DEPARTMENT OF COMPUTING

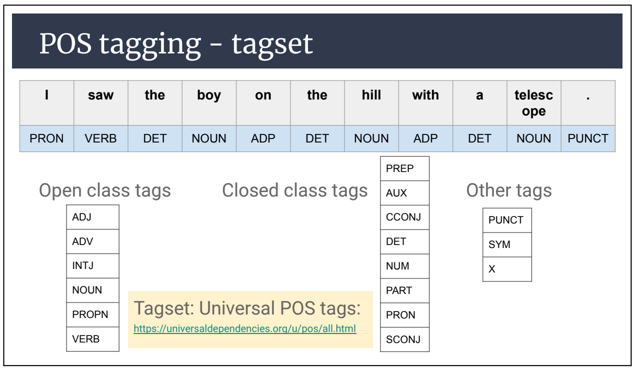
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

POS tagging

Author: Anton Zhitomirsky

Contents

1 Part Of Speech Tagging



- POS tries to find labels we're interested in (verb adjective, etc.)

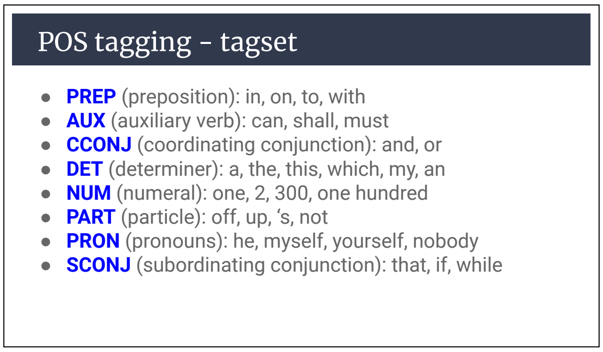
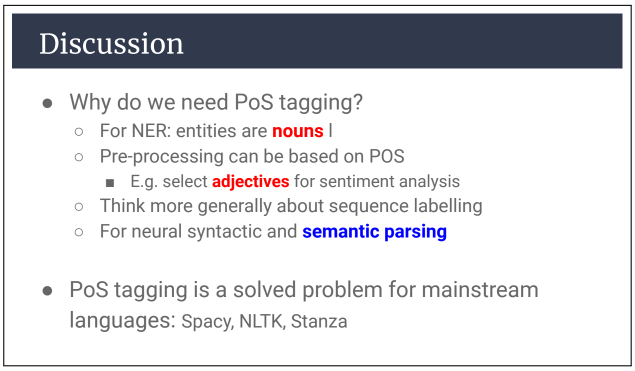


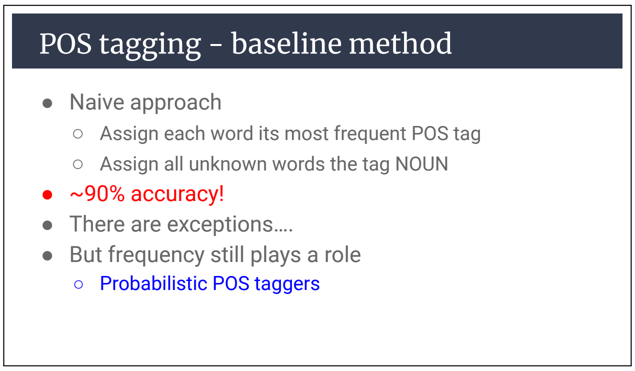
Figure 1: Some examples of attributes we may wish to tag

1.1 Why do we need POS tagging?



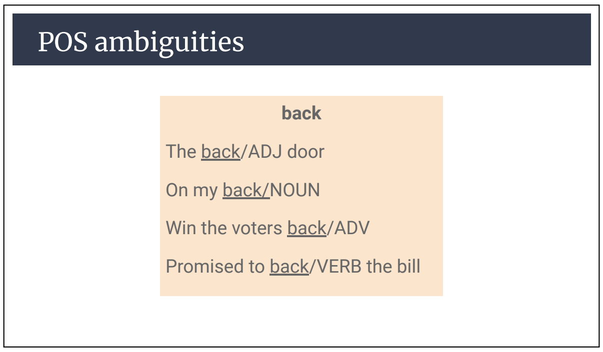
- it is a solved problem
- There are still tasks we may need to do at scale (e.g. a language filtering problem with many messages we may not be able to run a transformed based language model on all of them)

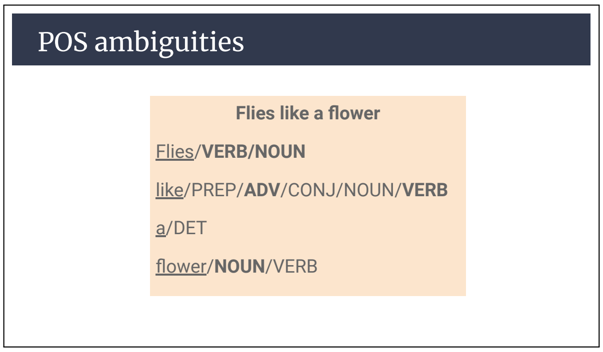
1.2 Baseline method



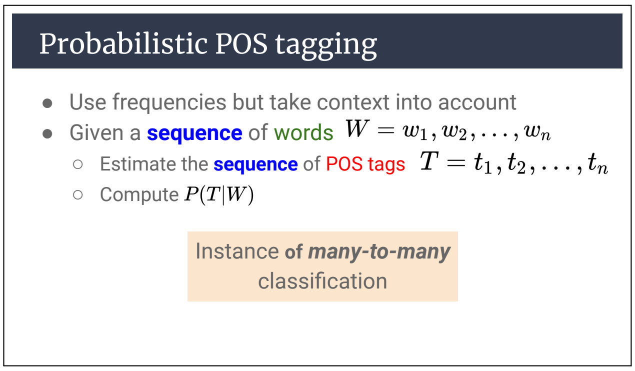
- NOUN is the most common tag, so we could just tag everything as a noun
- This is a baseline method, and we can do better than this

1.2.1 Ambiguities

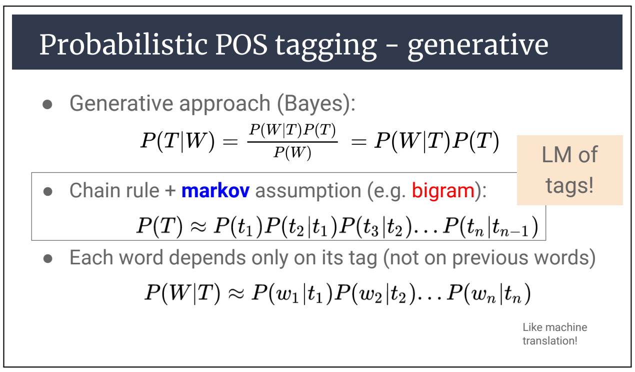




1.3 Probabilistic POS tagging



- Use the frequences but take the context into account.



- The probability of a word sequence given the word sequence is 1 becuase we're not drawing $P(W)$ from a word distribution.
- We take the bigram model as assumption
- note above w is a word and t is a tag

Probabilistic POS tagging - generative

- Putting both together:

$$P(T|W) \approx \boxed{P(t_1)P(t_2|t_1)\ldots P(t_n|t_{n-1})}\boxed{P(w_1|t_1)P(w_2|t_2)\ldots P(w_n|t_n)}$$
$$\approx P(t_1)P(w_1|t_1)P(t_2|t_1)P(w_2|t_2)\ldots P(t_n|t_{n-1})P(w_n|t_n)$$

where, as before $P(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})}$

and $P(w_i|t_i) = \frac{C(w_i,t_i)}{C(t_i)}$

- as a baseline, we would a basic statistical count.
- The first says, how many times i encounter a tak t_i after i observe a tag t_{i-1} given that the total number of times I have ever seen the tag t_{i-1}.
- similarly for the second.

1.3.1 Example

Probabilistic POS tagging - generative

- For example, given a training corpus:

John/PROPN is/VERB expected/VERB to/PART race/VERB

This/DET is/VERB the/DET race/NOUN I/PRON wanted/VERB

Bring/VERB this/DET to/PART the/DET race/NOUN

- Compute the necessary probabilities

- Here is the training corpus

Probabilistic POS tagging - generative

t_0

$P(t_i|t_{i-1})$ $P(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})}$

	PROPN	VERB	PART	NOUN	PRON	DET
<s> (3)	1/3	1/3	0	0	0	1/3
PROPN (1)	0	1/1	0	0	0	0
VERB (6)	0	1/6	1/6	0	0	2/6
PART (2)	0	1/2	0	0	0	1/2
NOUN (2)	0	0	0	0	1/2	0
PRON (1)	0	1/1	0	0	0	0
DET (4)	0	1/4	1/4	2/4	0	0

How to make the sum of each row **equal 1**?

- from the start symbol (we have 3 sentences above) we form a table.
- However, this doesn't add up to 1.

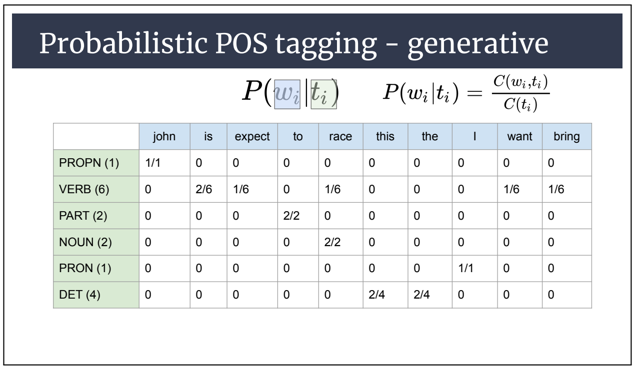
Probabilistic POS tagging - generative

t_0

$P(t_i|t_{i-1})$ $P(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})}$

	PROPN	VERB	PART	NOUN	PRON	DET	</s>
<s> (3)	1/3	1/3	0	0	0	1/3	0
PROPN (1)	0	1/1	0	0	0	0	0
VERB (6)	0	1/6	1/6	0	0	2/6	2/6
PART (2)	0	1/2	0	0	0	1/2	0
NOUN (2)	0	0	0	0	1/2	0	1/2
PRON (1)	0	1/1	0	0	0	0	0
DET (4)	0	1/4	1/4	2/4	0	0	0

- Therefore, we add a terminating tag, which then fills the rest of the probability in.
- In the verb case, we have observed 6 verbs, yet 2 out of the 6 verbs have been used to finish the sentence.



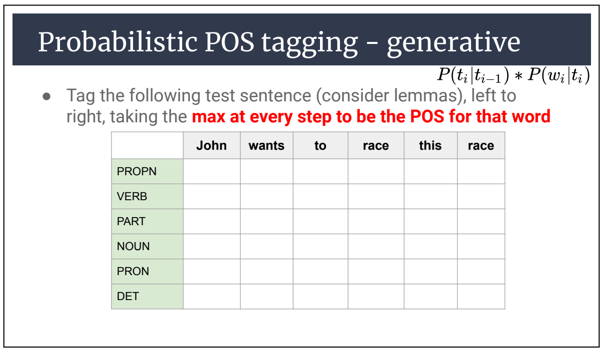
Probabilistic POS tagging - generative

$P(w_i|t_i)$ $P(w_i|t_i) = \frac{C(w_i,t_i)}{C(t_i)}$

	john	is	expect	to	race	this	the	I	want	bring
PROPN (1)	1/1	0	0	0	0	0	0	0	0	0
VERB (6)	0	2/6	1/6	0	1/6	0	0	0	1/6	1/6
PART (2)	0	0	0	2/2	0	0	0	0	0	0
NOUN (2)	0	0	0	0	2/2	0	0	0	0	0
PRON (1)	0	0	0	0	0	0	0	1/1	0	0
DET (4)	0	0	0	0	0	2/4	2/4	0	0	0

- you do the same given for the word given the tag case.
- this is a simple counting baseline

We then apply the formula to get the posterior (the tag given the word sequence), and at each column we apply the max. This runs at lightning speed and is very parallelizeable, and gives a good baseline.

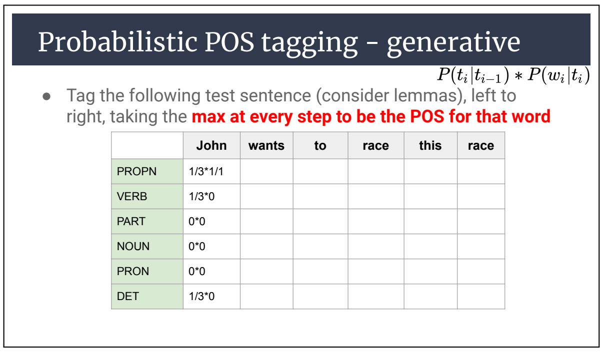


Probabilistic POS tagging - generative

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- Tag the following test sentence (consider lemmas), left to right, taking the **max at every step to be the POS for that word**

	John	wants	to	race	this	race
PROPN						
VERB						
PART						
NOUN						
PRON						
DET						

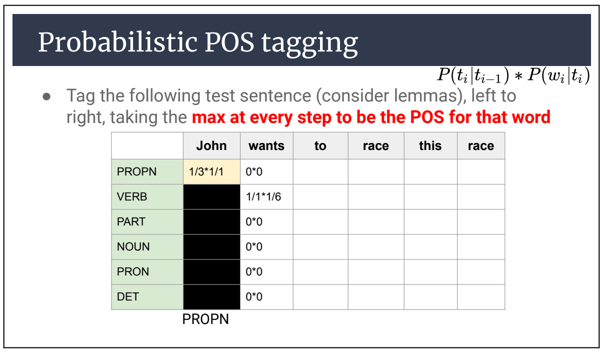


Probabilistic POS tagging - generative

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- Tag the following test sentence (consider lemmas), left to right, taking the **max at every step to be the POS for that word**

	John	wants	to	race	this	race
PROPN	1/3*1/1					
VERB	1/3*0					
PART	0*0					
NOUN	0*0					
PRON	0*0					
DET	1/3*0					



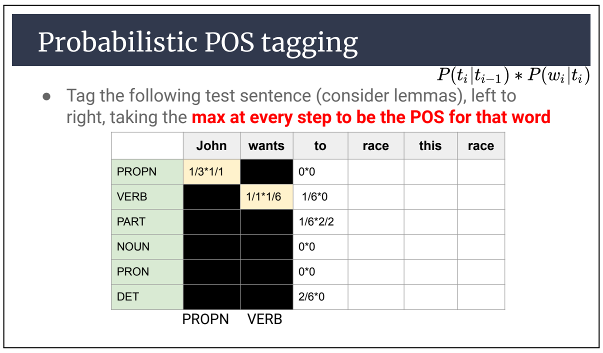
Probabilistic POS tagging

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- Tag the following test sentence (consider lemmas), left to right, taking the **max at every step to be the POS for that word**

	John	wants	to	race	this	race
PROPN	1/3*1/1	0*0				
VERB		1/1*1/6				
PART		0*0				
NOUN		0*0				
PRON		0*0				
DET		0*0				

PROPN



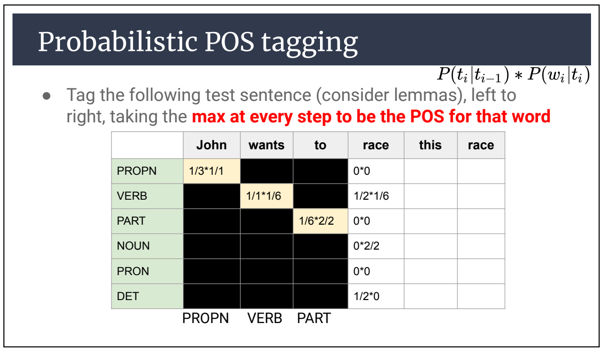
Probabilistic POS tagging

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- Tag the following test sentence (consider lemmas), left to right, taking the **max at every step to be the POS for that word**

	John	wants	to	race	this	race
PROPN	1/3*1/1		0*0			
VERB		1/1*1/6	1/6*0			
PART			1/6*2/2			
NOUN			0*0			
PRON			0*0			
DET			2/6*0			

PROPN VERB



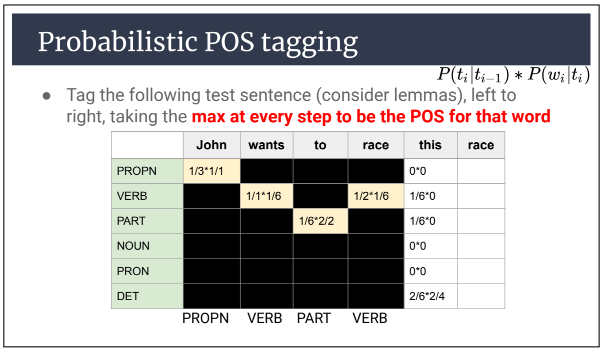
Probabilistic POS tagging

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- Tag the following test sentence (consider lemmas), left to right, taking the **max at every step to be the POS for that word**

	John	wants	to	race	this	race
PROPN	1/3*1/1			0*0		
VERB		1/1*1/6		1/2*1/6		
PART			1/6*2/2	0*0		
NOUN				0*2/2		
PRON				0*0		
DET				1/2*0		

PROPN VERB PART



Probabilistic POS tagging

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- Tag the following test sentence (consider lemmas), left to right, taking the **max at every step to be the POS for that word**

	John	wants	to	race	this	race
PROPN	1/3*1/1				0*0	
VERB		1/1*1/6		1/2*1/6	1/6*0	
PART			1/6*2/2		1/6*0	
NOUN					0*0	
PRON					0*0	
DET					2/6*2/4	

PROPN VERB PART VERB

Probabilistic POS tagging

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- Tag the following test sentence (consider lemmas), left to right, taking the **max at every step to be the POS for that word**

	John	wants	to	race	this	race
PROPN	1/3*1/1					0*0
VERB		1/1*1/6		1/2*1/6		1/4*1/6
PART			1/6*2/2			1/4*0
NOUN						2/4*2/2
PRON						0*0
DET					2/6*2/4	0*0
	PROPN	VERB	PART	VERB	DET	

Probabilistic POS tagging

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- Tag the following test sentence (consider lemmas), left to right, taking the **max at every step to be the POS for that word**

	John	wants	to	race	this	race
PROPN	1/3*1/1					
VERB		1/1*1/6		1/2*1/6		
PART			1/6*2/2			
NOUN						2/4*2/2
PRON						
DET					2/6*2/4	
	PROPN	VERB	PART	VERB	DET	NOUN

1.3.2 Example Problem

Probabilistic POS tagging

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- "John wants to race this, race fast"; P(fast|ADV) = 1; P(ADV|VERB) = 1/6

	PROPN	VERB	PART	VERB	DET	NOUN	?	
	John	wants	to	race	this	race	fast	
PROPN							*0	
VERB							*0	
PART							*0	
NOUN						2/4*2/2	*0	
PRON							*0	
DET							*0	
ADV							0*1/1	P(ADV|NOUN) = 0

- If we extend this sentence with 'fast' we don't have a transition in the training sample.
- We have tagged this as a noun, even though it should be adverb. This is becuase it doesn't have a transition in the training data even though the only time we observed fast before, it was an adverb.
- However, we have never observed an adverb after a noun so we never transitioned from this point.

> ☞ Therfore, we aren't really considering the entire sequence, we're only considering local decisions by the greedy approach.

Probabilistic POS tagging

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- "John wants to race this, race fast"; P(fast|ADV) = 1; P(ADV|VERB) = 1/6

	PROPN	VERB	PART	VERB	DET	NOUN/VERB	
	John	wants	to	race	this	race	fast
PROPN						0*0	
VERB						1/4*1/6	
PART						1/4*0	
NOUN						2/4*2/2	
PRON						0*0	
DET						0*0	
ADV						0*0	

Probabilistic POS tagging

$P(t_i|t_{i-1}) * P(w_i|t_i)$

- "John wants to race this, race fast"; P(fast|ADV) = 1; P(ADV|VERB) = 1/6

	PROPN	VERB	PART	VERB	DET	VERB	ADV	
	John	wants	to	race	this	race	fast	
PROPN						0*0	*0	
VERB						1/4*1/6	*0	
PART						1/4*0	*0	
NOUN						2/4*2/2	*0	
PRON						0*0	*0	
DET						0*0	*0	
ADV						0*0	1/6*1/1	P(ADV|VERB) = 1/6

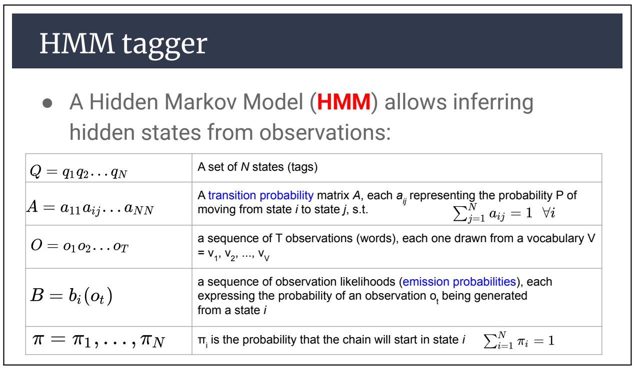
HMM tagger

- Issues with tagging left to right based on local max?
  - We are ignoring more promising paths overall by sticking to one decision at a step
- Instead, compute best tag **sequence** $\hat{T}$, one that maximizes $P(T|W)$

$$\hat{T} = argmax_T P(T|W)$$

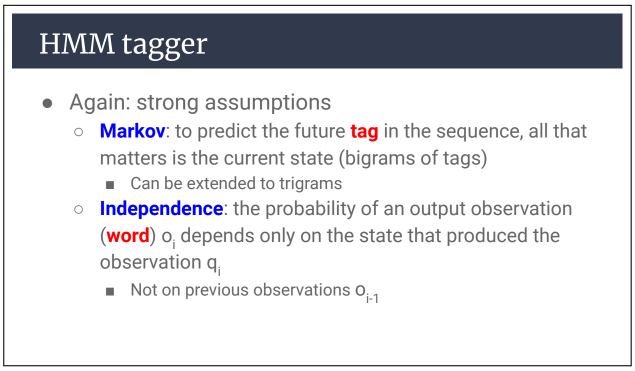
- We're ignoring more promising paths.
- We are therefore interested in maximising the posterior.

1.4 Hidden Markov Model Tagger

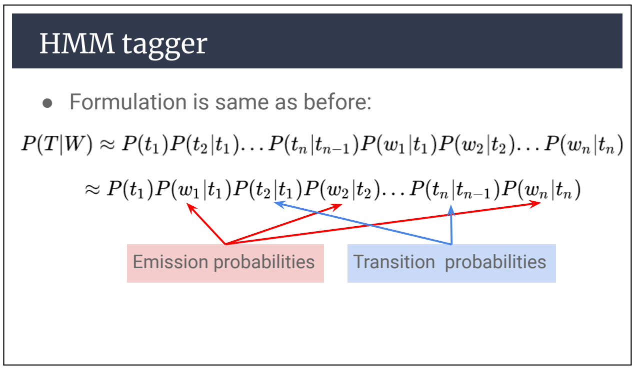


1. **Markov Chains**: Model probabilities of seqeunces of random variables (states)
2. **Hidden Markov Chains**: states are not given, but hidden. This means that words are observed, but the POS tags are hidden.

This allows us to inferr hidden states from observations.



- we're not taking into account the FULL history. We can e.g. exapnd into trigrams.



- We use the same formula as before, only this time w is representing the emission probability (i.e. the probability of observing a word given a tag) and a transition probability (i.e. the probability of observing a tag given a previous tag).

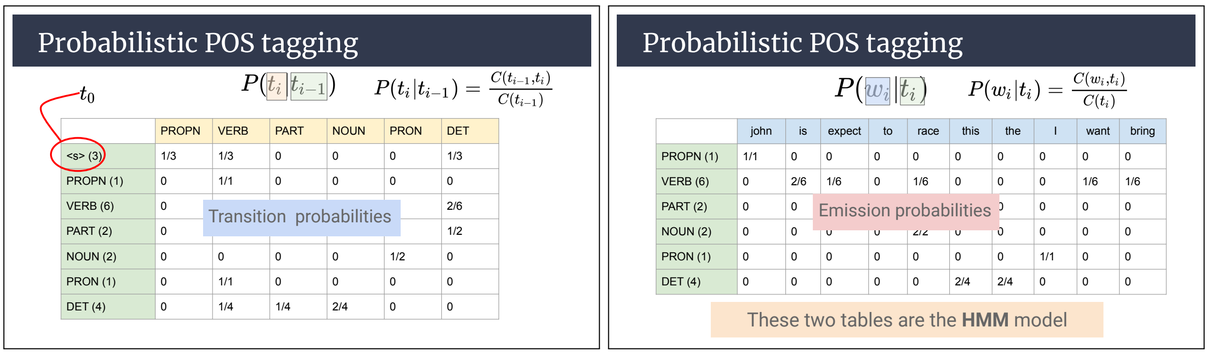
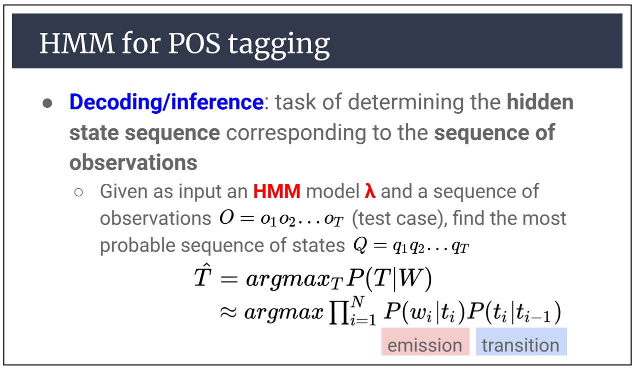


Figure 2: Tables are unchanged



- Here, the hidden state is the tag, and the seqeunce of observations are words.
- As before, we're not doing this greedily, we want to maximise the posterior, but instead we are now working with emission and transition probabilities.

1.4.1 Viterbi algorithm

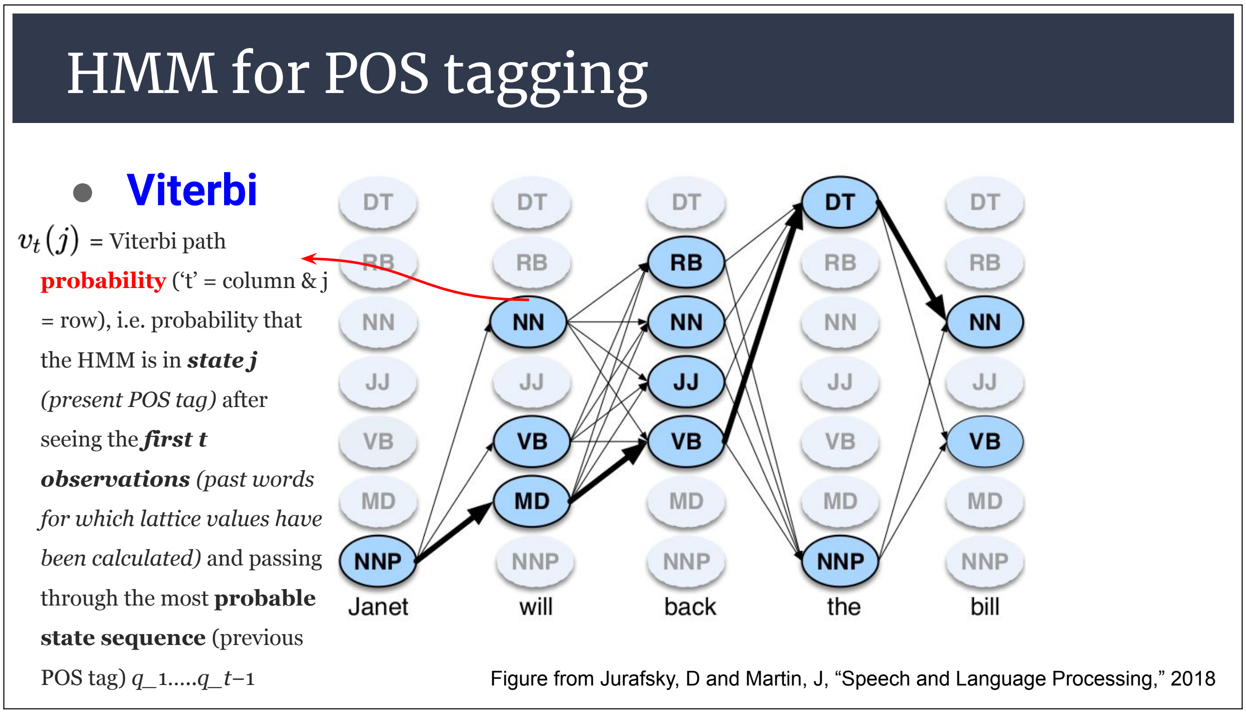


Figure 3: For a particular state, what is the porbability that the model is in state j, i.e. has that tag, after seeing the first p observations. Here, t is the time. and passing through the probable sequence.

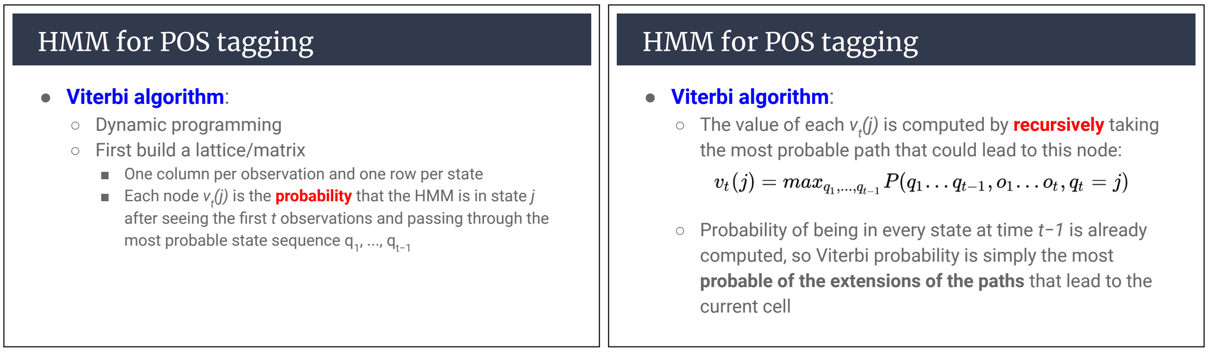


Figure 4: Dynamic programming algorithm: we can reuse subprobelms.

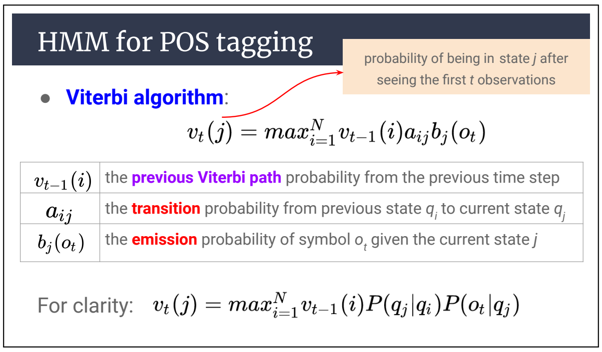
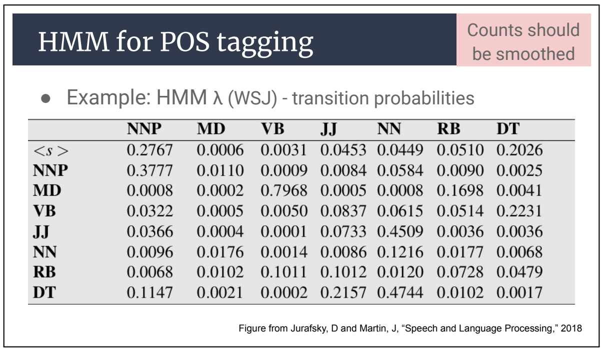


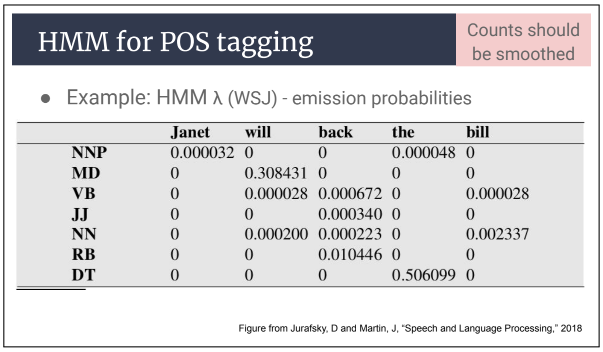
Figure 5: Recursive definition

1.4.2 Example



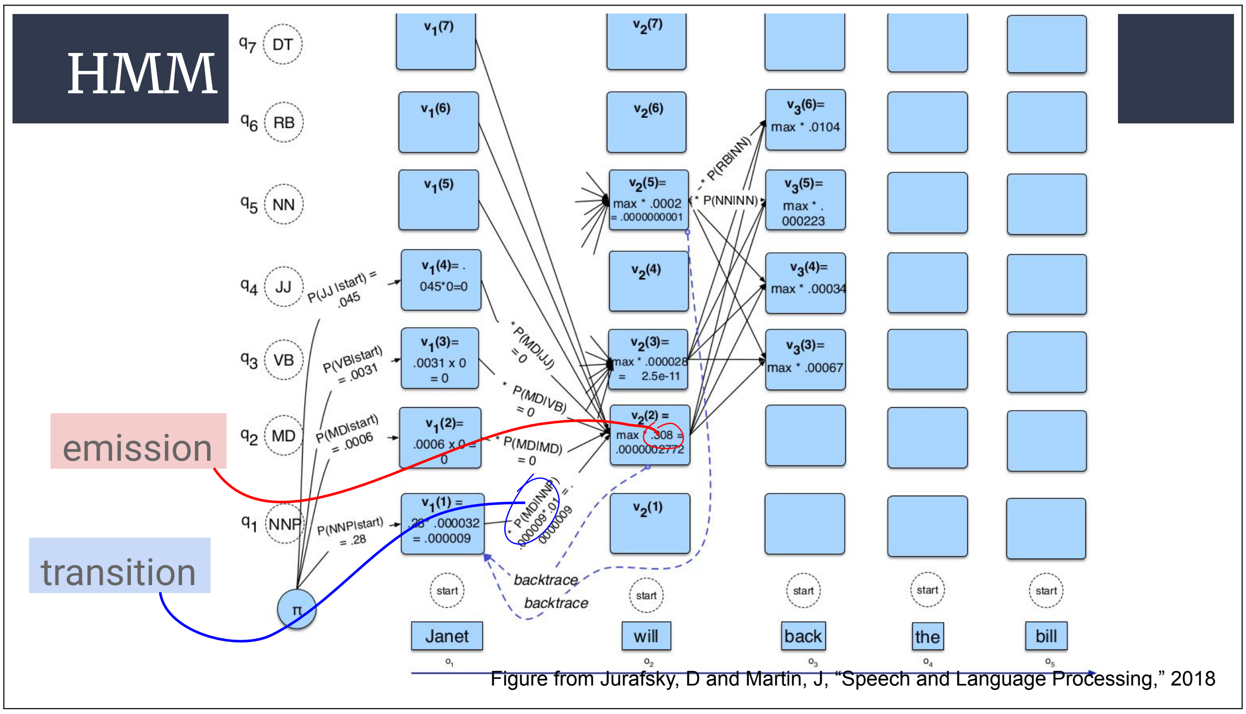
	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	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

(a) computed by counting



	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

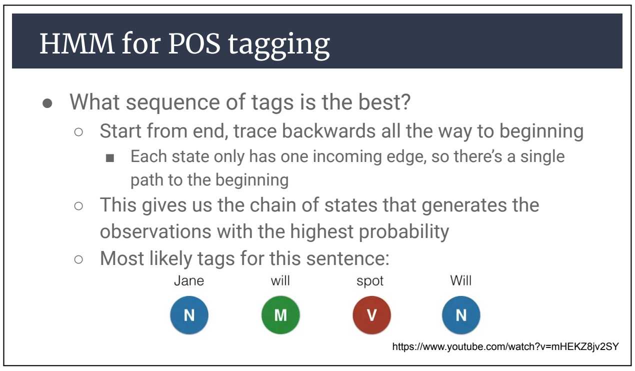
(b) emission probabilities, e.g. we have never observed the word 'Janet' that is a verb



1. We have starting probabilities. e.g. what is the probability that we will transition into a noun phrase from the start token? It is given as .28 in the transition probabilities above.
2. Then we get to the blue box. Here, we compute the probability of the first word given the tag. e.g. the probability of the word 'Janet' given that it is a noun is 0.000032.

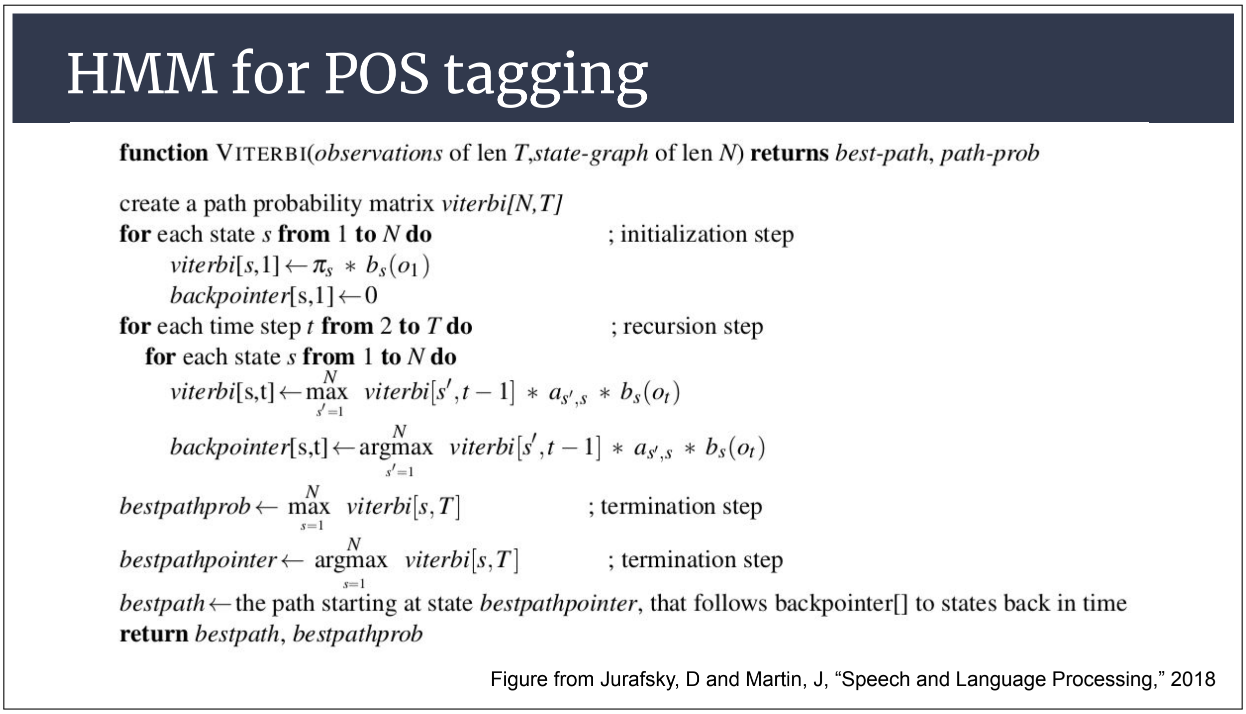
3. Then we transition from NNP to MD, and the probability of this transition is the probaility of Janet being an NNP multipled by MD following NNP which in the table above is 0.0110.
4. We then for each incoming array take the max and multiply it by the proabbility of will being an MD which is 0.308 from the emission table above.

We should realistically use log space instead of multiplying probabilities, but this is the general idea. This is becuase the probabilities are very small and multiplying them together will result in a very small number.



•

1.4.3 Pseudocode



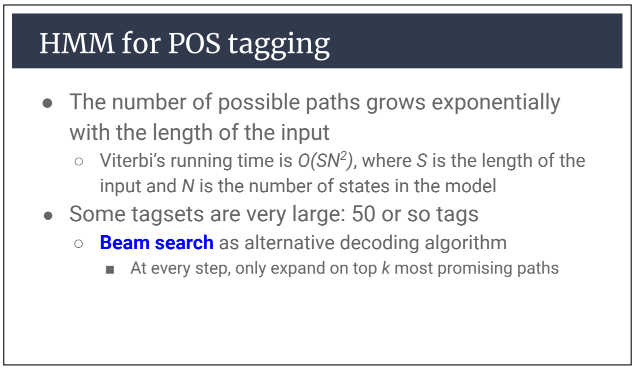
HMM for POS tagging

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

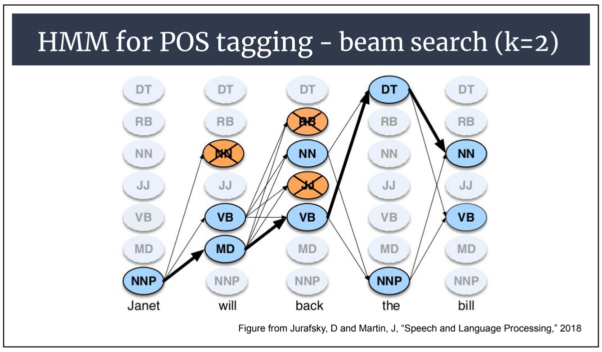
create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

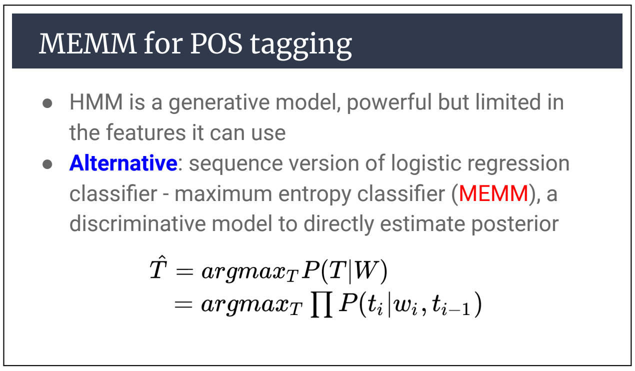
1.4.4 Problem with HMM



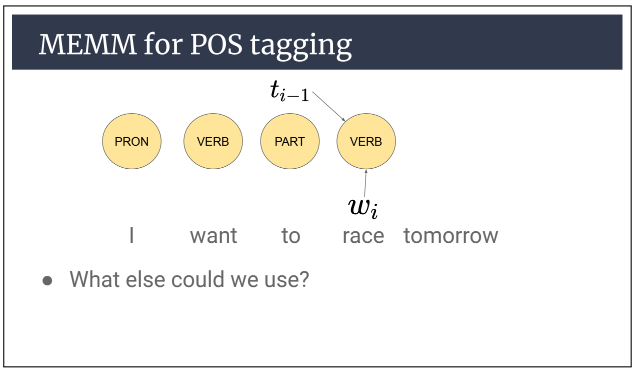
- becuase we're considering every possible path.
- This is a problem becuase we're considering every possible path, and this is not scalable.
- We can use a beam search to limit the number of paths we consider.



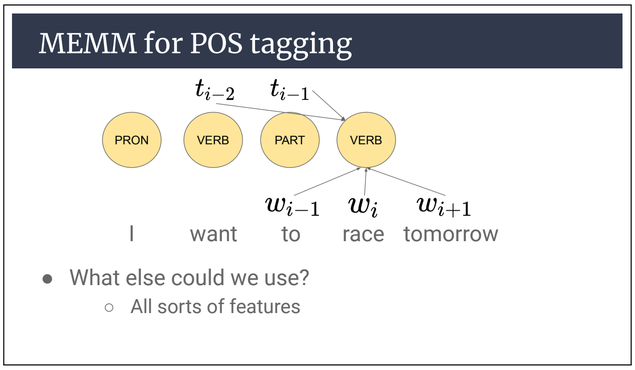
1.5 MEMM – Maximum entropy classifier

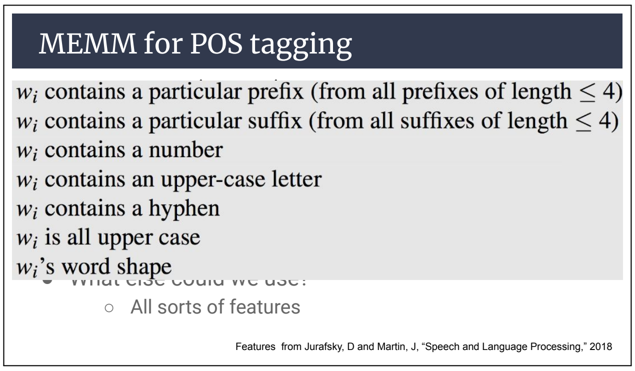


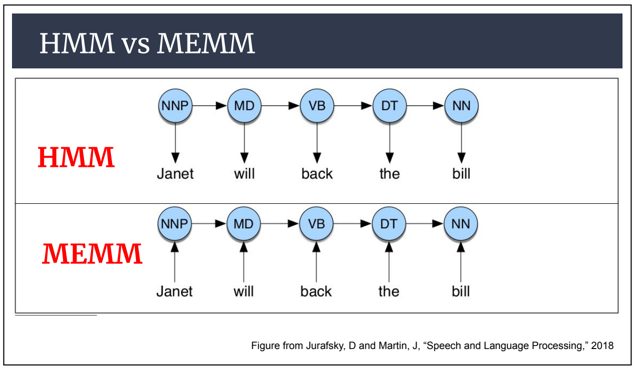
•

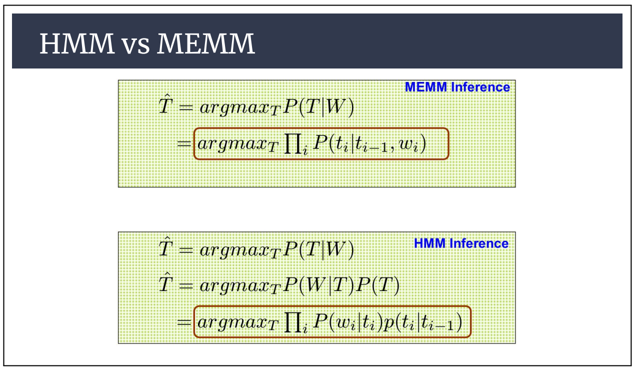


•


MEMM for POS tagging
t_{i-2}
t_{i-1}
PRON
VERB
PART
VERB
w_{i-1}
w_i
w_{i+1}
I
want
to
race
tomorrow
What else could we use?
All sorts of features


MEMM for POS tagging
w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i contains a number
w_i contains an upper-case letter
w_i contains a hyphen
w_i is all upper case
w_i's word shape
All sorts of features
Features from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018


HMM vs MEMM
HMM
NNP
MD
VB
DT
NN
Janet
will
back
the
bill
MEMM
NNP
MD
VB
DT
NN
Janet
will
back
the
bill
Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018


HMM vs MEMM
MEMM Inference
$\hat{T} = argmax_T P(T|W)$
$= argmax_T \prod_i P(t_i|t_{i-1}, w_i)$
HMM Inference
$\hat{T} = argmax_T P(T|W)$
$\hat{T} = argmax_T P(W|T)P(T)$
$= argmax_T \prod_i P(w_i|t_i)p(t_i|t_{i-1})$

HMM vs MEMM

- Versus VITERBI for HMM

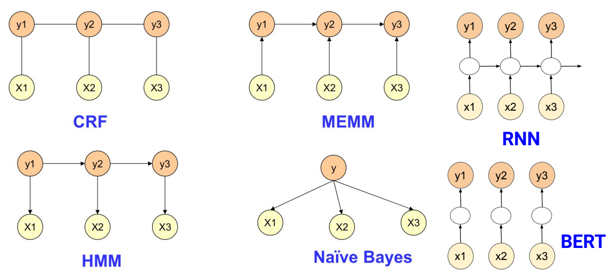
$$v_t(j) = max_{i=1}^{N} v_{t-1}(i) P(q_j|q_i) P(o_t|q_j)$$

- Viterbi for MEMM

$$v_t(j) = max_{i=1}^{N} v_{t-1}(i) P(q_j|q_i, o_t)$$

  - o_t could be any feature, not just words

Other approaches to POS tagging



RNN for POS tagging

- RNN to assign a label from (small) tagset to each word in the sequence
  - **Inputs**: word embedding per word
  - **Outputs**: tag probabilities from a softmax layer over tagset
  - **RNN**: 1 input, 1 output, 1 hidden layer; *U*, *V* and *W* shared

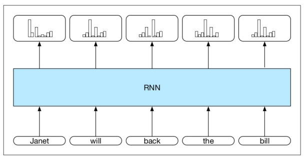


Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

RNN for POS tagging

- **Training**
  - Cross-entropy loss over the tagset for each word
  - Sequence loss is the sum of loss for all words
- **Inference**
  - Run forward inference over the input sequence and select the most likely tag from the softmax at each step
  - Decision for each word in the sequence is taken independently from decision for other words - not optimising for **sequence** of tags

•

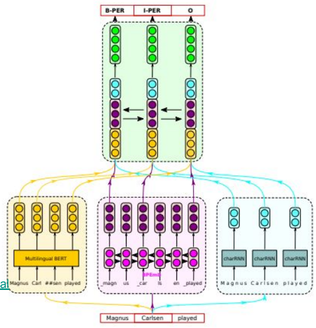
•

•

•

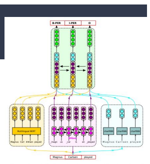
SOTA

- Using Universal Dependencies (UD) as tagset for various languages
- Also used for Named Entity Recognition



Sequence Tagging with Contextual and Non-Contextual Subword Representations: A Multilingual Evaluation.
Heinzerling† & Strube, 2019.

SOTA



- Ensemble of subword representations
- Input encoded using
  - Multilingual BERT (yellow, bottom left)
  - LSTM with BPEmb (pink, bottom middle)
  - Character-RNN (blue, bottom right)
- Meta-LSTM (green, center) combines the different encodings before classification (top)
- Horizontal arrows are bidirectional LSTMs.

SOTA for POS tagging - high res langs.

Previous SOTA

					BPEmb			BERT	
Lang.	BiLSTM	Adv.	FastText	BPEmb	+char	+shape	BERT	+char	+char+BPemb
Avg.	96.4	96.6	95.6	95.2	96.4	95.7	95.6	96.3	**96.8**
bg	98.0	98.5	97.7	97.8	98.5	97.9	98.0	98.5	**98.7**
cs	98.2	98.8	98.3	98.5	98.9	98.7	98.4	98.8	**99.0**
da	96.4	96.7	95.3	94.9	96.4	95.9	95.8	96.3	**97.2**
de	93.4	**94.4**	90.8	92.7	93.8	93.5	93.7	93.8	**94.4**
en	95.2	95.8	94.3	94.2	95.5	94.9	95.0	95.5	**96.1**
es	95.7	96.4	96.3	96.1	96.6	96.0	96.1	96.3	**96.8**
eu	95.5	94.7	94.6	94.3	**96.1**	94.8	93.4	95.0	96.0
fa	**97.5**	**97.5**	97.1	95.9	97.0	96.0	95.7	96.5	97.3
fi	**95.8**	95.4	92.8	92.8	94.4	93.5	92.1	93.8	94.3
fr	96.1	**96.6**	96.0	95.5	96.1	95.8	96.1	96.5	96.5
he	97.0	**97.4**	97.0	96.3	96.8	96.0	96.5	96.8	97.3
hi	97.1	97.2	97.1	96.9	97.2	96.9	96.3	96.8	**97.4**
hr	**96.8**	96.3	95.5	93.6	95.4	94.5	96.2	96.6	**96.8**
id	93.4	**94.0**	91.9	90.7	93.4	93.0	92.2	93.0	93.5
it	98.0	**98.1**	97.4	97.0	97.8	97.3	97.5	97.9	98.0
nl	93.3	93.1	90.0	91.7	93.2	92.5	91.5	92.6	93.3
no	98.0	98.1	97.4	97.0	98.2	97.8	97.5	98.0	**98.5**
pl	97.6	97.6	96.2	95.8	97.1	96.1	96.5	**97.7**	97.6
pt	97.9	98.1	97.3	96.3	97.7	97.2	97.5	97.8	98.1
sl	96.8	**98.1**	97.1	96.2	97.7	96.8	96.3	97.4	97.9
sv	96.7	96.7	96.7	95.3	96.7	95.7	96.2	97.1	**97.4**