NNP

How can we take advantage of these cues?

→ Treat them as features

# Unsupervised POS tagging

- Unlabeled data
  - ➔ learn word clusters
- What else could be available?
  - A lexicon: all allowed tags for each word
    - ➔ use unambiguous words as anchors
  - A few examples (prototypes): e.g., “book” is a noun, “the” is a determiner

# Additional slides

# Use HMM for trigram tagger

POS tag sequence vs. state sequence in HMM:

- Orig sent:                   the table is expensive
- With markers:   <s>       the       table is       expensive </s>
- Tag sequence:   BOS     DT     N     V     ADJ       EOS
- State sequence: BOS\_BOS BOS\_DT DT\_N   N\_V   V\_ADJ       ADJ\_EOS

# Possible vs. impossible transition

- Possible transition: Even if a tag bigram (e.g., DT V) does not appear in the training data, you should allow it. The tag bigram “DT V” will be handled by the transition  $x\_DT \Rightarrow DT\_V$  (where x is another tag)
- Impossible transition is the one where the 2<sup>nd</sup> tag of the from-state is different from the 1<sup>st</sup> tag of the to-state, such as  $x\_DT \Rightarrow y\_V$  where y is not DT

Do not include impossible transitions in your HMM file.

# Emission probability

- Recall that the output symbol is emitted by a state, not by a tag. But the trigram tagger uses  $P(w \mid \text{tag})$ .
- To resolve this “mismatch”, make the emission probability to rely on only the 2<sup>nd</sup> tag of a state.

Ex:  $P(\text{table} \mid N) = 0.01$  becomes

“x\_N table 0.01” in HMM, where x is any POS tag.