Lecture 12: Algorithms for HMMs

Nathan Schneider

(some slides from Sharon Goldwater)

ENLP | 17 October 2016

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(VBD|NN) and P(</s>|VBD) are high.

Recap: HMM

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

Recap: HMM

- Given a sentence $\mathbf{W}=\mathrm{w}_1\ldots\mathrm{w}_\mathrm{n}$ with tags $\mathbf{T}=\mathrm{t}_1\ldots\mathrm{t}_\mathrm{n}$, compute $\mathrm{P}(\mathbf{W},\mathbf{T})$ 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 $\mathrm{argmax}_{\mathbf{T}}\, P(\mathbf{T}|\mathbf{W})$without enumerating all possible tag sequences $\mathbf{T}$
  - Use a greedy approximation, or
  - Use Viterbi algorithm to store partial computations.

Greedy Tagging

Words:	<s>	one	dog	bit	</s>
Possible tags: (ordered by frequency for each word)	<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:

Possible tags:
(ordered by frequency for each word)

<s>	one	dog	bit	</s>
<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

- 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: high-level picture

- Want to find $\text{argmax}_{\mathbf{T}}\, P(\mathbf{T}|\mathbf{W})$
- Intuition: the best path of length i ending in state t must include the best path of length $\mathrm{i\text{-}1}$ to the previous state. So,
  - Find the best path of length $\mathrm{i\text{-}1}$ 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
  - $\mathrm{T} \times \mathrm{N}$ table, where $v(t, i)$ is the probability* of the best state sequence for $\mathrm{w}_1 \ldots \mathrm{w}_\mathrm{i}$ that ends in state t.

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

Viterbi algorithm

- Use a **chart** to store partial results as we go
  - $\mathrm{T} \times \mathrm{N}$ table, where $v(t,i)$ is the probability* of the best state sequence for $\mathrm{w}_1 \ldots \mathrm{w}_\mathrm{i}$ 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 $P(w_1 \ldots w_i, t_1 \ldots t_{i-1}, t_i = t | \lambda)$

Transition and Output Probabilities

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

	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	.1
Det	.9	.01	.01	.01	.07	0
Prep	.4	.05	.4	.1	.05	0
Adv	.1	.5	.1	.1	.1	.1

Emission matrix: $P(w_i \mid t_i)$:

	a	cat	doctor	in	is	the	very
Noun	0	.5	.4	0	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:

$\boldsymbol{v}$	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
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	</s>
Noun	0				
Verb	0				
Det	.21				
Prep	0				
Adv	0				

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

…

$v(\text{Det}, \text{the}) = P(\text{Det}|\text{<s>})\,P(\text{the}|\text{Det})$=.3(.7)

The second column

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

v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
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(.12), 0, 0 } = .0252

v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252			
Verb	0				
Det	.21				
Prep	0				
Adv	0				

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

The second column

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

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

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

v	$\text{w}_1\text{=the}$	$\text{w}_2\text{=doctor}$	$\text{w}_3\text{=is}$	$\text{w}_4\text{=in}$	</s>
Noun	0	.0252			
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	$\text{w}_1\text{=the}$	$\text{w}_2\text{=doctor}$	$\text{w}_3\text{=is}$	$\text{w}_4\text{=in}$	</s>
Noun	0	.0252			
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 { .0252(.02), .00021(.03), 0, 0, 0 } = .000504

v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252	.000504		
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 { .0252(.36), .00021(.045), 0, 0, 0 } = .009072

v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252	.000504		
Verb	0	.00021	.009072		
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 { .000504(.3), .009072(.2), 0, 0, 0 } = .001814

v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252	.000504	0	
Verb	0	.00021	.009072	0	
Det	.21	0	0	0	
Prep	0	0	0	.001814	
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(.03), .009072(.02), 0, 0, 0 } = .0001814

v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252	.000504	0	
Verb	0	.00021	.009072	0	
Det	.21	0	0	0	
Prep	0	0	0	.001814	
Adv	0	0	0	.0001814	

$P(\text{Prep}|\text{Noun})\ P(\text{in}|\text{Prep})$=.3(.1)

$P(\text{Prep}|\text{Verb})\ P(\text{in}|\text{Prep})$=.2(.1)

End of sentence

$v(</s>)$

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

= max { 0, 0, 0, .001814(0), .0001814(.1) } = .00001814

v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252	.000504	0	
Verb	0	.00021	.009072	0	
Det	.21	0	0	0	.00001814
Prep	0	0	0	.001814	
Adv	0	0	0	.0001814	

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

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

Completed Viterbi Chart

$\boldsymbol{v}$	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252	.000504	0	
Verb	0	.00021	.009072	0	
Det	.21	0	0	0	.00001814
Prep	0	0	0	.001814	
Adv	0	0	0	.0001814	

Following the Backtraces

v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252	.000504	0	
Verb	0	.00021	.009072	0	
Det	.21	0	0	0	.00001814
Prep	0	0	0	.001814	
Adv	0	0	0	.0001814	

Following the Backtraces

v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252	.000504	0	.00001814
Verb	0	.00021	.009072	0	
Det	.21	0	0	0	
Prep	0	0	0	.001814	
Adv	0	0	0	.0001814	

Following the Backtraces

v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252	.000504	0	
Verb	0	.00021	.009072	0	
Det	.21	0	0	0	.000018 14
Prep	0	0	0	.001814	
Adv	0	0	0	.0001814	

Following the Backtraces

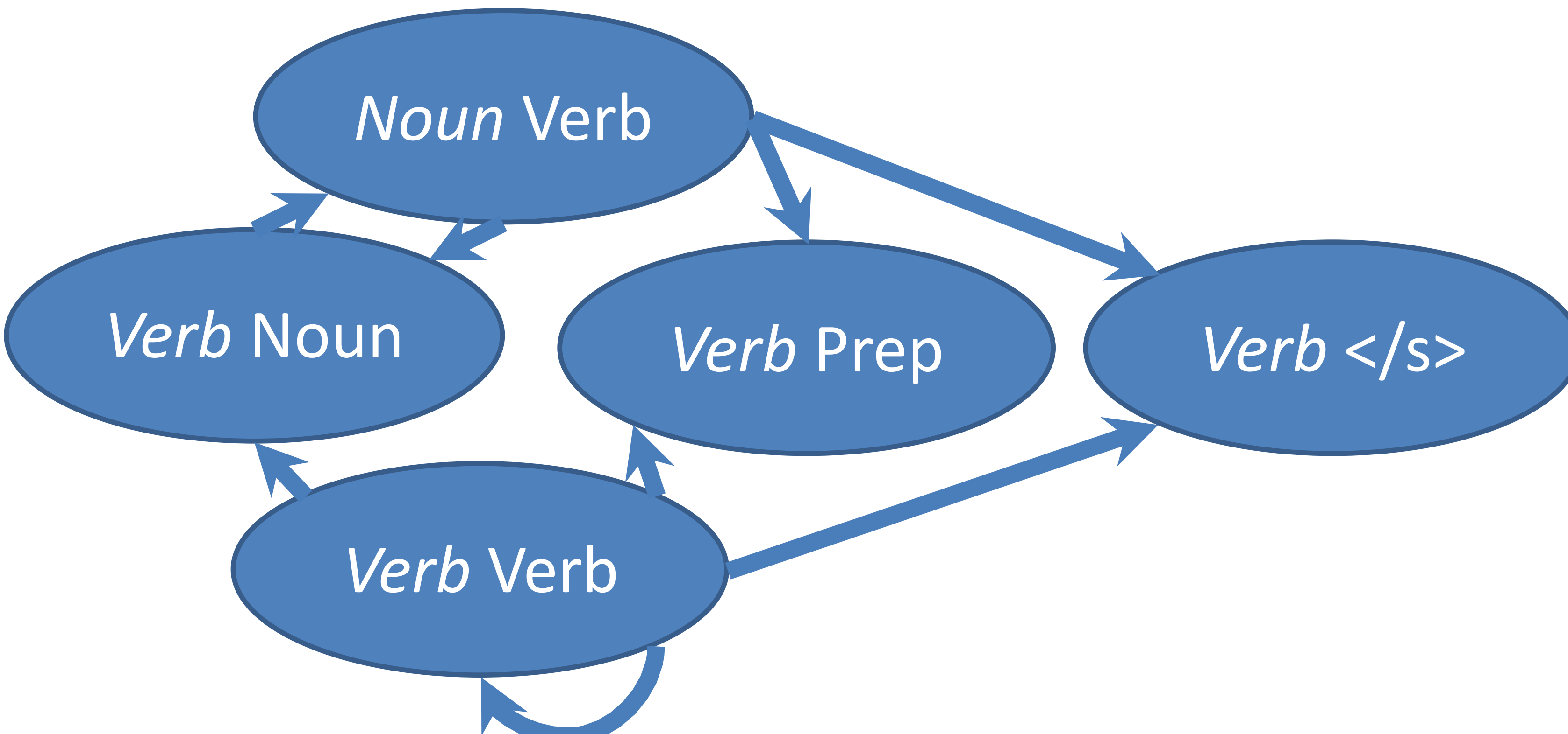
v	w_1=the	w_2=doctor	w_3=is	w_4=in	</s>
Noun	0	.0252	.000504	0	
Verb	0	.00021	.009072	0	
Det	.21	0	0	0	.00001814
Prep	0	0	0	.001814	
Adv	0	0	0	.0001814	
	Det	Noun	Verb	Prep	

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 $O(T^nN)$ 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 Ts.
- 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*)