POS tagging (2)

LING 570
Fei Xia
Week 6: 11/04/09

Unit 2 (so far)

- LM
  - Ngram model
  - Smoothing: add-one, GT, backoff, interpolation
- 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: $\{s_1, s_2, \ldots, s_N\}$
  - Output symbols: $\{w_1, w_2, \ldots, w_m\}$
  - Initial prob: π_i
  - Transition: a_{ij}
  - Emission: b_{jk}

- How to use HMM to build a n-gram tagger?

N-gram POS tagger

$$argmax_{t_1^n} P(t_1^n | w_1^n)$$

$$\approx argmax_{t_1^n} \prod_i P(w_i | t_i) P(t_i | t_{i-N+1}^{i-1})$$

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, <s>, </s>
- Initial probability: π(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)

$$O_1^n = w_1^n$$

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

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

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

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

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

The trigram tagger

- States: a tag pair, a tag is a POS tag or BOS or EOS
- Output symbols: words, <s>, </s>
- Initial probability: π(BOS_BOS) = 1.
- Transition probability:

  $a_{ij} = P(t_3 | t_1, t_2)$, where $s_i=(t_1, t_2)$, and $s_j=(t_2, t_3)$

  $= 0$, where $s_i=(t_1, t_2)$, and $s_j=(t_2', t_3)$, and $t_2 \,!= t_2'$
- Emission probability:

  $b_{jk} = P(w_k | t)$, where $s_j=(t',t)$ for any t'

The trigram tagger (cont)

$$O_1^n = w_1^n$$

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

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

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

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

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

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?

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

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

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

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: 95%

Remaining issues

- Viterbi algorithm

  ➔ Week 7

- Other algorithms

  ➔ Week 8-9 and ling572

- How to exploit unlabeled data?

  ➔ semi- and unsupervised learning

Additional slides

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