PART OF SPEECH TAGGING USING HIDDEN MARKOV MODELS

Stephen Pulman
stephen.pulman@comlab.ox.ac.uk

- What is tagging?
- What are (Hidden) Markov Models?
- Outline of a HMM POS tagger
- The Viterbi algorithm
- Appendix: Bayes's Theorem in a nutshell

References

Allen, Natural Language Understanding, Chapter 7

Jurafsky and Martin, Section 8.5 (first edition)

Jurafsky and Martin, Chapter 5 (second edition)

PART OF SPEECH TAGGING

i.e. assigning a part of speech to the words in a sentence:
the/dt cat/nn sat/vbd on/p the/dt mat/nn ./.

Why?

- useful preprocessing step for later parsing, since it reduces ambiguity:

  time flies (two+ readings)
  How the time/nn flies
  I want you to time/vb flies

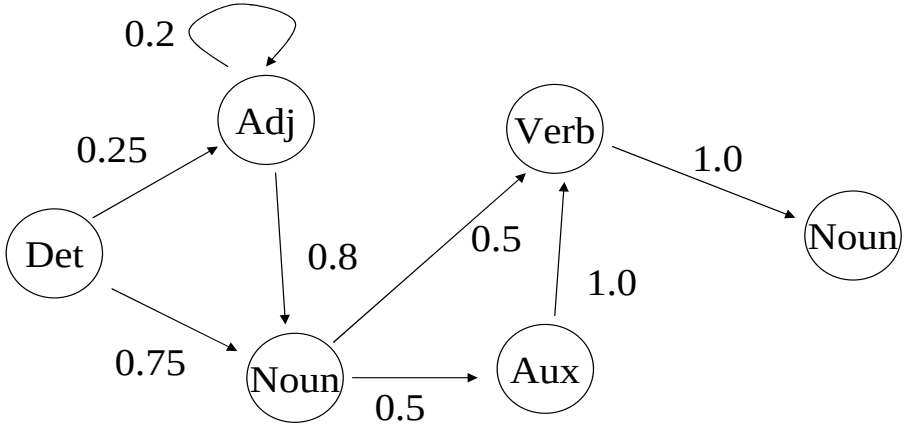
- enrich a corpus with useful information: 'find all occurrences of 'time' as a verb, followed by a noun'

- can help guess categories of unknown words:

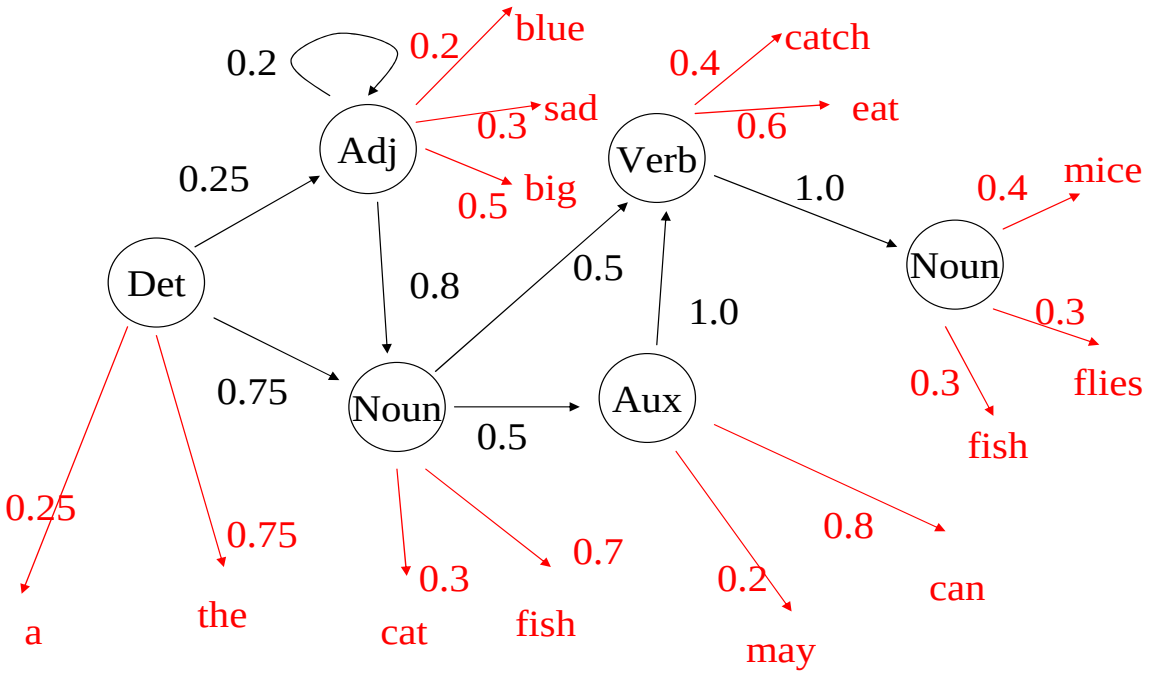
  He sashayed into the room (vs. this is a sashayed design from the Ming period)
  It's the latest wonderful gruntfuttock from Microsoft

Markov Models (aka Chains, Processes)

A Markov Model is essentially a finite state machine with probabilities on the transitions:



A **Hidden Markov Model** is a Markov model which also emits symbols when in a particular state: each symbol having a probability of being emitted:



Hidden Markov Models are so called because in their usual application, we are given a sequence of output symbols and we must decide which sequence of states was most likely to have emitted that sequence. In speech recognition, for example, the 'symbols' are representations of sounds, and the states we are trying to recover are the words of which these sounds are pronunciations.

In our case, we have a sequence of words, and we want to know their parts of speech *in that sequence*. We conceptualise this as an HMM problem by regarding each such word sequence as the output of a traversal through a sequence of 'part of speech' (POS) states.

Now, if we can build such an HMM, it gives us the probability of emitting a word when in a POS state, and also the probability of making a transition from one POS state to the next. But how do we reverse this - work out the most probable sequence of POS states from the output sequence? We use Bayes' Theorem.

Statistical POS tagging with Hidden Markov Models

We want to find $P(C_1,...,C_n \mid W_1,...,W_n)$. **Bayes' Theorem** tells us that this is the same as:

$$\frac{P(C_1...C_n) * P(W_1...W_n \mid C_1...C_n)}{P(W_1...W_n)}$$

We want to find the sequence of Cs that give us the highest value in this equation. Notice that the value of $P(W_1...W_n)$ will be the same for all sequences (the words are given) so we can ignore it. But the top line is still too much to calculate directly, since we would never have enough data to estimate the probabilities from.

The chain rule:

$$P(A,B) = P(A|B)*P(B)$$
$$P(A,B,C) = P(A|B,C)*P(B|C)*P(C)$$
$$P(A,B,C,D) = P(A|B,C,D)*P(B|C,D)*P(C|D)*P(D)$$
$$P(C_1...C_n) = P(C_1|C_2...C_n)*P(C_2|C_3...C_n)*...*P(C_n)$$

So we can approximate P(C_1,...,C_n) (or rather, P(C_n,...,C_1) - we really need to apply the chain rule in the other direction) as:

$\prod_{i=1}^{n} P(C_i|C_{i-1})$
('bigram' case)

or

$\prod_{i=1}^{n} P(C_i|C_{i-1},C_{i-2})$ ('trigram' case)

We just ignore all but one or two of the conditioning variables in:

P(C_1...C_n) = P(C_1|C_2...C_n)*P(C_2|C_3...C_n)*...*P(C_n)

If we had enough training data we could use longer 'n-grams'.

We usually invent a start and end sentence marker so that C_{i-1} exists even where i=1.

Bigram tagger:

Calculate P($C_1 \ldots C_n$) approximately by multiplying the values for i=1 to n of P($C_i|C_{i-1}$)

Calculate P($W_1 \ldots W_n|C_1 \ldots C_n$) approximately by multiplying the values for i=1 to n of P($W_i|C_i$).

So the revised formula for the bigram case is: find the sequence of Cs that gives the highest value for

$$\prod_{i=1}^{n} P(C_i \mid C_{i-1}) * P(W_i \mid C_i)$$

We can gather these statistics from an already tagged corpus:

$$\text{P(W|C)} = \frac{\text{number of times W appears as a C}}{\text{number of times any word appears as a C}}$$

$$P(C_i|C_{i-1}) = \frac{\text{number of times the sequence } ...C_{i-1},C_i... \text{ appears}}{\text{number of times } ...C_{i-1},X_i... \text{ appears, where X is any category at all}}$$

$$P(C_i|C_{i-1},C_{i-2}) = \frac{\text{number of times the sequence } C_{i-2},C_{i-1},C_i \text{ appears}}{\text{number of times } C_{i-2},C_{i-1},X_i \text{ appears, where X is any category at all.}}$$

and so on.

There are various suitable corpora around: the Wall Street Journal corpus (UPenn), the British National Corpus (Oxford Text Archive), LOB (Lancaster), Suzanne (Oxford Text Archive) etc.

The Viterbi Algorithm

How can we compute this efficiently, given that for an N word long sentence, with an average of M POS cats per word, there will be M^N possible sequences? The Viterbi algorithm is a dynamic programming algorithm that allows us to compute the best sequence ('path'), discarding others as we go along.

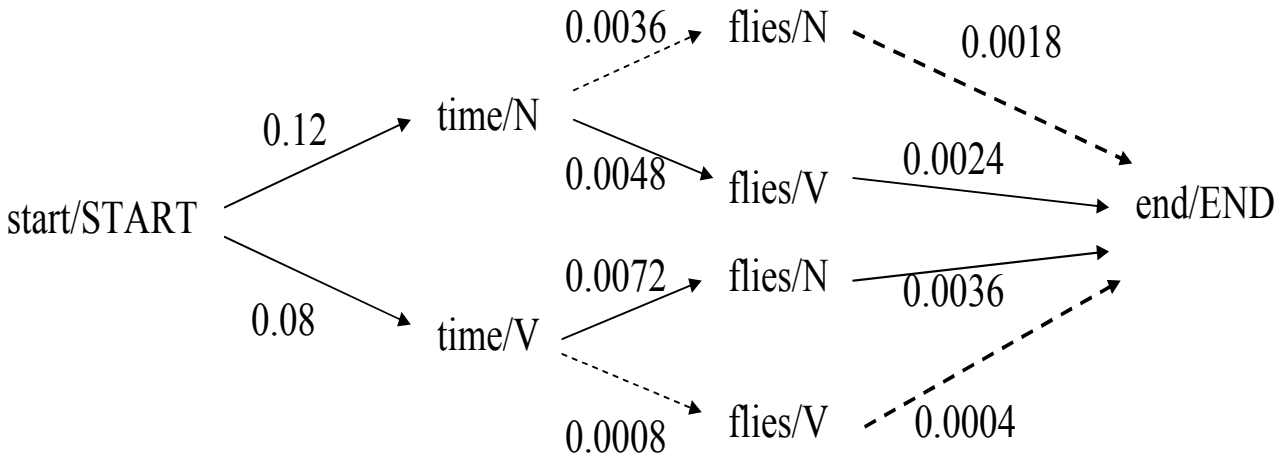
for each Word $W_{1...n}$
  for each POS category C_j that W_i can have,
    for each path ending in some C_k for W_{i-1}
      compute $P(C_j|C_k)$*$P(W_i|C_j)$*score of path
      keep a record of best scoring path to C_j

Some invented probabilities, for the sentence:

$\langle start \rangle$ time flies $\langle end \rangle$

(assume $\langle$start$\rangle$ and $\langle$end$\rangle$ pseudo-words, with tags START and END)

P(W|C):		P(C_i|C_{i-1}):	
P(time|N)	=0.2	P(N|start)	= 0.6
P(time|V)	=0.4	P(V|start)	= 0.2
P(flies| N)	=0.3	P(N|V)	= 0.3
P(flies|V)	=0.1	P(V|N)	= 0.4
P(start|START)	= 1	P(N|N)	= 0.1
P(end|END)	= 1	P(V|V)	= 0.1
etc.		P(END|N)	= 0.5
		P(END|V)	= 0.5


start/START
0.12
0.08
time/N
time/V
0.0036
0.0048
0.0072
0.0008
flies/N
flies/V
flies/N
flies/V
0.0018
0.0024
0.0036
0.0004
end/END

Implementing the Viterbi Algorithm

For a sentence $W_1 \ldots W_m$ M words long, assuming a set $C_1 \ldots C_n$ of N distinct part of speech tags, create an MxN array called 'Score', and another MxN array called 'Backpointer'.

Initialise:

For i from 1 to N
    Score(i,1) = $P(W_1|C_i) * P(C_i|\langle start \rangle)$
    Backpointer(i,1) = 0

Run:

For j from 2 to M
  For i from 1 to N
    Score(i,j) = $\max_{k=1 \ldots N}$(Score(k,j-1)*$P(C_i|C_k)$ * $P(W_j|C_i)$)
    Backpointer(i,j) = max k from previous line

Recover best sequence:

C_M = $\max_i$ Score(i,M)
For i from M-1 to 1
    C_i = Backpointer(C_{i+1},i+1)

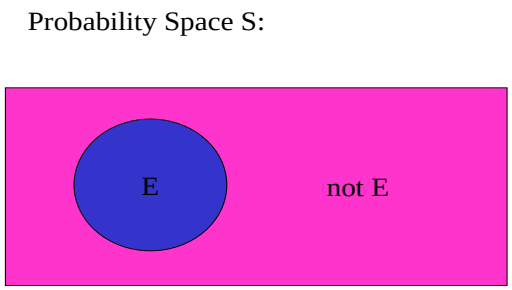
Appendix: Bayes' Theorem in a nutshell

Assume a 'possibility space' S of equally likely outcomes ('events'). Then the probability of an event E is defined as

P(E) = number of times E occurs/number of events in S

- always a number between 0 and 1.

This is easy to visualise:

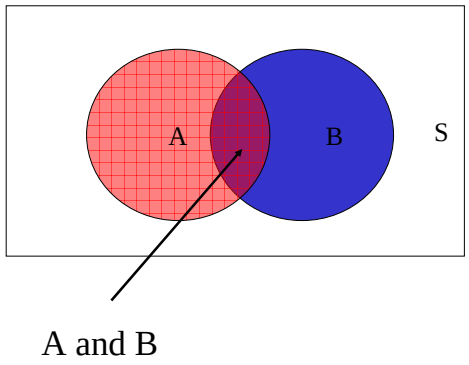


E + not E = S
P(E) = count(E)/count(S)
P(not E) = 1-P(E)
P(E) + P(not E) = 1

If we have two events A and B, with non-zero probabilities, then:

P(A or B) = P(A) + P(B) - P(A and B)

i.e. the probability of each occurring separately added together, less the probability of them occurring jointly (otherwise there would be some double counting). Note that the joint probability P(A and B) is usually written P(A,B).



Conditional Probability

If A and B have non-zero probabilities, then the conditional probability P(A|B) (i.e. the probability of A given that B has already occurred) is:

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

This is easy to see from the previous picture: it's effectively the ratio of the number of events in the intersection of A and B (i.e. the A events where B has already occurred) to the number of B events (with or without A).

(NB If A and B are independent, then P(A|B)=P(A). So then P(A,B)=P(A)*P(B)).

Bayes' Theorem

We just saw that $P(A|B) = \frac{P(A,B)}{P(B)}$

So by rearranging, we get that $P(A,B) = P(A|B)*P(B)$

If B is conditional on A, then $P(B|A) = \frac{P(B,A)}{P(A)}$

Rearranging this too we get that $P(B,A)=P(B|A)*P(A)$

But obviously $P(A,B)=P(B,A)$

So $P(A|B)*P(B)=P(B|A)*P(A)$

And so $P(A|B)=\frac{P(B|A)*P(A)}{P(B)}$

This is **Bayes' Theorem**. P(A) and P(B) are often called the 'prior' probabilities of A and B. P(A|B) is called the 'posterior' probability. P(B|A) is called the 'likelihood'.