

POS tagging

Author: Anton Zhitomirsky

Contents

1 Part Of Speech Tagging	2
1.1 Why do we need POS tagging?	2
1.2 Baseline method	3
1.2.1 Ambiguities	3
1.3 Probabilistic POS tagging	3
1.3.1 Example	4
1.3.2 Example Problem	6
1.4 Hidden Markov Model Tagger	7
1.4.1 Viterbi algorithm	8
1.4.2 Example	9
1.4.3 Pseudocode	10
1.4.4 Problem with HMM	11
1.5 MEMM – Maximum entropy classifier	11

1 Part Of Speech Tagging

POS tagging – tagset										
I	saw	the	boy	on	the	hill	with	a	telescope	.
PRON	VERB	DET	NOUN	ADP	DET	NOUN	ADP	DET	NOUN	PUNCT

Open class tags	Closed class tags	Other tags
ADJ	PREP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Tagset: Universal POS tags:
<https://universaldependencies.org/u/pos/all.html>

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

POS tagging – tagset	
•	ADJ (adjective): old, beautiful, smarter, clean...
•	ADV (adverb): slowly, there, quite, gently, ...
•	INTJ (interjection): psst, ouch, hello, ow
•	NOUN (noun): person, cat, mat, baby, desk, play
•	PROPN (proper noun): UK, Jack, London
•	VERB (verb): enter, clean, play
•	PUNCT (punctuation): . , ()
•	SYM (symbol): \$, %, §, ©, +, -, ×, ÷, =, <, >, :, ❤️, 🍷, 🤔
•	X (other): ? (code switching)

POS tagging – tagset	
•	PREP (preposition): in, on, to, with
•	AUX (auxiliary verb): can, shall, must
•	CCONJ (coordinating conjunction): and, or
•	DET (determiner): a, the, this, which, my, an
•	NUM (numeral): one, 2, 300, one hundred
•	PART (particle): off, up, 's, not
•	PRON (pronouns): he, myself, yourself, nobody
•	SCONJ (subordinating conjunction): that, if, while

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

1.1 Why do we need POS tagging?

Discussion	
•	Why do we need PoS tagging? <ul style="list-style-type: none"> ○ For NER: entities are nouns I ○ Pre-processing can be based on POS <ul style="list-style-type: none"> ■ E.g. select adjectives for sentiment analysis ○ Think more generally about sequence labelling ○ For neural syntactic and semantic parsing
•	PoS tagging is a solved problem for mainstream languages: Spacy, NLTK, Stanza

- 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

POS tagging – baseline method

- Naive approach
 - Assign each word its most frequent POS tag
 - Assign all unknown words the tag NOUN
- ~90% accuracy!
- There are exceptions....
- But frequency still plays a role
 - Probabilistic POS taggers

- 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

POS ambiguities

back

The back/ADJ door

On my back/NOUN

Win the voters back/ADV

Promised to back/VERB the bill

POS ambiguities

Flies like a flower

Flies/VERB/NOUN

like/PREP/ADV/CONJ/NOUN/VERB

a/DET

flower/NOUN/VERB

1.3 Probabilistic POS tagging

Probabilistic POS tagging

- Use frequencies but take context into account
- Given a **sequence** of **words** $W = w_1, w_2, \dots, w_n$
 - Estimate the **sequence** of **POS tags** $T = t_1, t_2, \dots, t_n$
 - Compute $P(T|W)$

Instance of **many-to-many**
classification

- Use the frequencies but take the context into account.

Probabilistic POS tagging – generative

- Generative approach (Bayes):

$$P(T|W) = \frac{P(W|T)P(T)}{P(W)} = P(W|T)P(T)$$
- Chain rule + **markov** assumption (e.g. **bigram**):

$$P(T) \approx P(t_1)P(t_2|t_1)P(t_3|t_2) \dots P(t_n|t_{n-1})$$
- Each word depends only on its tag (not on previous words)

$$P(W|T) \approx P(w_1|t_1)P(w_2|t_2) \dots P(w_n|t_n)$$

LM of
tags!

Like machine
translation!

- The probability of a word sequence given the word sequence is 1 because 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:

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

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

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

	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"; $P(\text{fast}|\text{ADV}) = 1$; $P(\text{ADV}|\text{VERB}) = 1/6$

	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	1

$P(\text{ADV}|\text{NOUN}) = 0$

- 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 because 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.

Therefore, 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"; $P(\text{fast}|\text{ADV}) = 1$; $P(\text{ADV}|\text{VERB}) = 1/6$

	John	wants	to	race	this	race	fast
PROPN							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"; $P(\text{fast}|\text{ADV}) = 1$; $P(\text{ADV}|\text{VERB}) = 1/6$

	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

$P(\text{ADV}|\text{VERB}) = 1/6$

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 $P(T|W)$

$$\hat{T} = \operatorname{argmax}_T P(T|W)$$

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

1.4 Hidden Markov Model Tagger

HMM tagger

- A Hidden Markov Model (**HMM**) allows inferring hidden states from observations:

$Q = q_1 q_2 \dots q_N$	A set of N states (tags)
$A = a_{11} a_{12} \dots a_{NN}$	A transition probability matrix A , each a_{ij} representing the probability P of moving from state i to state j , s.t. $\sum_{j=1}^N a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \dots o_T$	a sequence of T observations (words), each one drawn from a vocabulary $V = v_1, v_2, \dots, v_V$
$B = b_i(o_i)$	a sequence of observation likelihoods (emission probabilities), each expressing the probability of an observation o_i being generated from a state i
$\pi = \pi_1, \dots, \pi_N$	π_i is the probability that the chain will start in state i $\sum_{i=1}^N \pi_i = 1$

- Markov Chains:** Model probabilities of sequences of random variables (states)
- 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 infer hidden states from observations.

HMM tagger

- Again: strong assumptions
 - Markov:** to predict the future **tag** in the sequence, all that matters is the current state (bigrams of tags)
 - Can be extended to trigrams
 - Independence:** the probability of an output observation (**word**) o_i depends only on the state that produced the observation q_i
 - Not on previous observations O_{i-1}

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

HMM tagger

- Formulation is same as before:

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

$$\approx P(t_1)P(w_1|t_1)P(t_2|t_1)P(w_2|t_2) \dots P(t_n|t_{n-1})P(w_n|t_n)$$



- 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).

Probabilistic POS tagging

t_0

$P(\underline{t}_i | \underline{t}_{i-1}) \quad P(t_i | t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$

	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					2/6
PART (2)	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

Transition probabilities

Probabilistic POS tagging

$P(\underline{w}_i | \underline{t}_i) \quad P(w_i | t_i) = \frac{C(w_i, t_i)}{C(t_i)}$

	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	0	2/2	0	0	0	0	0
NOUN (2)	0	0	0	0	0	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

Emission probabilities

These two tables are the HMM model

Figure 2: Tables are unchanged

HMM for POS tagging

- **Decoding/inference:** task of determining the **hidden state sequence** corresponding to the **sequence of observations**

- Given as input an **HMM model** λ and a sequence of observations $O = o_1 o_2 \dots o_T$ (test case), find the most probable sequence of states $Q = q_1 q_2 \dots q_T$

$$\hat{T} = \underset{T}{\operatorname{argmax}} P(T|W)$$

$$\approx \underset{T}{\operatorname{argmax}} \prod_{i=1}^N P(w_i | t_i) P(t_i | t_{i-1})$$

emission
transition

- Here, the hidden state is the tag, and the sequence 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

HMM for POS tagging

• **Viterbi**

$v_t(j)$ = Viterbi path

probability ('t' = column & j

= row), i.e. probability that

the HMM is in **state j**

(present POS tag) after

seeing the **first t**

observations (past words

for which lattice values have

been calculated) and passing

through the most **probable**

state sequence (previous

POS tag) $q_1 \dots q_{t-1}$

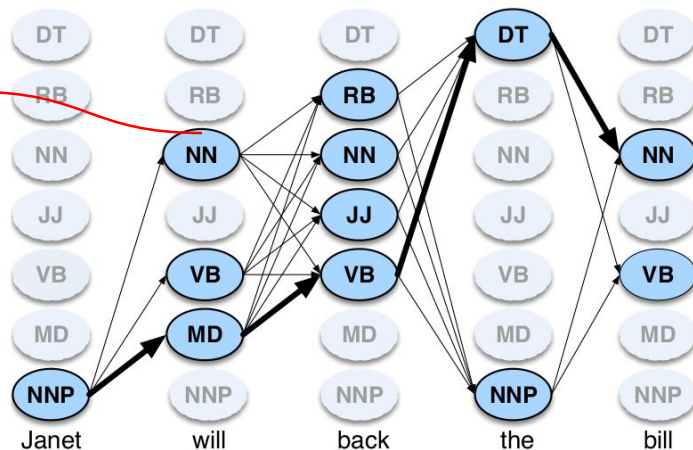


Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

Figure 3: For a particular state, what is the probability 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.

HMM for POS tagging

• **Viterbi algorithm:**

- Dynamic programming
- First build a lattice/matrix
 - One column per observation and one row per state
 - Each node $v_t(j)$ is the **probability** that the HMM is in state j after seeing the first t observations and passing through the most probable state sequence q_1, \dots, q_{t-1}

HMM for POS tagging

• **Viterbi algorithm:**

- The value of each $v_t(j)$ is computed by **recursively** taking the most probable path that could lead to this node:

$$v_t(j) = \max_{q_1, \dots, q_{t-1}} P(q_1 \dots q_{t-1}, o_1 \dots o_t, q_t = j)$$

- Probability of being in every state at time $t-1$ is already computed, so Viterbi probability is simply the most **probable of the extensions of the paths** that lead to the current cell

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

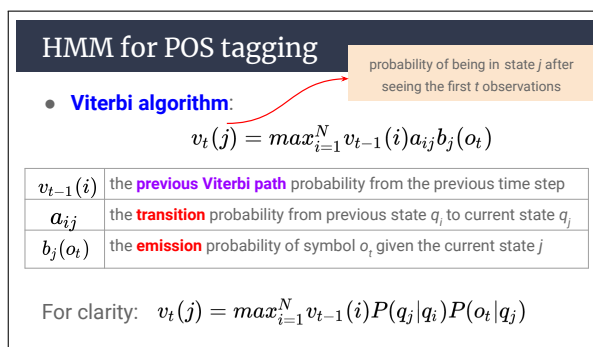


Figure 5: Recursive definition

1.4.2 Example

HMM for POS tagging

Counts should be smoothed

• Example: HMM λ (WSJ) - transition probabilities

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

(a) computed by counting

HMM for POS tagging

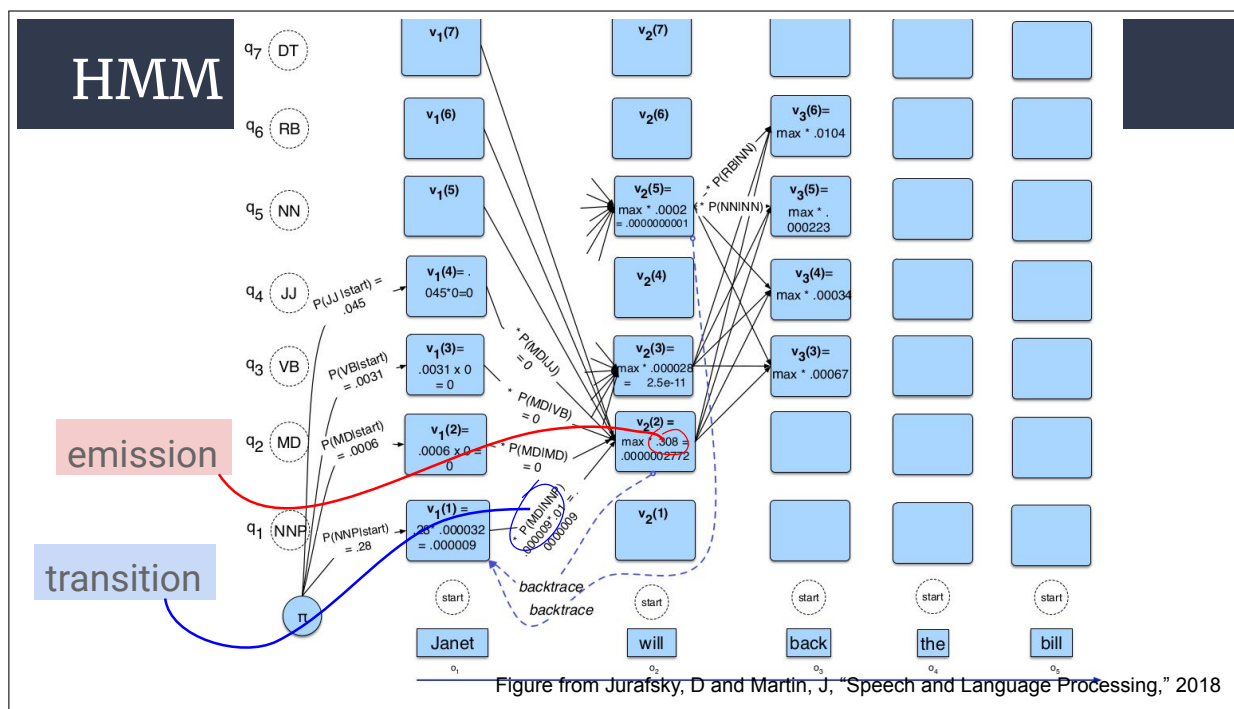
Counts should be smoothed

• Example: HMM λ (WSJ) - emission probabilities

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

(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 probability of Janet being an NNP multiplied 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 probability 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 because the probabilities are very small and multiplying them together will result in a very small number.

HMM for POS tagging

- What sequence of tags is the best?
 - Start from end, trace backwards all the way to beginning
 - Each state only has one incoming edge, so there's a single path to the beginning
 - This gives us the chain of states that generates the observations with the highest probability
 - Most likely tags for this sentence:

Jane
N

will
M

spot
V

Will
N

<https://www.youtube.com/watch?v=miHEKZ8jv2SY>

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
     $bestpath \leftarrow$  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

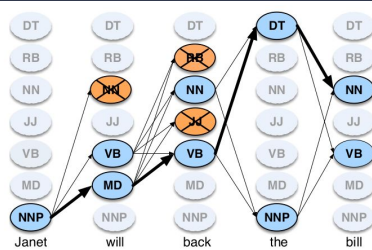
1.4.4 Problem with HMM

HMM for POS tagging

- The number of possible paths grows exponentially with the length of the input
 - Viterbi's running time is $O(SN^2)$, where S is the length of the input and N is the number of states in the model
- Some tagsets are very large: 50 or so tags
 - **Beam search** as alternative decoding algorithm
 - At every step, only expand on top k most promising paths

- because we're considering every possible path.
- This is a problem because 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.

HMM for POS tagging - beam search (k=2)



1.5 MEMM – Maximum entropy classifier

MEMM for POS tagging

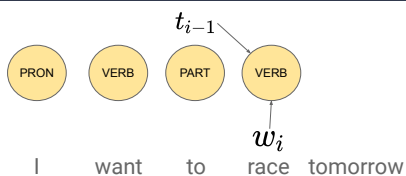
- HMM is a generative model, powerful but limited in the features it can use
- **Alternative**: sequence version of logistic regression classifier - maximum entropy classifier (**MEMM**), a discriminative model to directly estimate posterior

$$\hat{T} = \operatorname{argmax}_T P(T|W)$$

$$= \operatorname{argmax}_T \prod P(t_i | w_i, t_{i-1})$$

•

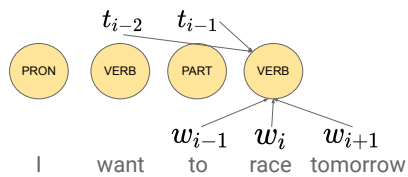
MEMM for POS tagging



- What else could we use?

•

MEMM for POS tagging



- 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

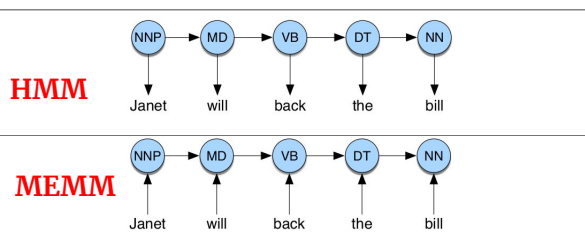


Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

HMM vs MEMM

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

$$\begin{aligned}
 \hat{T} &= \operatorname{argmax}_T P(T|W) && \text{HMM Inference} \\
 \hat{T} &= \operatorname{argmax}_T P(W|T)P(T) \\
 &= \operatorname{argmax}_T \prod_i P(w_i | t_i) p(t_i | t_{i-1})
 \end{aligned}$$

HMM vs MEMM

- Versus VITERBI for HMM

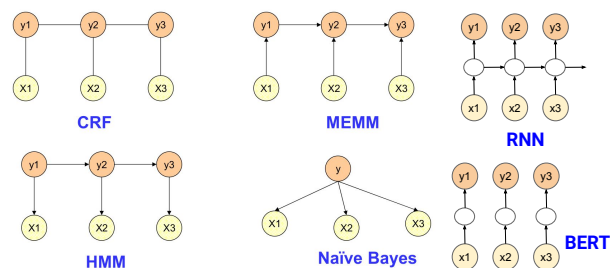
$$v_t(j) = \max_{i=1}^N v_{t-1}(i) P(q_j | q_i) P(o_t | q_j)$$

- Viterbi for MEMM

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

- o_i 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

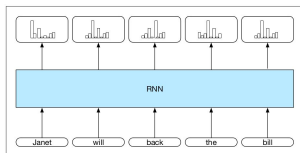


Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

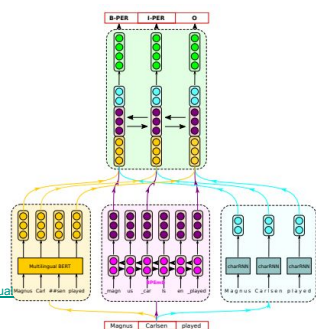
RNN for POS tagging

- **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

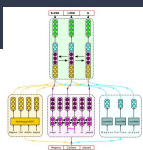
- Using Universal Dependencies (UD) as tagset for various languages
- Also used for Named Entity Recognition

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



SOTA

- Ensemble of subword representations
 - 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

Lang.	BiLSTM	Adv.	FastText	BPEmb	BPEmb +char	+shape	BERT	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