

Lecture 10: Algorithms for HMMs

Nathan Schneider

(some slides from Sharon Goldwater;
thanks to Jonathan May for bug fixes)

ENLP | 28 February 2023

Recap: tagging

- POS tagging is a sequence labelling task.
- We can tackle it with a model (HMM) that uses two sources of information:
 - The word itself
 - The tags assigned to surrounding words
- The second source of information means we can't just tag each word independently.

Local Tagging

Words:

<s>	one	dog	bit	</s>
-----	-----	-----	-----	------

Possible tags:
(ordered by
frequency for
each word)

<s>	CD	NN	NN	</s>
	NN	VB	VBD	
	PRP			

- Choosing the best tag for each word independently, i.e. not considering tag context, gives the wrong answer (<s> CD NN NN </s>).
- Though NN is more frequent for 'bit', tagging it as VBD may yield a better *sequence* (<s> CD NN VB </s>)
 - because $P(\text{VBD} | \text{NN})$ and $P(\text{</s>} | \text{VBD})$ are high.

Recap: HMM

- Elements of HMM:
 - Set of states (tags)
 - Output alphabet (word types)
 - Start state (beginning of sentence)
 - State transition probabilities $P(t_i / t_{i-1})$
 - Output probabilities from each state $P(w_i / t_i)$

Recap: HMM

- Given a sentence $\mathbf{W}=w_1\dots w_n$ with tags $\mathbf{T}=t_1\dots t_n$, compute $P(\mathbf{W},\mathbf{T})$ as:

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

- But we want to find $\operatorname{argmax}_{\mathbf{T}} P(\mathbf{T}|\mathbf{W})$ without enumerating all possible tag sequences \mathbf{T}
 - Use a greedy approximation, or
 - Use Viterbi algorithm to store partial computations.

Greedy Tagging

Words:

Possible tags:
(ordered by
frequency for
each word)

<s>	one	dog	bit	</s>
<s>	CD	NN	NN	</s>
	NN	VB	VBD	
	PRP			

- For $i = 1$ to N : choose the tag that maximizes
 - transition probability $P(t_i|t_{i-1}) \times$
 - emission probability $P(w_i|t_i)$
- This uses tag context but is still suboptimal. Why?
 - It commits to a tag before seeing subsequent tags.
 - It could be the case that ALL possible next tags have low transition probabilities. E.g., if a tag is unlikely to occur at the end of the sentence, that is disregarded when going left to right.

Greedy vs. Dynamic Programming

- The greedy algorithm is **fast**: we just have to make one decision per token, and we're done.
 - Runtime complexity?
 - $O(TN)$ with T tags, length- N sentence
- But subsequent words have no effect on each decision, so the result is likely to be **suboptimal**.
- Dynamic programming search gives an **optimal global** solution, but requires some bookkeeping (= more computation). Postpones decision about any tag until we can be sure it's optimal.

Viterbi Tagging: intuition

Words:

<s>	one	dog	bit	</s>
-----	-----	-----	-----	------

Possible tags:
(ordered by
frequency for
each word)

<s>	CD	NN	NN	</s>
	NN	VB	VBD	
	PRP			

- Suppose we have already computed
 - a) The best tag sequence for <s> ... bit that ends in NN.
 - b) The best tag sequence for <s> ... bit that ends in VBD.
- Then, the best full sequence would be either
 - sequence (a) extended to include </s>, or
 - sequence (b) extended to include </s>.

Viterbi Tagging: intuition

Words:	<s>	one	dog	bit	</s>
Possible tags: (ordered by frequency for each word)	<s>	CD	NN	NN	</s>
		NN	VB	VBD	
		PRP			

- But similarly, to get
 - a) The best tag sequence for <s> ... bit that ends in NN.
- We could extend one of:
 - The best tag sequence for <s> ... dog that ends in NN.
 - The best tag sequence for <s> ... dog that ends in VB.
- And so on...

Viterbi: high-level picture

- Want to find $\operatorname{argmax}_{\mathbf{T}} P(\mathbf{T}|\mathbf{W})$
- Intuition: the best path of length i ending in state t must include the best path of length $i-1$ to the previous state. So,
 - Find the best path of length $i-1$ to each state.
 - Consider extending each of those by 1 step, to state t .
 - Take the best of those options as the best path to state t .

Viterbi algorithm

- Use a **chart** to store partial results as we go
 - $T \times N$ table, where $v(t, i)$ is the probability* of the best state sequence for $w_1 \dots w_i$ that ends in state t .

*Specifically, $v(t, i)$ stores the max of the joint probability $P(w_1 \dots w_i, t_1 \dots t_{i-1}, t_i = t | \lambda)$

Viterbi algorithm

- Use a **chart** to store partial results as we go
 - $T \times N$ table, where $v(t, i)$ is the probability* of the best state sequence for $w_1 \dots w_i$ that ends in state t .
- Fill in columns from left to right, with
$$v(t, i) = \max_{t'} v(t', i - 1) \cdot P(t|t') \cdot P(w_i|t_i)$$
 - The max is over each possible previous tag t'
- Store a **backtrace** to show, for each cell, which state at $i - 1$ we came from.

*Specifically, $v(t, i)$ stores the max of the joint probability $P(w_1 \dots w_i, t_1 \dots t_{i-1}, t_i = t | \lambda)$

Transition and Output Probabilities

Transition matrix: $P(t_i | t_{i-1})$:

	Noun	Verb	Det	Prep	Adv	</s>
<s>	.3	.1	.3	.2	.1	0
Noun	.2	.4	.01	.3	.04	.05
Verb	.3	.05	.3	.2	.1	.05
Det	.9	.01	.01	.01	.07	0
Prep	.4	.05	.4	.1	.05	0
Adv	.1	.5	.1	.1	.1	.1

Emission matrix: $P(w_i | t_i)$:

	a	cat	doctor	in	is	the	very
Noun	0	.5	.4	0	.1	0	0
Verb	0	0	.1	0	.9	0	0
Det	.3	0	0	0	0	.7	0
Prep	0	0	0	1.0	0	0	0
Adv	0	0	0	.1	0	0	.9

Example

Suppose W =the doctor is in. Our initially empty table:

v	w_1 =the	w_2 =doctor	w_3 =is	w_4 =in	$\langle /s \rangle$
Noun					
Verb					
Det					
Prep					
Adv					

Filling in the first column

Suppose W =the doctor is in. Our initially empty table:

v	w_1 =the	w_2 =doctor	w_3 =is	w_4 =in	$\langle /s \rangle$
Noun	0				
Verb	0				
Det	.21				
Prep	0				
Adv	0				

$$v(\text{Noun, the}) = P(\text{Noun}|\langle s \rangle)P(\text{the}|\text{Noun})=.3(0)$$

$$v(\text{Det, the}) = P(\text{Det}|\langle \ddot{s} \rangle) P(\text{the}|\text{Det})=.3(.7)$$

The second column

$v(\text{Noun, doctor})$

$$= \max_{t'} v(t', \text{the}) \cdot P(\text{Noun}|t') \cdot P(\text{doctor}|\text{Noun})$$

v	$w_1=\text{the}$	$w_2=\text{doctor}$	$w_3=\text{is}$	$w_4=\text{in}$	$\langle /s \rangle$
Noun	0	?			
Verb	0				
Det	.21				
Prep	0				
Adv	0				

$$P(\text{Noun}|\text{Det}) P(\text{doctor}|\text{Noun}) = .3(.4)$$


The second column

$v(\text{Noun}, \text{doctor})$

$$= \max_{t'} v(t', \text{the}) \cdot P(\text{Noun}|t') \cdot P(\text{doctor}|\text{Noun})$$

$$= \max \{ 0, 0, .21(.36), 0, 0 \} = .0756$$

v	$w_1=\text{the}$	$w_2=\text{doctor}$	$w_3=\text{is}$	$w_4=\text{in}$	$\langle /s \rangle$
Noun	0	.0756			
Verb	0				
Det	.21				
Prep	0				
Adv	0				



$$P(\text{Noun}|\text{Det}) P(\text{doctor}|\text{Noun}) = .9(.4)$$


The second column

$v(\text{Verb}, \text{doctor})$

$$= \max_{t'} v(t', \text{the}) \cdot P(\text{Verb}|t') \cdot P(\text{doctor}|\text{Verb})$$

$$= \max \{ 0, 0, .21(.001), 0, 0 \} = .00021$$

v	$w_1=\text{the}$	$w_2=\text{doctor}$	$w_3=\text{is}$	$w_4=\text{in}$	$\langle /s \rangle$
Noun	0	.0756			
Verb	0	.00021			
Det	.21				
Prep	0				
Adv	0				



$$P(\text{Verb}|\text{Det}) P(\text{doctor}|\text{Verb}) = .01(.1)$$


The second column

$v(\text{Verb}, \text{doctor})$

$$= \max_{t'} v(t', \text{the}) \cdot P(\text{Verb}|t') \cdot P(\text{doctor}|\text{Verb})$$

$$= \max \{ 0, 0, .21(.001), 0, 0 \} = .00021$$

v	$w_1=\text{the}$	$w_2=\text{doctor}$	$w_3=\text{is}$	$w_4=\text{in}$	$\langle /s \rangle$
Noun	0	.0756			
Verb	0	.00021			
Det	.21	0			
Prep	0	0			
Adv	0	0			




$$P(\text{Verb}|\text{Det}) P(\text{doctor}|\text{Verb}) = .01(.1)$$

The third column

$v(\text{Noun}, \text{is})$

$$= \max_{t'} v(t', \text{doctor}) \cdot P(\text{Noun}|t') \cdot P(\text{is}|\text{Noun})$$
$$= \max \{ .0756(.02), .00021(.03), 0, 0, 0 \} = .001512$$

v	$w_1=\text{the}$	$w_2=\text{doctor}$	$w_3=\text{is}$	$w_4=\text{in}$	$\langle /s \rangle$
Noun	0	.0756	.001512		
Verb	0	.00021			
Det	.21	0			
Prep	0	0			
Adv	0	0			



$$P(\text{Noun}|\text{Noun}) P(\text{is}|\text{Noun}) = .2(.1) = .02$$

$$P(\text{Noun}|\text{Verb}) P(\text{is}|\text{Noun}) = .3(.1) = .03$$

The third column

$v(\text{Verb}, \text{is})$

$$= \max_{t'} v(t', \text{doctor}) \cdot P(\text{Verb}|t') \cdot P(\text{is}|\text{Verb})$$
$$= \max \{ .0756(.36), .00021(.045), 0, 0, 0 \} = .027216$$

v	$w_1=\text{the}$	$w_2=\text{doctor}$	$w_3=\text{is}$	$w_4=\text{in}$	$\langle /s \rangle$
Noun	0	.0756	.001512		
Verb	0	.00021	.027216		
Det	.21	0	0		
Prep	0	0	0		
Adv	0	0	0		

$$P(\text{Verb}|\text{Noun}) P(\text{is}|\text{Verb}) = .4(.9) = .36$$

$$P(\text{Verb}|\text{Verb}) P(\text{is}|\text{Verb}) = .05(.9) = .045$$

The fourth column

$v(\text{Prep}, \text{in})$

$$= \max_{t'} v(t', \text{is}) \cdot P(\text{Prep}|t') \cdot P(\text{in}|\text{Prep})$$

$$= \max \{ .001512(.3), .027216(.2), 0, 0, 0 \} = .005443$$

v	$w_1=\text{the}$	$w_2=\text{doctor}$	$w_3=\text{is}$	$w_4=\text{in}$	$\langle /s \rangle$
Noun	0	.0756	.001512	0	
Verb	0	.00021	.027216	0	
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0		

$$P(\text{Prep}|\text{Noun}) P(\text{in}|\text{Prep}) = .3(1.0)$$

$$P(\text{Prep}|\text{Verb}) P(\text{in}|\text{Prep}) = .2(1.0)$$

The fourth column

$v(\text{Prep}, \text{in})$

$$= \max_{t'} v(t', \text{is}) \cdot P(\text{Prep}|t') \cdot P(\text{in}|\text{Prep})$$

$$= \max \{ .000504(.004), .027216(.01), 0, 0, 0 \} = .000272$$

v	$w_1=\text{the}$	$w_2=\text{doctor}$	$w_3=\text{is}$	$w_4=\text{in}$	$\langle /s \rangle$
Noun	0	.0756	.001512	0	
Verb	0	.00021	.027216	0	
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	

$$P(\text{Adv}|\text{Noun}) P(\text{in}|\text{Adv}) = .04(.1)$$

$$P(\text{Adv}|\text{Verb}) P(\text{in}|\text{Adv}) = .1(.1)$$

End of sentence

$v(</s>)$

$$= \max_{t'} v(t', \text{in}) \cdot P(</s>|t')$$

$$= \max \{ 0, 0, 0, .005443(0), .000272(.1) \} = .0000272$$

v	$w_1=\text{the}$	$w_2=\text{doctor}$	$w_3=\text{is}$	$w_4=\text{in}$	$</s>$
Noun	0	.0756	.001512	0	$.0000272$ 2
Verb	0	.00021	.027216	0	
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	

$$P(</s>|\text{Prep})=0$$

$$P(</s>|\text{Adv})=.1$$

Completed Viterbi Chart

v	w_1 =the	w_2 =doctor	w_3 =is	w_4 =in	$\langle /s \rangle$
Noun	0	.0756	.001512	0	.000027 2
Verb	0	.00021	.027216	0	
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	

Following the Backtraces

v	w_1 =the	w_2 =doctor	w_3 =is	w_4 =in	$\langle /s \rangle$
Noun	0	.0756	.001512	0	.000027 2
Verb	0	.00021	.027216	0	
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	

The diagram illustrates backtraces from the end token $\langle /s \rangle$ to the word 'the'. Red arrows show the following path: $\langle /s \rangle$ to the 'Adv' row (value .000272), then to the 'Prep' row (value .005443), then to the 'Verb' row (value .027216), then to the 'Noun' row (value .001512), and finally to the w_2 column (value .0756). A blue arrow points from the $\langle /s \rangle$ column to the 'Adv' row. The 'Adv' cell is highlighted in light blue.

Following the Backtraces

v	w_1 =the	w_2 =doctor	w_3 =is	w_4 =in	</s>
Noun	0	.0756	.001512	0	.000027 2
Verb	0	.00021	.027216	0	
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	

The diagram illustrates backtraces from the end token '</s>' to the words 'the' and 'is'. Red arrows show the path from '</s>' to 'the' (via the Det row) and from '</s>' to 'is' (via the Verb row). A blue arrow shows the path from '</s>' to 'is' (via the Adv row). The cell for 'is' in the Verb row is highlighted in light blue.

Following the Backtraces

v	w_1 =the	w_2 =doctor	w_3 =is	w_4 =in	$\langle /s \rangle$
Noun	0	.0756	.001512	0	.000027 2
Verb	0	.00021	.027216	0	
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	

The diagram illustrates backtraces from the end token $\langle /s \rangle$ to the word 'the' (w_1). Red arrows show the path: $\langle /s \rangle$ to the 'Det' row (value .21), then to the 'Noun' row (value .0756). Blue arrows show the path: $\langle /s \rangle$ to the 'Verb' row (value .027216), then to the 'Noun' row (value .0756). The 'Noun' row and 'Verb' row are highlighted in light blue.

Following the Backtraces

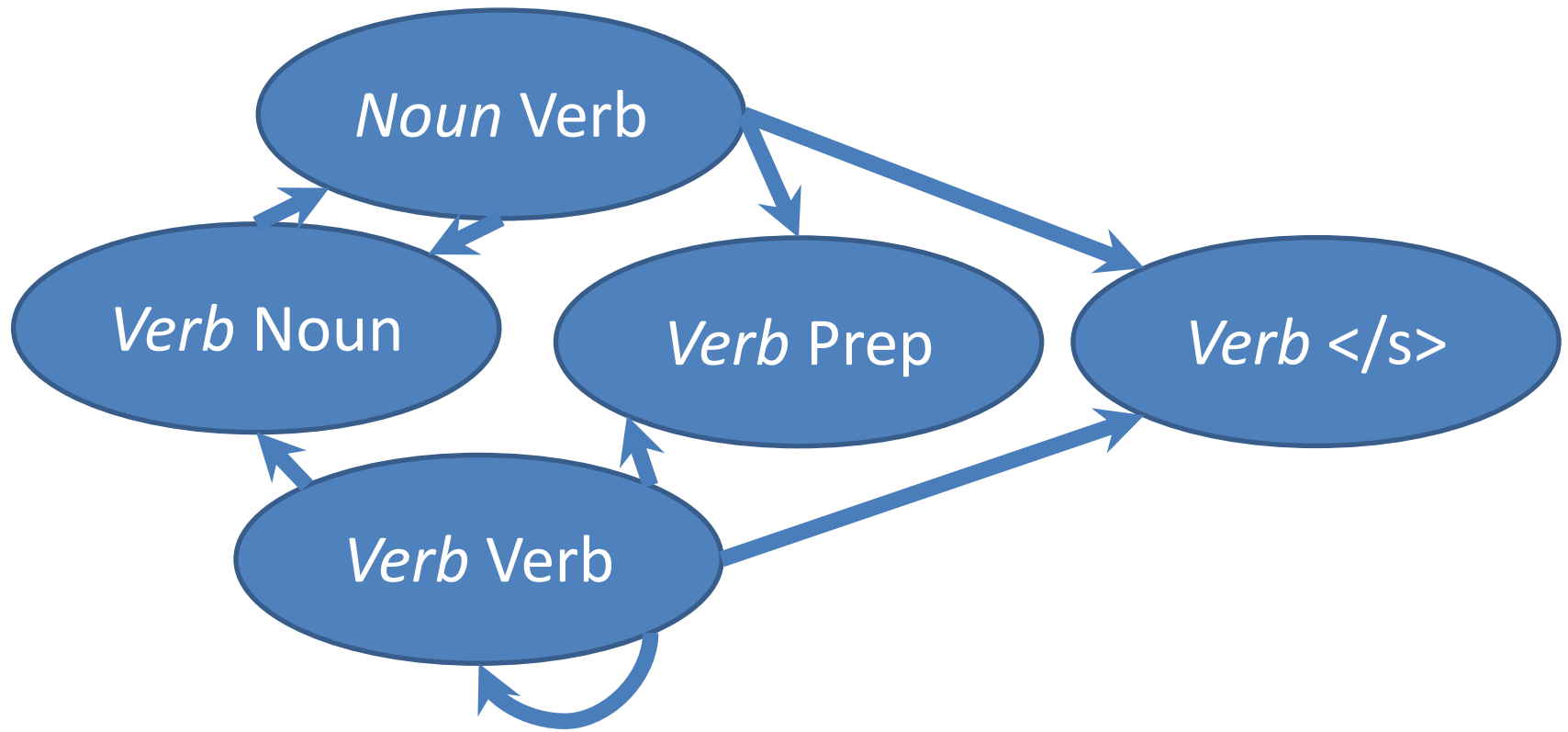
v	w_1 =the	w_2 =doctor	w_3 =is	w_4 =in	</s>
Noun	0	.0756	.001512	0	.000027 2
Verb	0	.00021	.027216	0	
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	
	Det	Noun	Verb	Prep	

Implementation and efficiency

- For sequence length N with T possible tags,
 - Enumeration takes $O(T^N)$ time and $O(N)$ space.
 - Bigram Viterbi takes $O(T^2N)$ time and $O(TN)$ space.
 - Viterbi is exhaustive: further speedups might be had using methods that prune the search space.
- As with N-gram models, chart probs get really tiny really fast, causing underflow.
 - So, we use **costs** (neg log probs) instead.
 - Take minimum over sum of costs, instead of maximum over product of probs.

Higher-order Viterbi

- For a tag **trigram** model with T possible tags, we effectively need T^2 states
 - n -gram Viterbi requires T^{n-1} states, takes $O(T^n N)$ time and $O(T^{n-1} N)$ space.



HMMs: what else?

- Using Viterbi, we can find the best tags for a sentence (**decoding**), and get $P(\mathbf{W}, \mathbf{T})$.
- We might also want to
 - Compute the **likelihood** $P(\mathbf{W})$, i.e., the probability of a sentence regardless of its tags (a language model!)
 - **learn** the best set of parameters (transition & emission probs.) given only an *unannotated* corpus of sentences.

Computing the likelihood

- From probability theory, we know that

$$P(\mathbf{W}) = \sum_{\mathbf{T}} P(\mathbf{W}, \mathbf{T})$$

- There are an exponential number of \mathbf{T} s.
- Again, by computing and storing partial results, we can solve efficiently.
- *(Advanced slides show the algorithm for those who are interested!)*

Summary

- HMM: a generative model of sentences using hidden state sequence
- Greedy tagging: fast but suboptimal
- Dynamic programming algorithms to compute
 - Best tag sequence given words (**Viterbi algorithm**)
 - Likelihood (forward algorithm—*see advanced slides*)
 - Best parameters from unannotated corpus (forward-backward algorithm, an instance of EM—*see advanced slides*)

Discriminative Sequence Taggers

- The HMM is generative and count-based
- Other approaches to sequence tagging are **discriminative feature-based linear models**, including:
 - **Structured perceptron**: mashup of the perceptron and Viterbi!
 - Linear-chain conditional random field (**CRF**): extension of MaxEnt classification + Viterbi!
 - *A separate set of slides introduces these*

Advanced Topics

(the following slides are just for people who are interested)

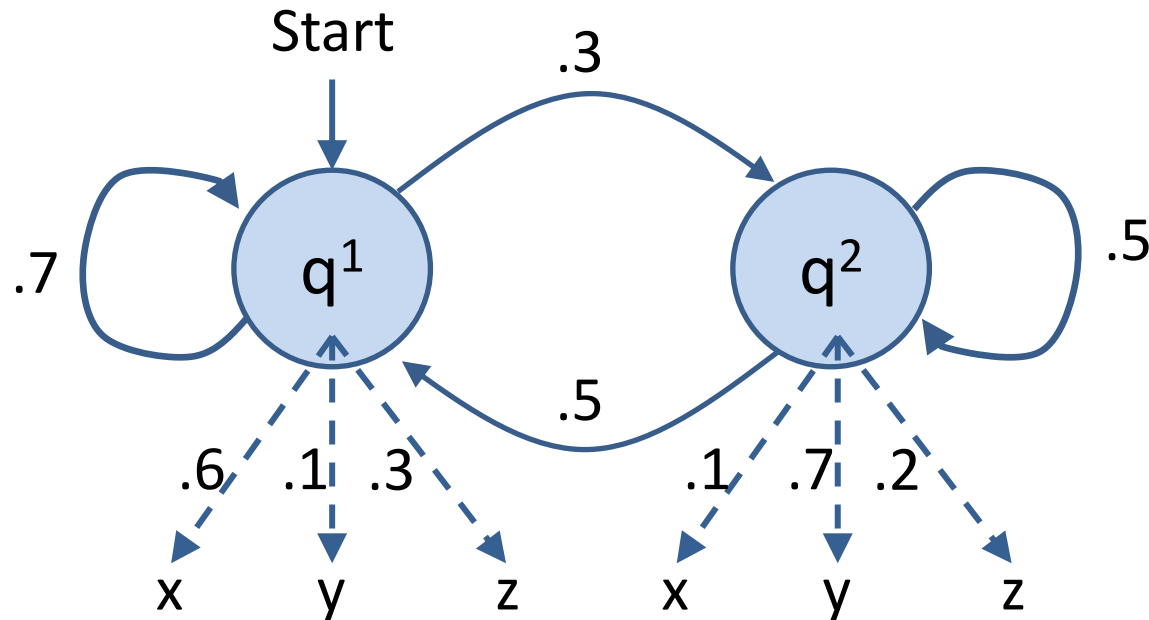
Notation

- Sequence of observations over time o_1, o_2, \dots, o_N
 - here, words in sentence
- Vocabulary size V of possible observations
- Set of possible states q^1, q^2, \dots, q^T (see note next slide)
 - here, tags
- A , an $T \times T$ matrix of transition probabilities
 - a_{ij} : the prob of transitioning from state i to j .
- B , an $T \times V$ matrix of output probabilities
 - $b_i(o_t)$: the prob of emitting o_t from state i .

Note on notation

- J&M use q_1, q_2, \dots, q_N for set of states, but *also* use q_1, q_2, \dots, q_N for state sequence over time.
 - So, just seeing q_1 is ambiguous (though usually disambiguated from context).
 - I'll instead use q^i for state names, and q_n for state at time n .
 - So we could have $q_n = q^i$, meaning: the state we're in at time n is q^i .

HMM example w/ new notation



- States $\{q^1, q^2\}$ (or $\{\langle s \rangle, q^1, q^2\}$): think *NN*, *VB*
- Output symbols $\{x, y, z\}$: think *chair*, *dog*, *help*

HMM example w/ new notation

- A possible sequence of outputs for this HMM:

$z\ y\ y\ x\ y\ z\ x\ z\ z$

- A possible sequence of states for this HMM:

$q^1\ q^2\ q^2\ q^1\ q^1\ q^2\ q^1\ q^1\ q^1$

- For these examples, $N = 9$, $q_3 = q^2$ and $o_3 = y$

Transition and Output Probabilities

- Transition matrix **A**:

$$a_{ij} = P(q^j | q^i)$$

$$\text{Ex: } P(q_n = q^2 | q_{n-1} = q^1) = .3$$

	q^1	q^2
$\langle s \rangle$	1	0
q^1	.7	.3
q^2	.5	.5

- Output matrix **B**:

$$b_i(o) = P(o | q^i)$$

$$\text{Ex: } P(o_n = y | q_n = q^1) = .1$$

	x	y	z
q^1	.6	.1	.3
q^2	.1	.7	.2

Forward algorithm

- Use a table with cells $\alpha(j,t)$: the probability of being in state j after seeing $o_1 \dots o_t$ (**forward probability**).

$$\alpha(j, t) = P(o_1, o_2, \dots, o_t, q_t = j | \lambda)$$

- Fill in columns from left to right, with

$$\alpha(j, t) = \sum_{i=1}^N \alpha(i, t-1) \cdot a_{ij} \cdot b_j(o_t)$$

- Same as Viterbi, but sum instead of max (and no backtrace).

Note: because there's a sum, we can't use the trick that replaces probs with costs. For implementation info, see <http://digital.cs.usu.edu/~cyan/CS7960/hmm-tutorial.pdf> and <http://stackoverflow.com/questions/13391625/underflow-in-forward-algorithm-for-hmms>.

Example

- Suppose $0=xzy$. Our initially empty table:

	$o_1=x$	$o_2=z$	$o_3=y$
q^1			
q^2			

Filling the first column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6		
q^2	0		

$$\alpha(1,1) = a_{\langle s \rangle_1} \cdot b_1(x) = (1)(.6)$$

$$\alpha(2,1) = a_{\langle s \rangle_2} \cdot b_2(x) = (0)(.1)$$

Starting the second column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	
q^2	0		

$$\begin{aligned}\alpha(1,2) &= \sum_{i=1}^N \alpha(i,1) \cdot a_{i1} \cdot b_1(z) \\ &= \alpha(1,1) \cdot a_{11} \cdot b_1(z) + \alpha(2,1) \cdot a_{21} \cdot b_1(z) \\ &= (.6)(.7)(.3) + (0)(.5)(.3) \\ &= .126\end{aligned}$$

Finishing the second column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	
q^2	0	.036	

$$\begin{aligned}\alpha(2,2) &= \sum_{i=1}^N \alpha(i,1) \cdot a_{i2} \cdot b_2(z) \\ &= \alpha(1,1) \cdot a_{12} \cdot b_2(z) + \alpha(2,1) \cdot a_{22} \cdot b_2(z) \\ &= (.6)(.3)(.2) + (0)(.5)(.2) \\ &= .036\end{aligned}$$

Third column and finish

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	.01062
q^2	0	.036	.03906

- Add up all probabilities in last column to get the probability of the entire sequence:

$$P(O|\lambda) = \sum_{i=1}^N \alpha(i, T)$$

Learning

- Given *only* the output sequence, learn the best set of parameters $\lambda = (A, B)$.
- Assume 'best' = maximum-likelihood.
- Other definitions are possible, won't discuss here.

Unsupervised learning

- Training an HMM from an annotated corpus is simple.
 - **Supervised** learning: we have examples labelled with the right 'answers' (here, tags): no hidden variables in training.
- Training from unannotated corpus is trickier.
 - **Unsupervised** learning: we have no examples labelled with the right 'answers': all we see are outputs, state sequence is hidden.

Circularity

- If we know the state sequence, we can find the best λ .
 - E.g., use MLE: $P(q^j|q^i) = \frac{C(q^i \rightarrow q^j)}{C(q^i)}$
- If we know λ , we can find the best state sequence.
 - use Viterbi
- But we don't know either!

Expectation-maximization (EM)

As in spelling correction, we can use EM to bootstrap, iteratively updating the parameters and hidden variables.

- Initialize parameters $\lambda^{(0)}$
- At each iteration k ,
 - E-step: Compute **expected counts** using $\lambda^{(k-1)}$
 - M-step: Set $\lambda^{(k)}$ using MLE on the expected counts
- Repeat until λ doesn't change (or other stopping criterion).

Expected counts??

Counting transitions from $q^i \rightarrow q^j$:

- Real counts:
 - count 1 each time we see $q^i \rightarrow q^j$ in true tag sequence.
- Expected counts:
 - With current λ , compute probs of all possible tag sequences.
 - If sequence Q has probability p , count p for each $q^i \rightarrow q^j$ in Q .
 - Add up these fractional counts across all possible sequences.

Example

- Notionally, we compute expected counts as follows:

Possible sequence				Probability of sequence
$Q_1 =$	q^1	q^1	q^1	p_1
$Q_2 =$	q^1	q^2	q^1	p_2
$Q_3 =$	q^1	q^1	q^2	p_3
$Q_4 =$	q^1	q^2	q^2	p_4
Observs:	x	z	y	

Example

- Notionally, we compute expected counts as follows:

Possible sequence				Probability of sequence
$Q_1 =$	q^1	q^1	q^1	p_1
$Q_2 =$	q^1	q^2	q^1	p_2
$Q_3 =$	q^1	q^1	q^2	p_3
$Q_4 =$	q^1	q^2	q^2	p_4
Observs:	x	z	y	

$$\hat{C}(q^1 \rightarrow q^1) = 2p_1 + p_3$$

Forward-Backward algorithm

- As usual, avoid enumerating all possible sequences.
- **Forward-Backward** (Baum-Welch) algorithm computes expected counts using forward probabilities and **backward probabilities**:

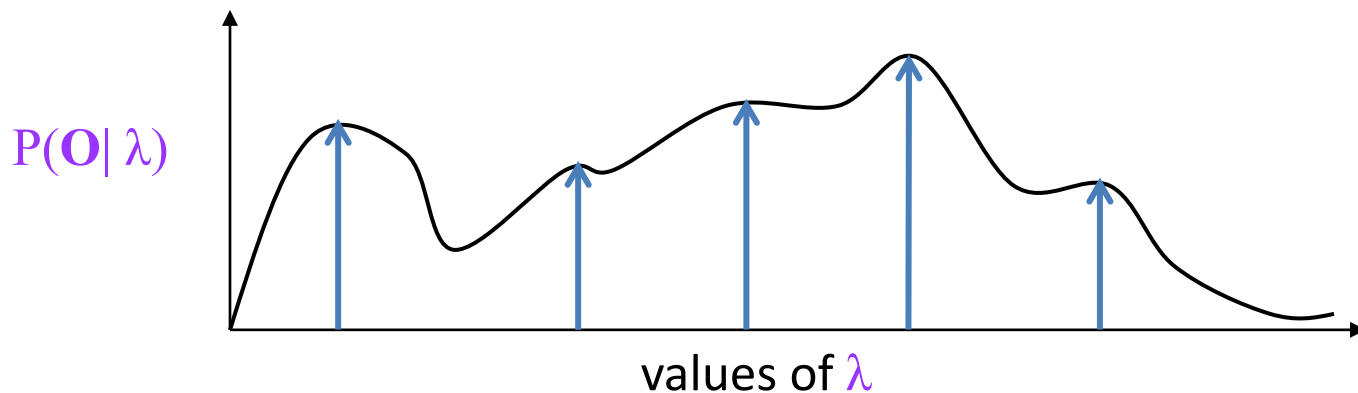
$$\beta(j, t) = P(q_t = j, o_{t+1}, o_{t+2}, \dots, o_T | \lambda)$$

– Details, see J&M 6.5

- EM idea is much more general: can use for many latent variable models.

Guarantees

- EM is guaranteed to find a **local** maximum of the likelihood.



- Not guaranteed to find **global** maximum.
- Practical issues: initialization, random restarts, early stopping. Fact is, it doesn't work well for learning POS taggers!