Machine Learning for Language Processing
ACS 2015/16
Stephen Clark

L2: POS Tagging with HMMs

UNIVERSITY OF CAMBRIDGE

The POS Tagging Problem

```
England|NNP 's|POS fencers|NNS won|VBD gold|NN on|IN
day|NN 4|CD in|IN Delhi|NNP with|IN a|DT medal|JJ
-winning|JJ performance|NN .|.

This|DT is|VBZ Dr.|NNP Black|NNP 's|POS second|JJ
gold|NN of|IN the|DT Games|NNP .|.
```

- Problem is difficult because of ambiguity

Probabilistic Formulation

$$y^* = \arg\max_{y \in Y} P(y|x)$$

where $x = (x_1, \ldots, x_n)$ is a sentence and $y = (y_1, \ldots, y_n) \in Y$ is a possible tag sequence for x

- Two problems:
  - where do the probabilities come from? (age-old question in statistical approaches to AI)
  - how do we find the arg max?
- Problem 1 is the problem of *model estimation*
- Problem 2 is the *search problem*

HMM Tagging Model

- $P(T|W) = \frac{P(W|T)P(T)}{P(W)}$ (Bayes theorem)
- $\arg\max_T P(T|W) = \arg\max_T P(W|T)P(T)$ (W is constant)
- Using Chain Rule and (Markov) independence assumptions:

$$
\begin{aligned}
P(W|T) &= P(w_1, \ldots, w_n | t_1, \ldots, t_n) \\
&= P(w_1|t_1, \ldots, t_n) P(w_2|w_1, t_1, \ldots, t_n) P(w_3|w_2, w_1, t_1, \ldots, t_n) \\
&= P(w_n|w_{n-1}, \ldots, w_1, t_1, \ldots, t_n) \\
&\approx \prod_{i=1}^{n} P(w_i|t_i)
\end{aligned}
$$

$$
\begin{aligned}
P(T) &= P(t_1, \ldots, t_n) \\
&= P(t_1) P(t_2|t_1) P(t_3|t_2, t_1) \ldots P(t_n|t_{n-1}, \ldots, t_1) \\
&\approx \prod_{i=1}^{n} P(t_i|t_{i-1})
\end{aligned}
$$

N-gram Generative Taggers

- A tagger which conditions on the previous tag is called a **bigram** tagger
- Trigram taggers are typically used (condition on previous 2 tags)
- HMM taggers use a **generative** model, so-called because the tags *and* words can be thought of as being generated according to some stochastic process
- More sophisticated **discriminative** models (e.g. CRFs) can condition on more aspects of the context, e.g. suffix information

Parameter Estimation

- Two sets of parameters:
  - $P(t_i|t_{i-1})$ tag transition probabilities
  - $P(w_i|t_i)$ word emission probabilities
- Note *not* $P(t_i|w_i)$ (reversed because of use of Bayes theorem)
  - one of the original papers on stochastic POS tagging reportedly got this wrong
- Estimation based on counting from manually labelled corpora
  - so we have a *supervised* machine learning approach
- For this problem, simple counting (relative frequency) method gives *maximum likelihood* estimates

Relative Frequency Estimation

- $\hat{P}(t_i|t_{i-1}) = \frac{f(t_{i-1},t_i)}{f(t_{i-1})}$
  - where $f(t_{i-1}, t_i)$ is the number of times t_i follows t_{i-1} in the training data; and $f(t_{i-1})$ is the number of times t_{i-1} appears in the data
- $\hat{P}(w_i|t_i) = \frac{f(w_i,t_i)}{f(t_i)}$
  - where $f(w_i, t_i)$ is the number of times w_i has tag t_i in the training data
- It turns out that for an HMM the intuitive relative frequency estimates are the estimates which *maximise the probability of the training data*
- What if the numerator (or denominator) is zero?

Smoothing for Tagging

- Tag sequence probabilities can be smoothed (or backed off):

$$\tilde{P}(t_i|t_{i-1}, t_{i-2}) = \lambda_1 \hat{P}(t_i|t_{i-1}, t_{i-2})$$
$$+(1-\lambda_1)(\lambda_2 \hat{P}(t_i|t_{i-1}) + (1-\lambda_2)\hat{P}(t_i))$$

- A simple solution for unknown words is to replace them with UNK:

$$P(w_i|t_i) = P(\textsf{UNK}|t_i)$$

where any word in the training data occurring less than, say, 5 times is replaced with UNK

Better Handling of Unknown Words

- Lots of clues as to what the tag of an unknown word might be:
  - proper nouns (NNP) likely to be unknown
  - if the word ends in *ing*, likely to be VBG
  - ...

$$P(w|t) = \frac{1}{Z} P(\text{unknown word}|t) P(\text{capitalized}|t) P(\text{endings}|t)$$

- but now we're starting to see the weaknesses of generative models for taggers
- *Conditional* models can deal with these features directly

The Search Problem for Tagging

$$T^* = \arg\max_T P(T|W) = \arg\max_T P(W|T)P(T)$$

- Number of tag sequences for a sentence of length n is $O(T^n)$ where T is the size of the tagset
- OK, but why is there a non-trivial search problem?
  - e.g. for a unigram model we can just take the most probable tag for each word, an algorithm which runs in $O(nT)$ time

A Non-Trivial Search Problem

$$T^* = \arg\max_T P(T|W) = \arg\max_T P(W|T)P(T)$$

- But what about a bigram model?
- Intuition: suppose I have two competing tags for word w_i, t_i^1 and t_i^2
- Compare:

$$\text{Score}(t_i^1) = P(t_i^1|t_{i-1})P(w_i|t_i^1)$$
$$\text{Score}(t_i^2) = P(t_i^2|t_{i-1})P(w_i|t_i^2)$$

Suppose $\text{Score}(t_i^1) > \text{Score}(t_i^2)$; can we be sure t_i^1 is part of the highest scoring tag *sequence*?

The Viterbi Algorithm

- Dynamic Programming (DP) algorithm, so requires the "optimal sub-problem property"
  - i.e. optimal solution to the complete problem can be defined in terms of optimal solutions to sub-problems
- So what are the sub-problems in this case?
  - intuition: suppose we want the optimal tag sequence ending at w_n, and we know the optimal sub-sequence ending at w_{n-1}, for all possible tags at w_{n-1}

Viterbi for a Bigram Tagger

$$\delta_{t_j}(n+1) = \max_{t_i} \delta_{t_i}(n) P(t_j|t_i) P(w_{n+1}|t_j)$$

where $\delta_{t_j}(n+1)$ is the probability of the most probable tag sequence ending in tag t_j at position $n+1$

- Recursion bottoms out at position 1 in the sentence
- Most probable tag sequence can be obtained by following the recursion from the right backwards
- Time complexity is $O(T^2 n)$ where T is the size of the tagset

Practicalities

- Choice of tags to be assigned to a particular word usually governed by a “tag dictionary”
- Accuracy measured by taking a manually created “gold-standard” for a set of held-out test sentences
- Accuracy of POS taggers on newspaper data is over 97%, close to the upper bound represented by human agreement (and existence of noise in the data)
- Linear time process (in length of sentence) means tagging can be performed very fast, e.g. hundreds of thousands of words per second