Part-of-speech tagging (2)

Steve Renals
s.renals@ed.ac.uk

ICL — 23 October 2006

HMM PoS tagging

- Also referred to as n-gram tagging
- Sequence classification task: given a sequence of N words, return a sequence of N tags

HMM PoS tagging

- Also referred to as n-gram tagging
- Sequence classification task: given a sequence of N words, return a sequence of N tags
- Basic idea: consider *all* possible sequences of tags and choose the most probable given the sequence of words.
- Notation: let w_1^N be the sequence of N words, and t_1^N the sequence of N tags. Then our best hypothesis of the correct tag sequence, $\hat{t}_1^N$ is:

$$\hat{t}_1^N = \arg\max_{t_1^n} P(t_1^N | w_1^N)$$

HMM PoS tagging

- Also referred to as n-gram tagging
- Sequence classification task: given a sequence of N words, return a sequence of N tags
- Basic idea: consider *all* possible sequences of tags and choose the most probable given the sequence of words.
- Notation: let w_1^N be the sequence of N words, and t_1^N the sequence of N tags. Then our best hypothesis of the correct tag sequence, $\hat{t}_1^N$ is:

$$\hat{t}_1^N = \arg\max_{t_1^n} P(t_1^N | w_1^N)$$

- Problem: how do we compute $P(t_1^N | w_1^N)$?

Bayes rule

- Bayes rule is a useful way to manipulate conditional probabilities:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

Bayes rule

- Bayes rule is a useful way to manipulate conditional probabilities:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

- So we can rewrite the above as

$$\hat{t}_1^N = \arg\max_{t_1^n} \frac{P(w_1^N|t_1^N)P(t_1^N)}{P(w_1^N)}$$

Bayes rule

- Bayes rule is a useful way to manipulate conditional probabilities:
$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$
- So we can rewrite the above as
$$\hat{t}_1^N = \arg\max_{t_1^n} \frac{P(w_1^N|t_1^N)P(t_1^N)}{P(w_1^N)}$$
- To find the most likely tag sequence involves comparing probabilities, given the same word sequence: hence $P(w_1^N)$ does not change and we can write:
$$\hat{t}_1^N = \arg\max_{t_1^n} P(w_1^N|t_1^N)P(t_1^N)$$

Prior and likelihood

- $P(w_1^N | t_1^N)$ is called the likelihood

Prior and likelihood

- $P(w_1^N | t_1^N)$ is called the likelihood
- $P(t_1^N)$ is called the prior

Prior and likelihood

- $P(w_1^N|t_1^N)$ is called the likelihood
- $P(t_1^N)$ is called the prior
- We need some simplifying assumptions to estimate these terms

Prior and likelihood

- $P(w_1^N|t_1^N)$ is called the likelihood
- $P(t_1^N)$ is called the prior
- We need some simplifying assumptions to estimate these terms
- (1) Likelihood: assume that the probability of a word depends only on its tag; independent of surrounding words and their tags:

$$P(w_1^N|t_1^N) \sim \prod_{i=1}^{N} P(w_i|t_i)$$

Prior and likelihood

- $P(w_1^N | t_1^N)$ is called the likelihood
- $P(t_1^N)$ is called the prior
- We need some simplifying assumptions to estimate these terms
- (1) Likelihood: assume that the probability of a word depends only on its tag; independent of surrounding words and their tags:

$$P(w_1^N | t_1^N) \sim \prod_{i=1}^{N} P(w_i | t_i)$$

- (2) Prior: use an n-gram assumption (eg bigram):

$$P(t_1^N) \sim \prod_{i=1}^{N} P(t_i | t_{i-1})$$

Putting it together

- Bigram part-of-speech tagger computes the following to estimate the most likely tag sequence:

$$\hat{t}_1^N = \arg\max_{t_1^n} P(t_1^N | w_1^N)$$
$$\sim \prod_{i=1}^{N} P(w_i | t_i) P(t_i | t_{i-1})$$

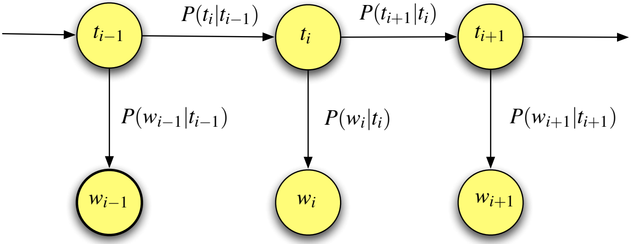
Putting it together

- Bigram part-of-speech tagger computes the following to estimate the most likely tag sequence:

$$\hat{t}_1^N = \arg\max_{t_1^n} P(t_1^N | w_1^N)$$
$$\sim \prod_{i=1}^{N} P(w_i|t_i) P(t_i|t_{i-1})$$

- For each word take the product of the word likelihood (given the tag) and the tag transition probability

HMM representation



Training and decoding

- Problem 1: Estimate the probability tables $P(w_i|t_i)$ and $P(t_i|t_{i-1})$ (**Training**)

Training and decoding

- Problem 1: Estimate the probability tables $P(w_i|t_i)$ and $P(t_i|t_{i-1})$ (**Training**)
- Problem 2: Given the probability tables and a sequence of words, what is the most likely sequence of tags (**Decoding**)

Training: Estimating the probabilities

- Use maximum likelihood to estimate the tag transition and word probabilities by computing a ratio of counts:

$$P'(t_i|t_{i-1}) = \frac{c(t_{i-1}, t_i)}{c(t_{i-1})}$$

$$P'(w_i|t_i) = \frac{c(t_i, w_i)}{c(t_i)}$$

Training: Estimating the probabilities

- Use maximum likelihood to estimate the tag transition and word probabilities by computing a ratio of counts:

$$P'(t_i|t_{i-1}) = \frac{c(t_{i-1}, t_i)}{c(t_{i-1})}$$

$$P'(w_i|t_i) = \frac{c(t_i, w_i)}{c(t_i)}$$

- Example: estimate of $P(NN|DT)$ in the treebank:

$$P'(NN|DT) = \frac{c(DT, NN)}{c(DT)} = \frac{56\,509}{116\,454} = 0.49$$

Training: Estimating the probabilities

- Use maximum likelihood to estimate the tag transition and word probabilities by computing a ratio of counts:

$$P'(t_i|t_{i-1}) = \frac{c(t_{i-1}, t_i)}{c(t_{i-1})}$$

$$P'(w_i|t_i) = \frac{c(t_i, w_i)}{c(t_i)}$$

- Example: estimate of $P(NN|DT)$ in the treebank:

$$P'(NN|DT) = \frac{c(DT, NN)}{c(DT)} = \frac{56\,509}{116\,454} = 0.49$$

- Example: estimate of $P(\text{is}|VBZ)$:

$$P'(\text{is}|VBZ) = \frac{c(VBZ, \text{is})}{c(VBZ)} = \frac{10\,073}{21\,627} = 0.47$$

Dealing with ambiguity (example from J&M)

Secretariat/NNP is/VBZ expected/VBZ to/TO race/VB tomorrow/NN

People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT race/NN for/IN outer/JJ space/NN

Dealing with ambiguity (example from J&M)

Secretariat/NNP is/VBZ expected/VBZ to/TO race/VB tomorrow/NN

People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT race/NN for/IN outer/JJ space/NN

- "race" is a verb in the first, a noun in the second.

Dealing with ambiguity (example from J&M)

Secretariat/NNP is/VBZ expected/VBZ to/TO race/VB tomorrow/NN

People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT race/NN for/IN outer/JJ space/NN

- "race" is a verb in the first, a noun in the second.
- Assume that race is the only untagged word, so we can assume the tags of the others.

Dealing with ambiguity (example from J&M)

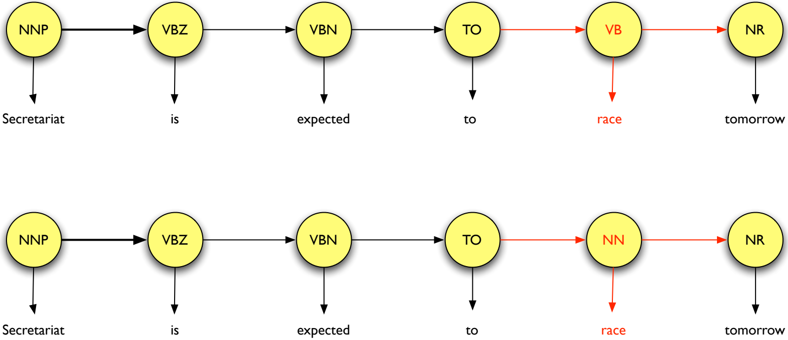
Secretariat/NNP is/VBZ expected/VBZ to/TO race/VB tomorrow/NN

People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT race/NN for/IN outer/JJ space/NN

- "race" is a verb in the first, a noun in the second.
- Assume that race is the only untagged word, so we can assume the tags of the others.
- Probabilities of "race" being a verb, or race being a noun in the first example:

$$P(\text{race is } VB) = P(\text{race}|VB)P(VB|TO)$$
$$P(\text{race is } NN) = P(\text{race}|NN)P(NN|TO)$$

HMM representation



Example (continued)

$$P(NN|TO) = 0.021$$
$$P(VB|TO) = 0.34$$

$$P(\text{race}|NN) = 0.00041$$
$$P(\text{race}|VB) = 0.00003$$

$$\begin{aligned} P(\text{race is } VB) &= P(\text{race}|VB)P(VB|TO) \\ &= 0.00003 \times 0.34 = 0.00001 \\ P(\text{race is } NN) &= P(\text{race}|NN)P(NN|TO) \\ &= 0.00041 \times 0.021 = 0.000007 \end{aligned}$$

Simple bigram tagging in NLTK

```
>>> default_pattern = (r'.*', 'NN')
>>> cd_pattern = (r' ^[0-9]+(.[0-9]+)?$', 'CD')
>>> patterns = [cd_pattern, default_pattern]
>>> NN_CD_tagger = tag.Regexp(patterns)
>>> unigram_tagger = tag.Unigram(cutoff=0, backoff=NN_CD_tagger)
>>> unigram_tagger.train(train_sents)
>>> bigram_tagger = tag.Bigram(backoff=unigram_tagger)
>>> bigram_tagger.train(train_sents)

# uses print_accuracy function defined in lecture PoS1
>>> print_accuracy(bigram_tagger, train_sents)
95.6%
>>> print_accuracy(bigram_tagger, test_sents)
84.2%
```

Limitation of NLTK n-gram taggers

- Does not find the most likely sequence of tags, simply works left to right always assigning the most probable single tag (given the previous tag assignments)
- Does not cope with zero probability problem well (no smoothing or discounting)

Limitation of NLTK n-gram taggers

- Does not find the most likely sequence of tags, simply works left to right always assigning the most probable single tag (given the previous tag assignments)
- Does not cope with zero probability problem well (no smoothing or discounting)
- (see module `nltk_lite.tag.hmm`)
- Next lecture: Viterbi algorithm—efficiently decoding the most likely sequence of tags given the words

Summary

- **Reading:**
  - Jurafsky and Martin, chapter 8 (esp. sec 8.5);
  - Manning and Schütze, chapter 10;
- HMMs and n-grams for statistical tagging
- Operation of a simple bigram tagger