

Sequence Tagging: Hidden Markov Models

COMP90042

Natural Language Processing

Lecture 6

Semester 1 2022 Week 3
Jey Han Lau



THE UNIVERSITY OF

MELBOURNE

POS Tagging Recap

- Janet will back the bill
- Janet/**NNP** will/**MB** back/**VP** the/**DT** bill/**NN**
- Local classifier: prone to **error propagation**
- What about treating the full sequence as a “class”?
 - ✖ ▶ Output: “NNP_MB_VP_DT_NN”
- Problems:
 - ▶ Exponentially many combinations: $|Tags|^M$, for length M
 - ▶ How to tag sequences of different lengths?

How human thinks A Better Approach

- Tagging is a sentence-level task but as humans we **decompose** it into **small word-level tasks**.
 - Janet/**NNP** will/**MB** back/**VP** the/**DT** bill/**NN**

Solution:

- Define a model that decomposes process into individual word level steps
- But that **takes into account** the **whole sequence** when learning and predicting (no error propagation)
global context
- This is the idea of **sequence labelling**, and more general, **structured prediction**.

A Probabilistic Model

- Goal: obtain best tag sequence $\hat{\mathbf{t}}$ from sentence \mathbf{w}
 - $\hat{\mathbf{t}} = \operatorname{argmax}_t P(\mathbf{t} \mid \mathbf{w})$
Highest prob.
 - $\hat{\mathbf{t}} = \operatorname{argmax}_t \frac{P(\mathbf{w} \mid \mathbf{t}) P(\mathbf{t})}{P(\mathbf{w})} = \operatorname{argmax}_t P(\mathbf{w} \mid \mathbf{t}) P(\mathbf{t})$ [Bayes]
How likely this tag sequence
- Let's decompose:
 - $P(\mathbf{w} \mid \mathbf{t}) = \prod_{i=1}^n P(w_i \mid t_i)$ [Prob. of a word depends only on the tag]
 - ↑ doesn't condition on anything else, besides its tag
(only care the current tag)
 - Strong assumption
 - $P(\mathbf{t}) = \prod_{i=1}^n P(t_i \mid t_{i-1})$ [Prob. of a tag depends only on the previous tag]
 - ↑
- This is a Hidden Markov Model (HMM)

Two Assumptions of HMM

$$\hat{t} = \operatorname{argmax}_t P(\mathbf{w} | t) P(t)$$

$$P(\mathbf{w} | t) = \prod_{i=1}^n P(w_i | t_i)$$

$$P(t) = \prod_{i=1}^n P(t_i | t_{i-1})$$

- Output independence
 - An observed event (word) depends only on the hidden state (tag)

hot affected
by other observed words

of the current tag
- Markov assumption
 - The current state (tag) depends only on previous state

HMMs - Training

↗ two sets of matrixes

- Parameters are the individual probabilities
 - ① • $P(w_i | t_i)$ = **emission (O) probabilities** how likely a given tag emits the w_i
 - ② • $P(t_i | t_{i-1})$ = **transition (A) probabilities** how likely a tag is going to transition into another tag
- Training uses Maximum Likelihood Estimation (MLE)
 - This is done by simply counting word frequencies according to their tags (just like N-gram LMs!)
 - $P(\text{like} | \text{VB}) = \frac{\text{count(VB, like)}}{\text{count(VB)}}$ Given a tagged corpus What's the likelihood of "VB" emitting the word "Like"
 - $P(\text{NN} | \text{DT}) = \frac{\text{count(DT, NN)}}{\text{count(DT)}}$

HMMs - Training

- What about the first tag?
 - Assume we have a symbol “<s>” that represents the start of your sentence

$$P(NN | < s >) = \frac{count(< s >, NN)}{count(< s >)}$$

- What about **unseen** (word, tag) and (tag, previous_tag) combinations?
 - **Smoothing techniques**

Transition Matrix

(A)

	NNP	MD	VB	JJ	NN	RB	DT
<i>< s ></i>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

Emission (Observation) Matrix

Why prob. of 0 << prob. of A

Vocab >> PoS classes

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

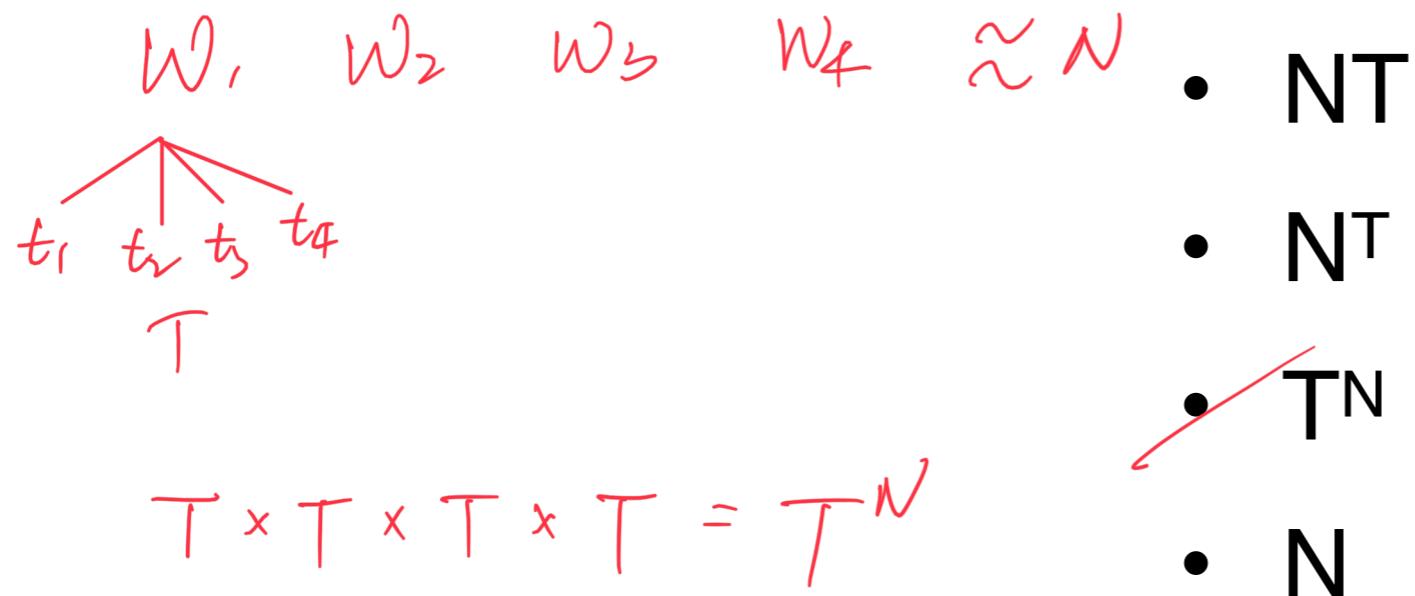
HMMs – Prediction (Decoding)

$$\begin{aligned}\hat{t} &= \operatorname{argmax}_t \sum_n P(w \mid t) P(t) \\ &= \operatorname{argmax}_t \prod_{i=1}^n P(w_i \mid t_i) P(t_i \mid t_{i-1})\end{aligned}$$

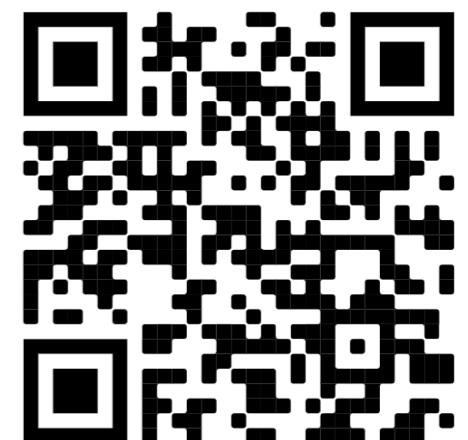
- Simple idea: for each word, take the tag that maximises $P(w_i \mid t_i) P(t_i \mid t_{i-1})$. Do it left-to-right, in *greedy* fashion.
- This is wrong! We are looking for argmax_t , not individual $\operatorname{argmax}_{t_i}$ terms.
 - This is a local classifier: error propagation
- Correct way: consider all possible tag combinations, evaluate them, take the max

What is the complexity of such a system that considers all possible tag combinations?

If sentence length = N and tag size = T



PollEv.com/jeyhanlau569



The Viterbi Algorithm

Consider all the possible previous combinations when you're

- Dynamic Programming to the rescue!
 - We can still proceed sequentially, as long as we are careful.

• POS tag: “can play”

- Best tag for “can” is easy: $\operatorname{argmax}_t P(\text{can} \mid t) P(t \mid \langle s \rangle)$
- Suppose best tag for “can” is NN. To get the tag for “play”, we can take $\operatorname{argmax}_t P(\text{play} \mid t) P(t \mid \text{NN})$ but this is wrong.
- Instead, we keep track of **scores for each tag** for “can” and check them with the different tags of “play”.

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP					
MD					
VB					
JJ					
NN					
RB					
DT					

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	$P(\text{Janet} \text{NNP}) * P(\text{NNP} <\text{s}>)$				
MD					
VB					
JJ	...				
NN	...				
RB	...				
DT	...				

$P(\text{Janet} \mid \text{NNP}) = ?; P(\text{NNP} \mid \langle s \rangle) = ?$

	NNP	MD	VB	JJ	NN	RB	DT
$\langle s \rangle$	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	0.000032 * 0.2767 = 8.8544e-06				
MD	P(Janet MD) * P(MD <s>)				
VB	...				
JJ	...				
NN	...				
RB	...				
DT	...				

$$P(\text{Janet} \mid \text{MD}) = ?; P(\text{MD} \mid \langle s \rangle) = ?$$

	NNP	MD	VB	JJ	NN	RB	DT
<i>< s ></i>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 ●				
MD	0				
VB	0				
JJ	0				
NN	0				
RB	0				
DT	0				

$P(\text{Janet} \mid \text{VB/JJ/etc}) = ?$

	NNP	MD	VB	JJ	NN	RB	DT
< s >	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(\text{VB}|MD)$ is 0.7968.

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	P(will NNP) * P(NNP <1s>) * $s(t_{Janet} Janet)$			
MD	0	$P(J MD) P(MD <1s>)$...	Consider all possible tags, and take the maximum	
VB	0	...		$\max(P(will NNP) * P(NNP NNP) * s(NNP Janet),$ $P(will NNP) * P(NNP MD) * s(MD Janet),$...	
JJ	0	...		$P(will NNP) * P(NNP DT) * s(DT Janet))$	
NN	0	...		$= P(will NNP) * P(NNP NNP) * 8.8544e-06$	
RB	0	...		0 0.3777	
DT	0	...			

$P(\text{will} \mid \text{NNP}) = ?; P(\text{NNP} \mid \text{NNP}) = ?$

	NNP	MD	VB	JJ	NN	RB	DT
<i>< s ></i>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0			
MD	0	...			
VB	0	...			
JJ	0	...			
NN	0	...			
RB	0	...			
DT	0	...			

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0			
MD	0	P(will MD) * P(MD NNP) * s(NNP Janet)			
VB	0	...			
JJ	0	...			
NN	0	...			
RB	0	...			
DT	0	...			

Diagram illustrating the Viterbi algorithm for the sequence "Janet will back the bill". The states are NNP, MD, VB, JJ, NN, RB, and DT. The words are Janet, will, back, the, and bill.

The table shows the forward probabilities (red arrows) and the states (black text) at each position. The final result is shown in a box:

$$\max(P(\text{will} | \text{MD}) * P(\text{MD} | \text{NNP}) * s(\text{NNP} | \text{Janet}),$$

$$P(\text{will} | \text{MD}) * P(\text{MD} | \text{MD}) * s(\text{MD} | \text{Janet}),$$

$$\dots$$

$$P(\text{will} | \text{MD}) * P(\text{MD} | \text{DT}) * s(\text{DT} | \text{Janet}))$$

$$= P(\text{will} | \text{MD}) * P(\text{MD} | \text{NNP}) * 8.8544e-06$$

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0			
MD	0	3.004e-08			
VB	0	...			
JJ	0	...	$\max(P(\text{will} \text{MD}) * P(\text{MD} \text{NNP}) * s(\text{NNP} \text{Janet}),$ $P(\text{will} \text{MD}) * P(\text{MD} \text{MD}) * s(\text{MD} \text{Janet}),$...		
NN	0	...	$P(\text{will} \text{MD}) * P(\text{MD} \text{DT}) * s(\text{DT} \text{Janet}))$		
RB	0	...	$= P(\text{will} \text{MD}) * P(\text{MD} \text{NNP}) * 8.8544e-06$		
DT	0	...			

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0			
MD	0	3.004e-8			
VB	0	2.231e-13			
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

Transition and Emission Matrix

	NNP	MD	VB	JJ	NN	RB	DT
< s >	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0			
MD	0	3.004e-8			
VB	0	2.231e-13			
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	$P(\text{back} \text{VB})^*$ $P(\text{VB} t_{\text{will}})^*$ $s(t_{\text{will}} \text{will})$		
JJ	0	0	$p(\text{back} \text{VB}) \cdot p(\text{VB} NNP) \cdot s(NNP \text{will})$		
NN	0	1.034e-10	$p(\text{back} \text{VB}) \cdot p(\text{VB} MD) \cdot s(MD \text{will})$ \dots		
RB	0	0			
DT	0	0			

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	MD: 1.6e-11 VB: 7.5e-19 NN: 9.7e-17		
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	MD: 1.6e-11 VB: 7.5e-19 NN: 9.7e-17		
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	1.6e-11		
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	1.6e-11		
JJ	0	0	5.1e-15		
NN	0	1.034e-10	5.4e-15		
RB	0	0	5.3e-11		
DT	0	0	0		

The diagram illustrates the Viterbi algorithm's state transitions. Red dashed arrows point from one state to the next, indicating the path of maximum probability. The states are NNP, MD, VB, JJ, NN, RB, and DT. The time steps are Janet, will, back, the, and bill. The path starts at NNP for Janet, moves to MD for will, then to VB for back, then to NN for the, and finally to RB for bill.

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	
MD	0	3.004e-8	0	0	
VB	0	2.231e-13	1.6e-11	0	
JJ	0	0	5.1e-15	0	
NN	0	1.034e-10	5.4e-15	0	
RB	0	0	5.3e-11	0	
DT	0	0	0	1.8e-12	

Done!

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-8	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	0	0
NN	0	1.034e-10	5.4e-15	0	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

Going Backwards To Get The Best Tag Sequence

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-8	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	0	0
NN	0	1.034e-10	5.4e-15	0	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

The diagram illustrates the process of backtracking to find the best tag sequence. Red dashed arrows point from the circled values (8.8544e-06, 3.004e-8, 1.6e-11, 2.0e-15, and 1.8e-12) back towards the first column, indicating the path of highest probability from right to left.

Going Backwards To Get The Best Tag Sequence

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-08	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	0	0
NN	0	1.034e-10	5.4e-15	0	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

Final Tagging Results

	Janet/NNP	will/MD	back/VB	the/DT	bill/NN
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-08	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	0	0
NN	0	1.034e-10	5.4e-15	0	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

What is the optimal POS tag sequence for:

they fish
 N 0.42 ✓ 0.1008 ✓
 ✓ 0.16 0.0504

	they	fish
N	0.7	0.3
V	0.4	0.6

Word Emission Probabilities

THEY FISH

$$0.6 \times 0.2 \times 0.42 = 0.0504 \checkmark$$

$$0.6 \times 0.3 \times 0.16 = 0.0288$$

	N	V
<S>	0.6	0.4
N	0.8	0.2
V	0.7	0.3

Transition Probabilities

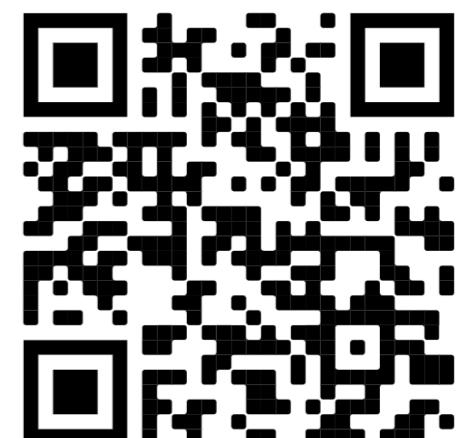
$$P(\text{they}|N)P(N|<\text{S}>) = 0.7 \times 0.6 = 0.42$$

$$P(\text{they}|V)P(V|<\text{S}>) = 0.4 \times 0.4 = 0.16$$

$$P(\text{fish}|N)P(N|N)S(N|\text{they}) = 0.3 \times 0.8 \times 0.42 = 0.1008 \checkmark$$

PollEv.com/jeyhanlau569

$$P(\text{fish}|N)P(N|V)S(V|\text{they}) = 0.3 \times 0.7 \times 0.16 = 0.0336$$



The Viterbi Algorithm

- Complexity: $O(T^2N)$, where T is the size of the tagset and N is the length of the sequence.
 - $T * N$ matrix, each cell performs T operations.
- Why does it work?
 - Because of the **independence assumptions** that decompose the problem.
 - Without these, we cannot apply DP.

Viterbi Pseudocode

```

alpha = np.zeros(M, T)
for t in range(T):
    alpha[1, t] = pi[t] * O[w[1], t]

{
    for i in range(2, M):
        for t_i in range(T): } M(N) x T x T  $\Rightarrow O(T^2N)$ 
            for t_last in range(T): # t_last means t_{i-1}
                s = alpha[i-1, t_last] * A[t_last, t_i] * O[w[i], t_i]
                if s > alpha[i, t_i]:
                    alpha[i, t_i] = s
                    back[i, t_i] = t_last

best = np.max(alpha[M-1,:])
return backtrace(best, back)

```

- Good practice: work with **log** probabilities to prevent underflow (multiplications become sums)
- Vectorisation (use matrix-vector operations)

HMMs In Practice

- We saw HMM taggers based on **bigrams** (first order HMM).
 - I.e. current tag depends only the immediate previous tag
- State-of-the-art use tag **trigrams** (second order HMM).
 - $P(t) = \prod_{i=1}^n P(t_i | t_{i-1}, t_{i-2})$ Viterbi now $O(T^3N)$
- Need to deal with sparsity: some tag trigram sequences might not be present in training data
 - Interpolation: $P(t_i | t_{i-1}, t_{i-2}) = \lambda_3 \hat{P}(t_i | t_{i-1}, t_{i-2}) + \lambda_2 \hat{P}(t_i | t_{i-1}) + \lambda_1 \hat{P}(t_i)$
 - $\lambda_1 + \lambda_2 + \lambda_3 = 1$
- With additional features, reach 96.5% accuracy on Penn Treebank (Brants, 2000)

Generative vs. Discriminative Taggers

- HMM is **generative**
 - ▶ $\hat{T} = \operatorname{argmax}_T P(T | W)$
 - ▶ $= \operatorname{argmax}_T P(W | T)P(T)$
 - ▶ $= \operatorname{argmax}_T \prod_i P(w_i | t_i)P(t_i | t_{i-1})$

look at the Emission Matrix
why? \Rightarrow generate first tag by Transition Matrix \rightarrow sample another word
- trained HMM can generate data (sentences)! conditioning on the tag
 $\overbrace{\text{by our model you've trained}}$ just sampled
- allows for unsupervised HMMs: learn model without any tagged data!

Assume X tags, assign tags \rightarrow words by the best prob.

Generative vs. Discriminative Taggers

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T \mid W) \\ &= \operatorname{argmax}_T \prod_i P(t_i \mid w_i, t_{i-1})\end{aligned}$$

- **Discriminative** models describe $P(T \mid W)$ directly
 - supports richer feature set, generally better accuracy when trained over large supervised datasets
 - $P(t_i \mid w_i, t_{i-1}, \underline{x_i}, \underline{y_i})$
 - E.g., Maximum Entropy Markov Model (MEMM), Conditional random field (CRF)
 - Most deep learning models of sequences are discriminative

HMMs in NLP

- HMMs are **highly effective** for part-of-speech tagging
 - ▶ trigram HMM gets 96.5% accuracy
 - ▶ related models are state of the art
 - ▶ MEMMs 97%
 - ▶ CRFs 97.6%
 - ▶ Deep CRF 97.9%
 - ▶ English Penn Treebank tagging accuracy [https://aclweb.org/aclwiki/index.php?title=POS_Tagging_\(State_of_the_art\)](https://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art))
- Apply out-of-the box to other sequence labelling tasks
 - ▶ named entity recognition (lecture 18), shallow parsing, ...
 - ▶ In other fields: DNA, protein sequences...

A Final Word

- HMMs are a simple, yet effective way to perform sequence labelling.
- Can still be competitive, and fast. Natural baseline for other sequence labelling models.
- Main drawback: not very flexible in terms of feature representation, compared to MEMMs and CRFs.

Readings

- JM3 Appendix A A.1-A.2, A.4
- See also E18 Chapter 7.3
- References:
 - Rabiner's HMM tutorial <http://tinyurl.com/2hqaf8>
 - Lafferty et al, Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001)