

# Natural Language Processing

Nuri Cingillioglu

<https://www.doc.ic.ac.uk/~nuric/>

Many thanks to Lucia Specia

# About

- Neuro-symbolic research
- Guest lecturer
- Used to teach Intro2ML
- NLP AI startup Semafind



# POS tagging

# Introduction

I saw the boy on the hill with a telescope.



<b>I</b>	<b>saw</b>	<b>the</b>	<b>boy</b>	<b>on</b>	<b>the</b>	<b>hill</b>	<b>with</b>	<b>a</b>	<b>telesc ope</b>	<b>.</b>
PRON	VERB	DET	NOUN	ADP	DET	NOUN	ADP	DET	NOUN	PUNCT

# POS tagging – tagset

I	saw	the	boy	on	the	hill	with	a	telesc ope	.
PRON	VERB	DET	NOUN	ADP	DET	NOUN	ADP	DET	NOUN	PUNCT

## Open class tags

ADJ
ADV
INTJ
NOUN
PROPN
VERB

## Closed class tags

PREP
AUX
CCONJ
DET
NUM
PART
PRON
SCONJ

## Other tags

PUNCT
SYM
X

Tagset: Universal POS tags:

<https://universaldependencies.org/u/pos/all.html>

# 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

# Discussion

- Why do we need PoS tagging?
  - For NER: entities are **nouns** |
  - Pre-processing can be based on POS
    - 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



# 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

# 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

# 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

# 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)

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

# Probabilistic POS tagging – generative

- Generative approach (Bayes):

$$P(T|W) = \frac{P(W|T)P(T)}{P(W)} = P(W|T)P(T)$$

LM of  
tags!

- 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)

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

Like machine  
translation!

# Probabilistic POS tagging – generative

- Putting both together:

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

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

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

# 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

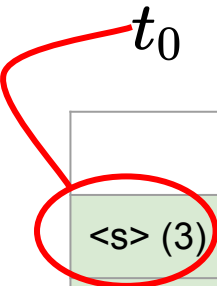


# Probabilistic POS tagging – generative

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

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

$t_0$



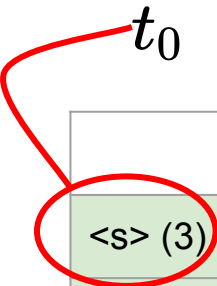
	PROPN	VERB	PART	NOUN	PRON	DET
<s> (3)	1/3	1/3	0	0	0	1/3
PROPN (1)	0	1/1	0	0	0	0
VERB (6)	0	1/6	1/6	0	0	2/6
PART (2)	0	1/2	0	0	0	1/2
NOUN (2)	0	0	0	0	1/2	0
PRON (1)	0	1/1	0	0	0	0
DET (4)	0	1/4	1/4	2/4	0	0

# Probabilistic POS tagging – generative

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

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

$t_0$



	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

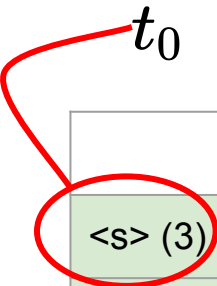
How to make the sum of each row **equal 1**?

# Probabilistic POS tagging – generative

$$P(\boxed{t_i} \boxed{t_{i-1}})$$

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

$t_0$



	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

# 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

# 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

# Probabilistic POS tagging

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

- “John wants to race this, race fast”;  $P(\text{fast}|\text{ADV}) = 1$ ;  $P(\text{ADV}|\text{VERB}) = 1/6$

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

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

# Probabilistic POS tagging

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

- “John wants to race this, race fast”;  $P(\text{fast}|\text{ADV}) = 1$ ;  $P(\text{ADV}|\text{VERB}) = 1/6$

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

	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

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

# 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  $P(T|W)$

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



# HMM tagger

- Let's improve on the approach using
  - **Markov Chains**: model probabilities of sequences of random variables (states)
  - **Hidden Markov Chains**: states are not given, but hidden
    - Words are observed
    - POS are hidden

# 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 <b>transition probability</b> 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 $V = v_1, v_2, \dots, v_V$
$B = b_i(o_t)$	a sequence of observation likelihoods ( <b>emission probabilities</b> ), each expressing the probability of an observation $o_t$ being generated from a state $i$
$\pi = \pi_1, \dots, \pi_N$	$\pi_i$ is the probability that the chain will start in state $i$ $\sum_{i=1}^N \pi_i = 1$

# 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  $o_{i-1}$

# HMM tagger

- Formulation is same as before:

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

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



Emission probabilities



Transition probabilities

# Probabilistic POS tagging

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

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

$t_0$

	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

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

	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	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

Emission probabilities

These two tables are the **HMM** model

# 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  $\lambda$  and a sequence of observations  $O = o_1 o_2 \dots o_T$  (test case), find the most probable sequence of states  $Q = q_1 q_2 \dots q_T$

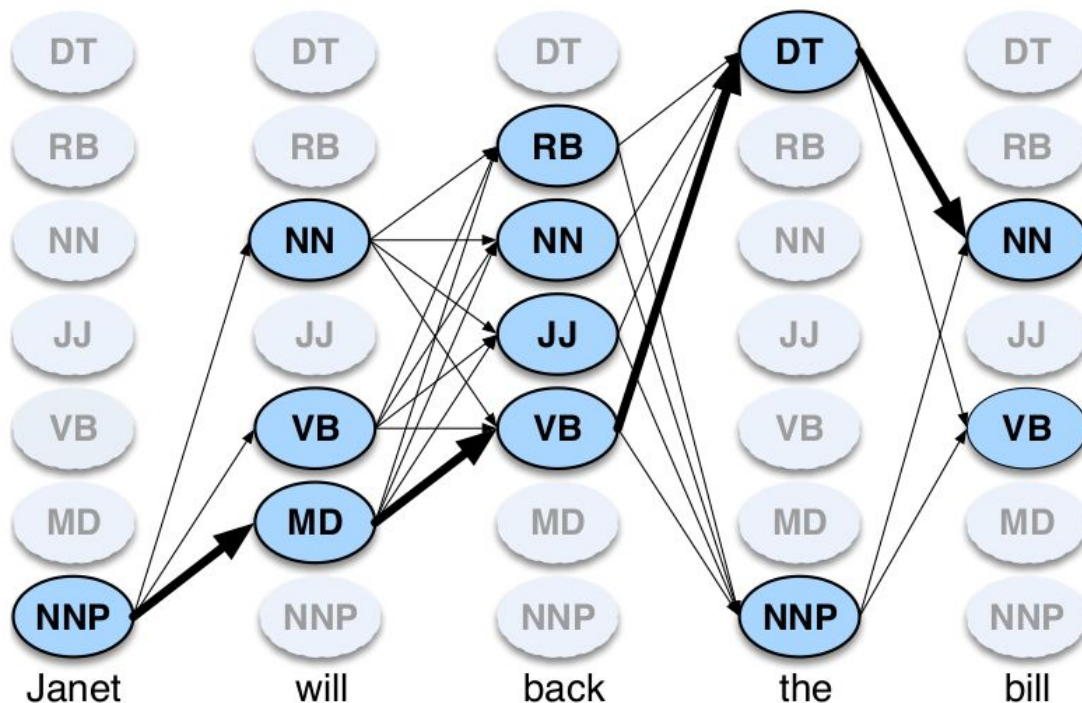
$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &\approx \operatorname{argmax} \prod_{i=1}^N P(w_i | t_i) P(t_i | t_{i-1})\end{aligned}$$

emission

transition

# HMM for POS tagging

- **Viterbi**





# 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)  $q_{-1}.....q_{-t-1}$

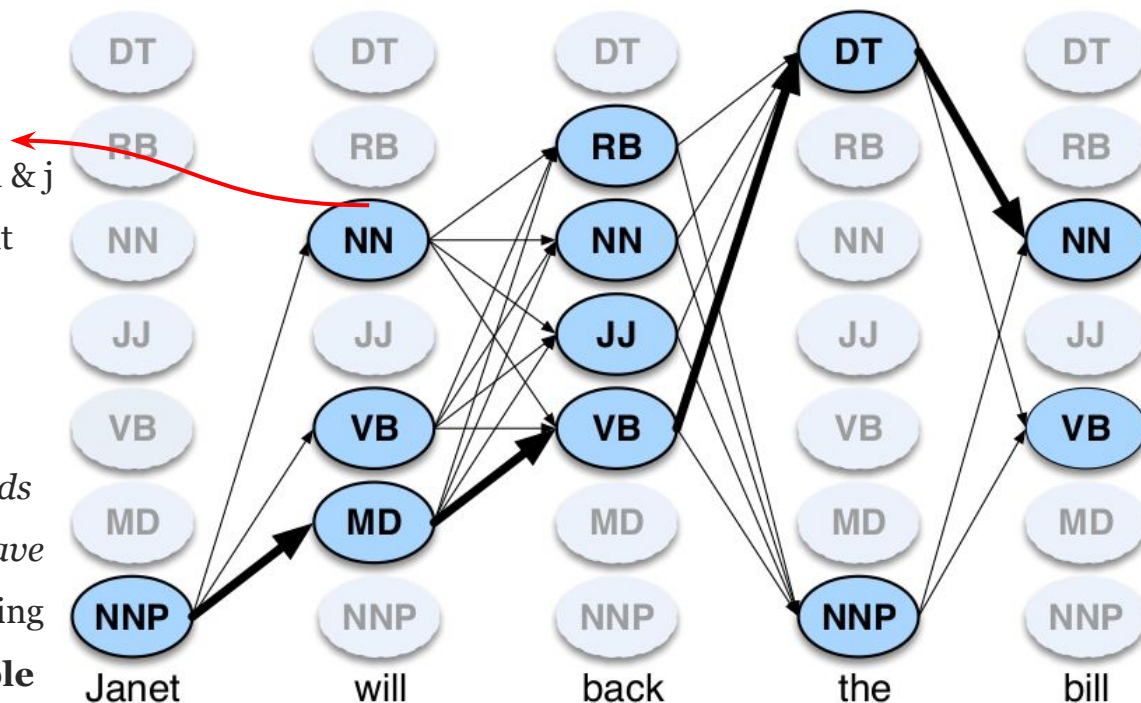


Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

# 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

# HMM for POS tagging

probability of being in state  $j$  after seeing the first  $t$  observations

- **Viterbi algorithm:**

$$v_t(j) = \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t)$$

$v_{t-1}(i)$	the <b>previous Viterbi path</b> probability from the previous time step
$a_{ij}$	the <b>transition</b> probability from previous state $q_i$ to current state $q_j$
$b_j(o_t)$	the <b>emission</b> probability of symbol $o_t$ given the current state $j$

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

# HMM for POS tagging

Counts should  
be smoothed

- Example: HMM  $\lambda$  (WSJ) - transition probabilities

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

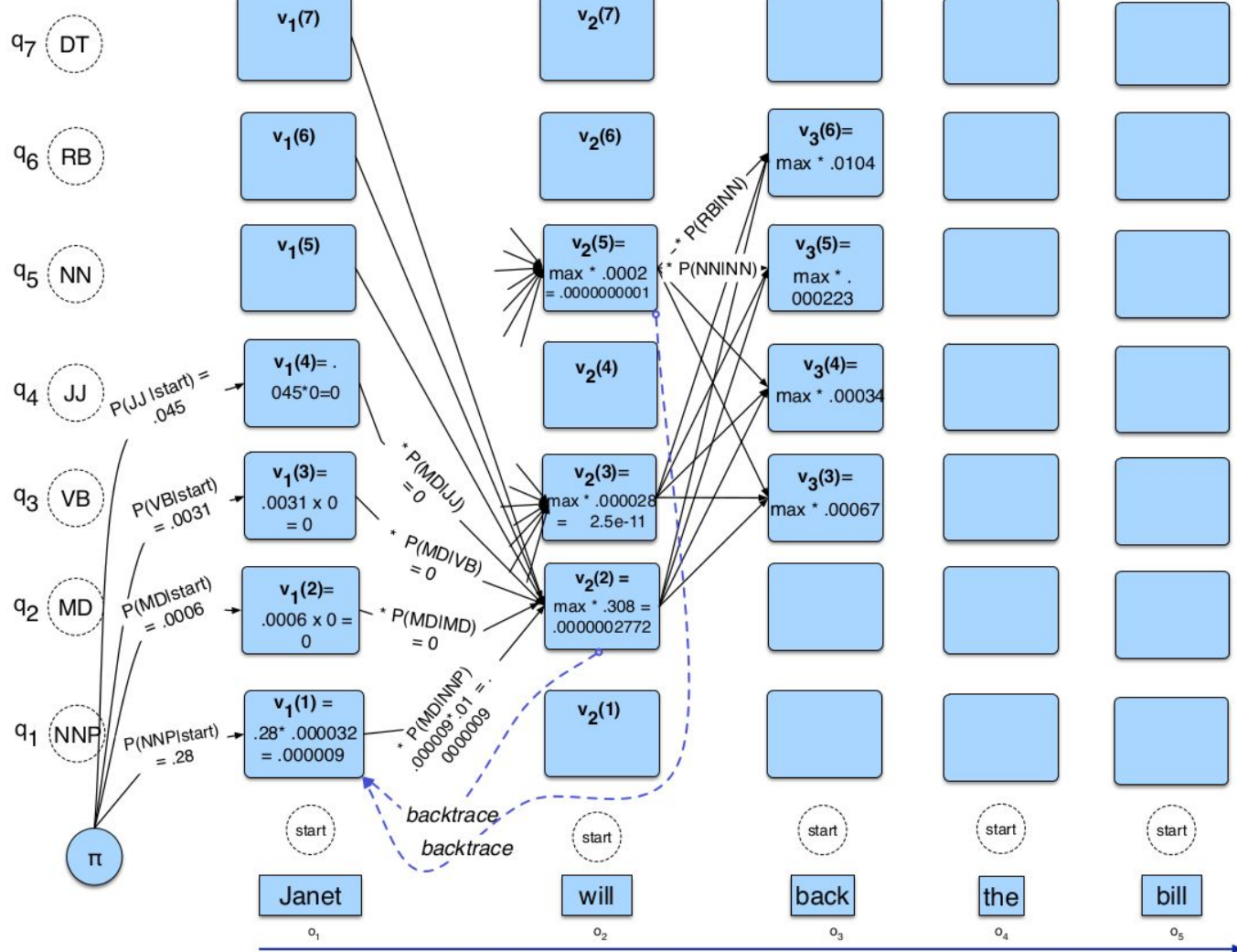
# HMM for POS tagging

Counts should  
be smoothed

- Example: HMM  $\lambda$  (WSJ) - emission probabilities

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

# HMM





# HMM

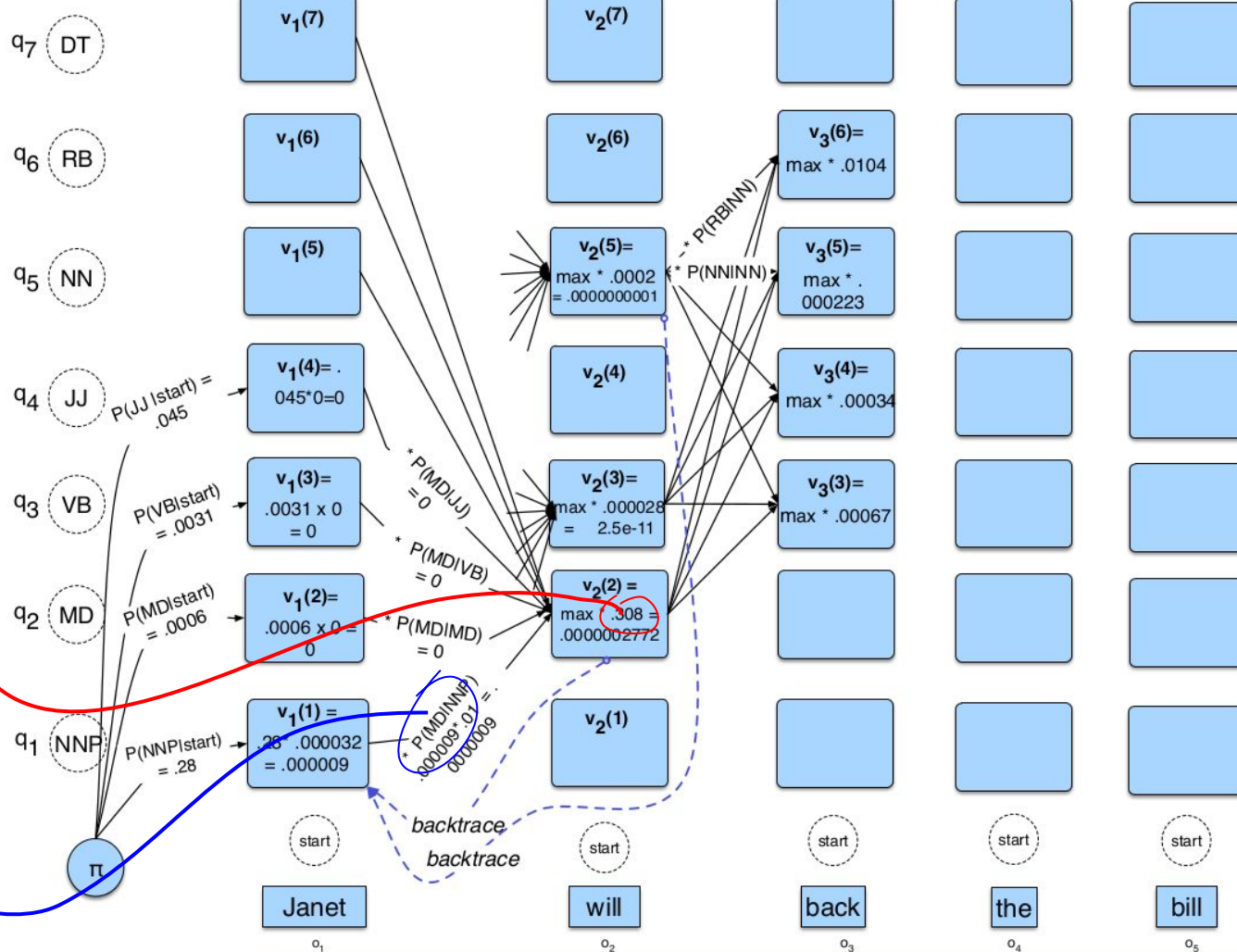


Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

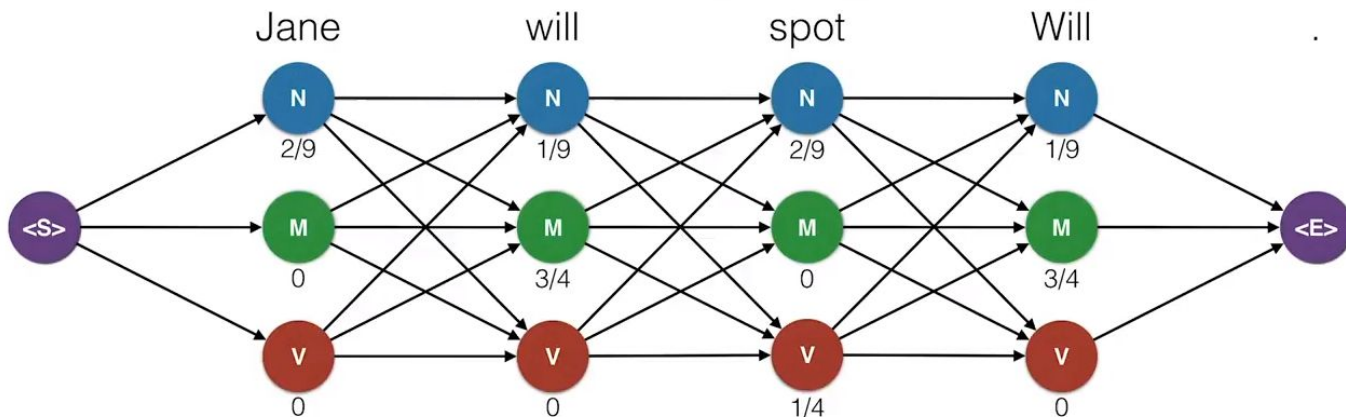


# HMM for POS tagging

Viterbi Algorithm

	N	M	V
Mary	4/9	0	0
Jane	2/9	0	0
Will	1/9	3/4	0
Spot	2/9	0	1/4
Can	0	1/4	0
See	0	0	1/2
Pat	0	0	1/4

	N	M	V	<E>
<S>	3/4	1/4	0	0
N	1/9	1/3	1/9	4/9
M	1/4	0	3/4	0
V	1	0	0	0

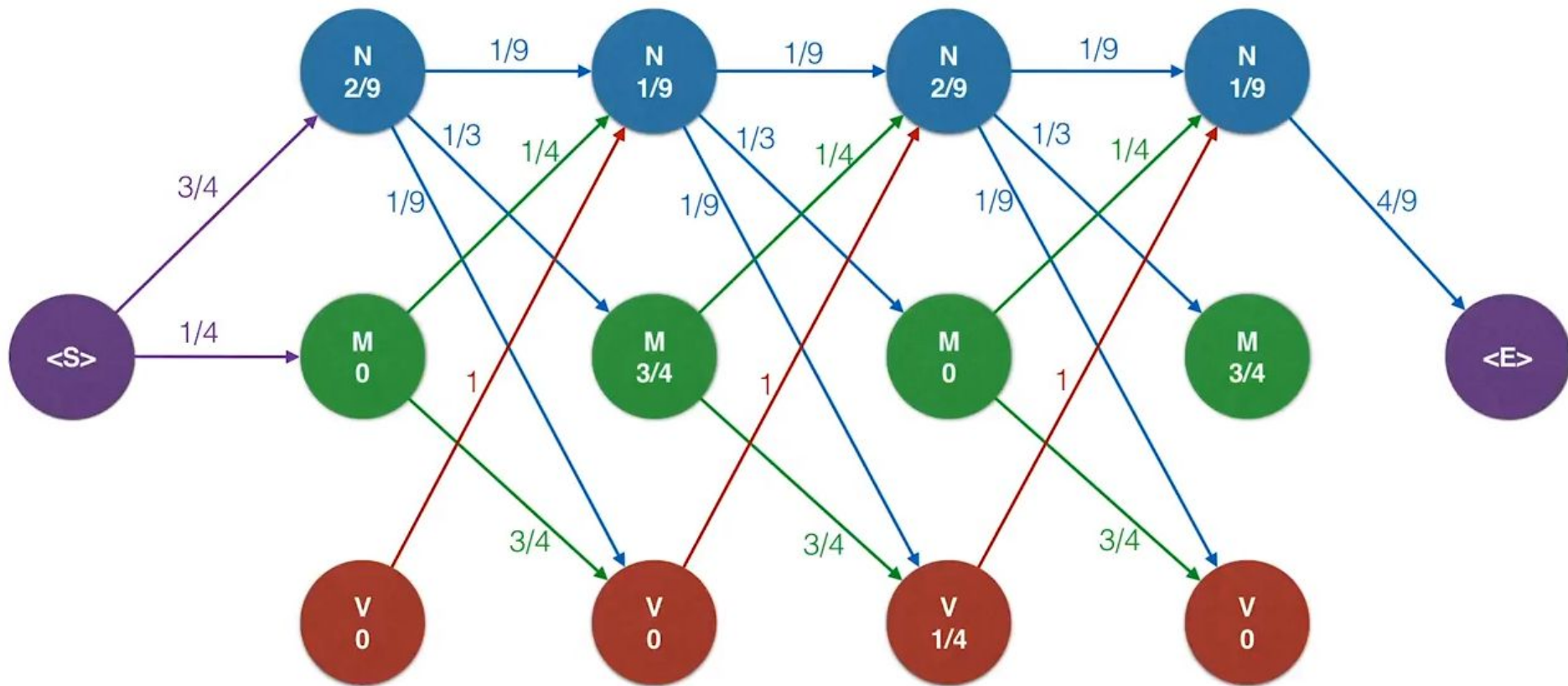


Jane

will

spot

Will



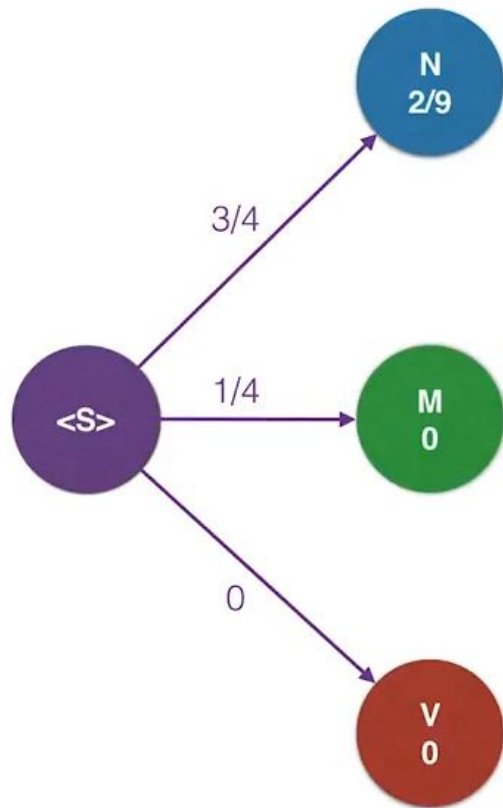
Jane

will

spot

Will

.



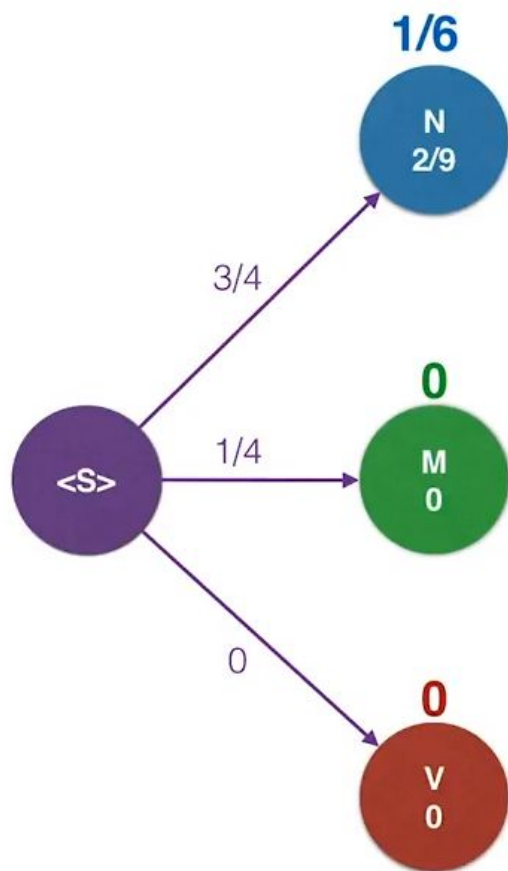
Jane

will

spot

Will

.



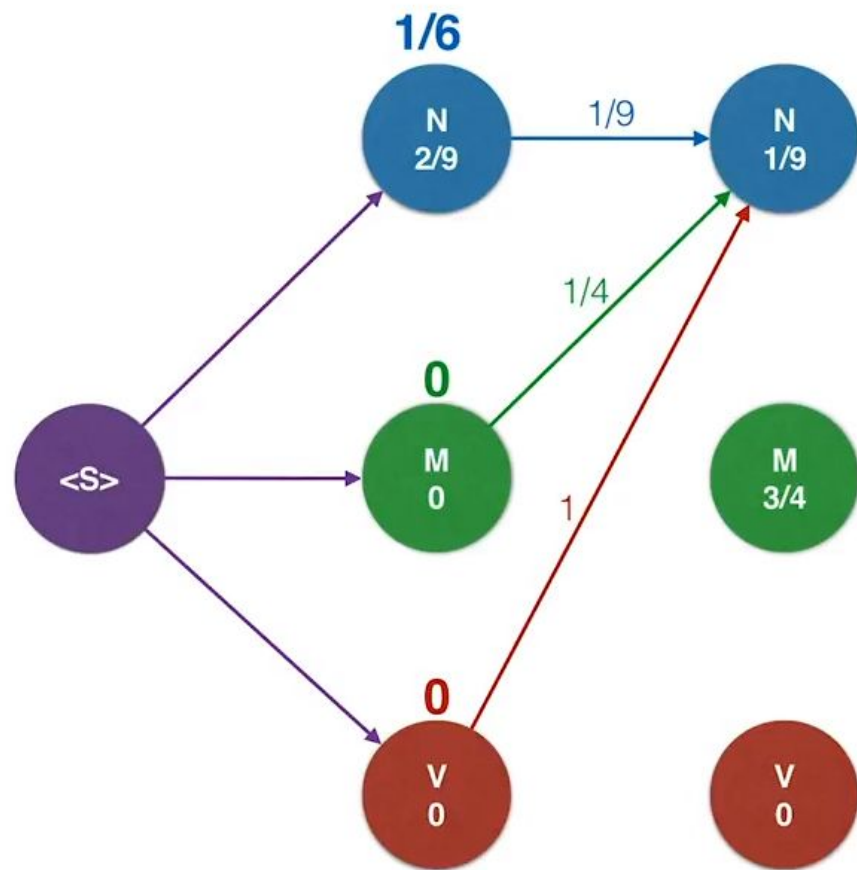
Jane

will

spot

Will

.



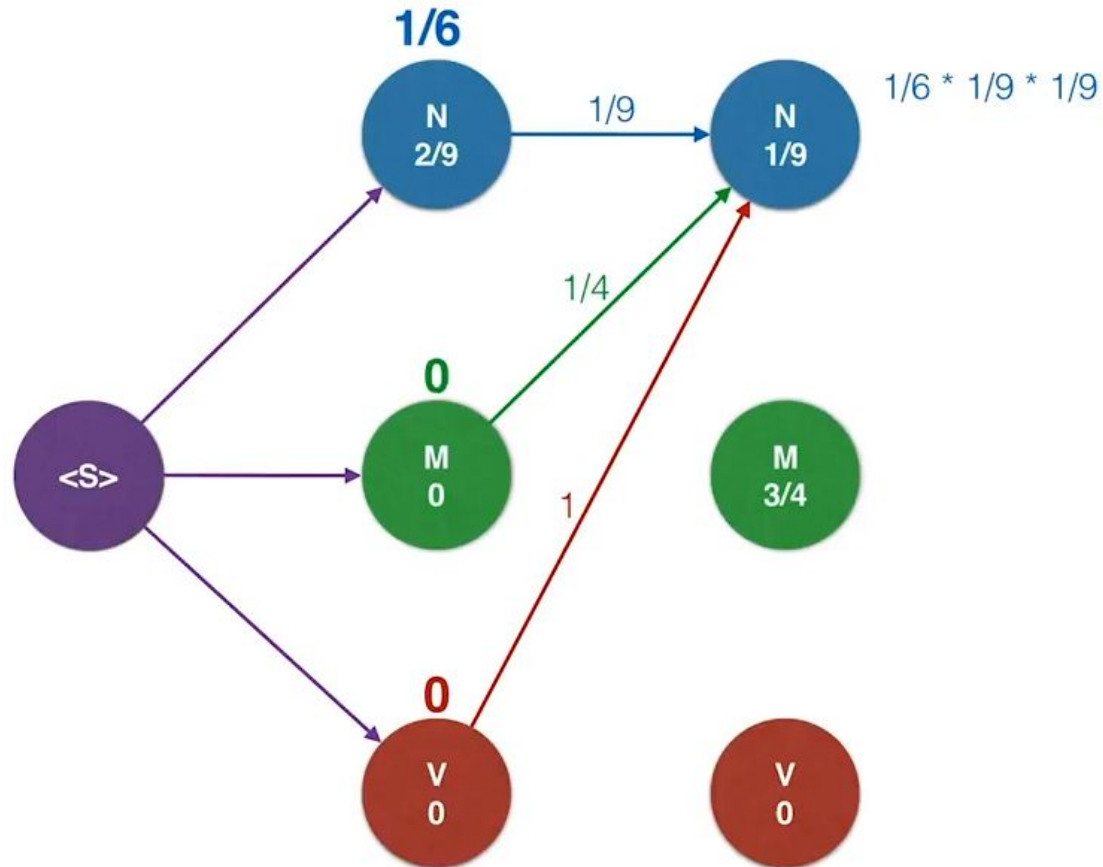
Jane

will

spot

Will

.



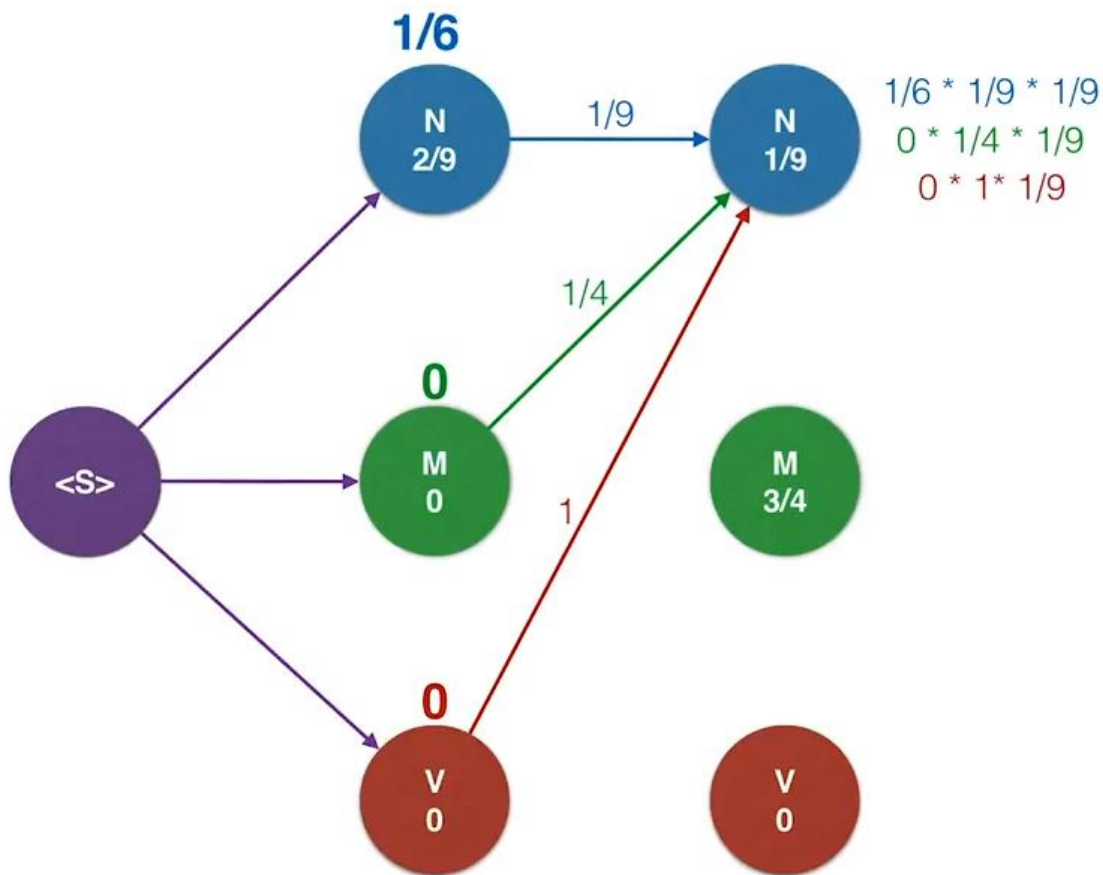
Jane

will

spot

Will

.



Jane

will

spot

Will

.

 $\frac{1}{6}$  $\frac{1}{486}$ 

$$\begin{aligned} & \frac{1}{6} * \frac{1}{9} * \frac{1}{9} \\ & 0 * \frac{1}{4} * \frac{1}{9} \\ & 0 * 1 * \frac{1}{9} \end{aligned}$$

 $\frac{1}{9}$  $\frac{1}{4}$ 

1

0

M  
0

0

V  
0N  
 $\frac{1}{9}$ M  
 $\frac{3}{4}$ V  
0

&lt;S&gt;





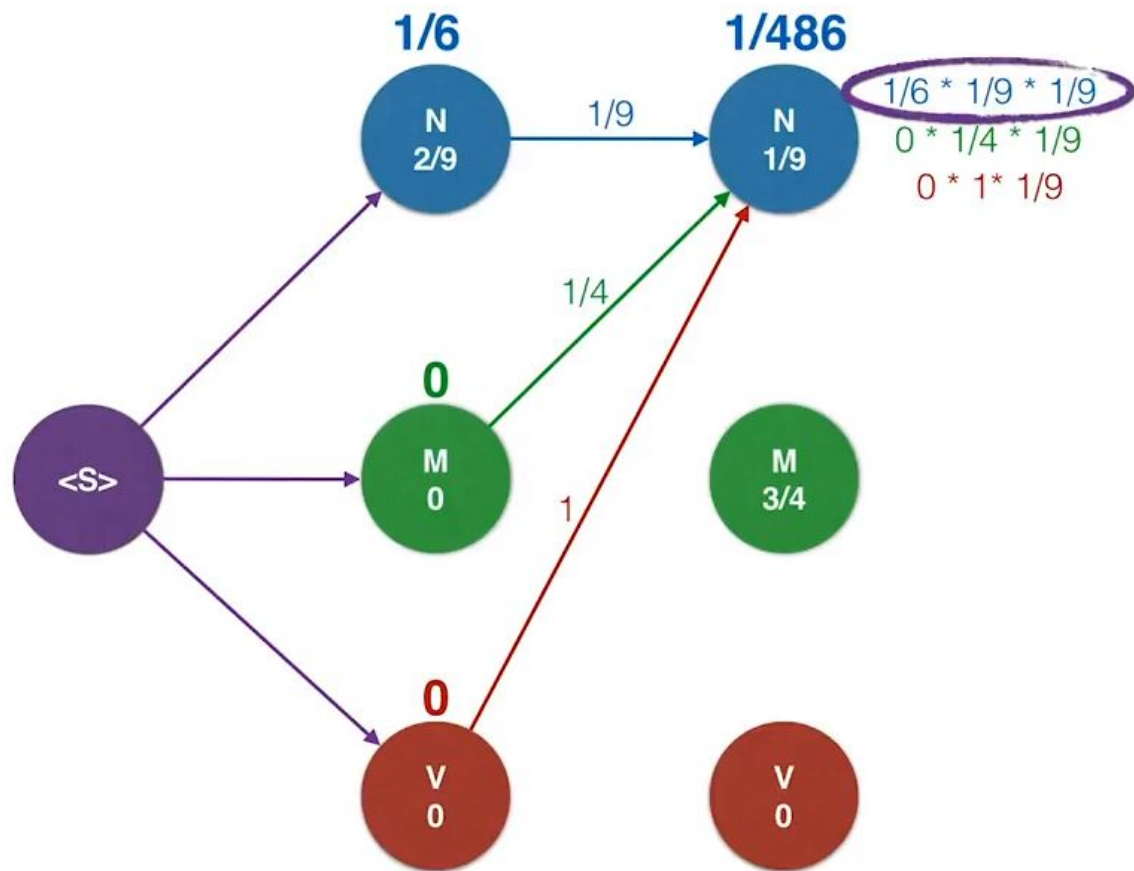
Jane

will

spot

Will

.



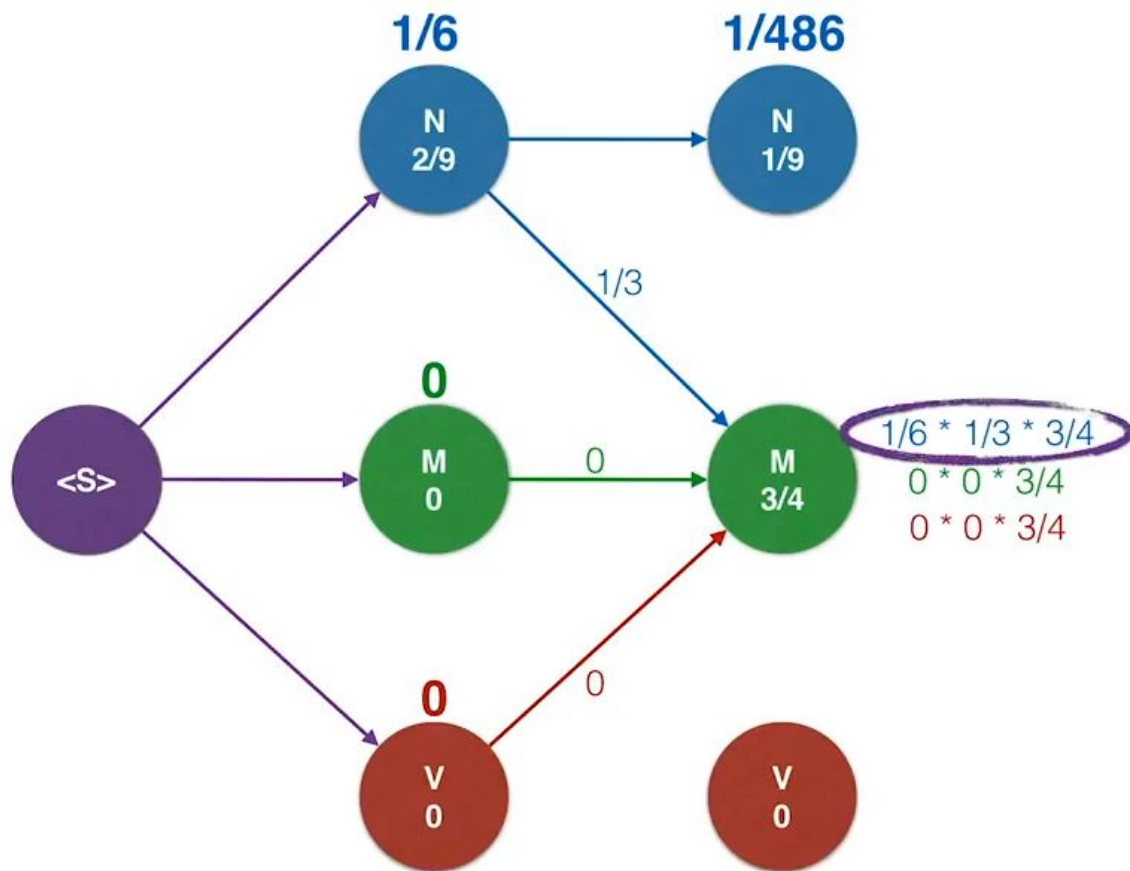
Jane

will

spot

Will

.



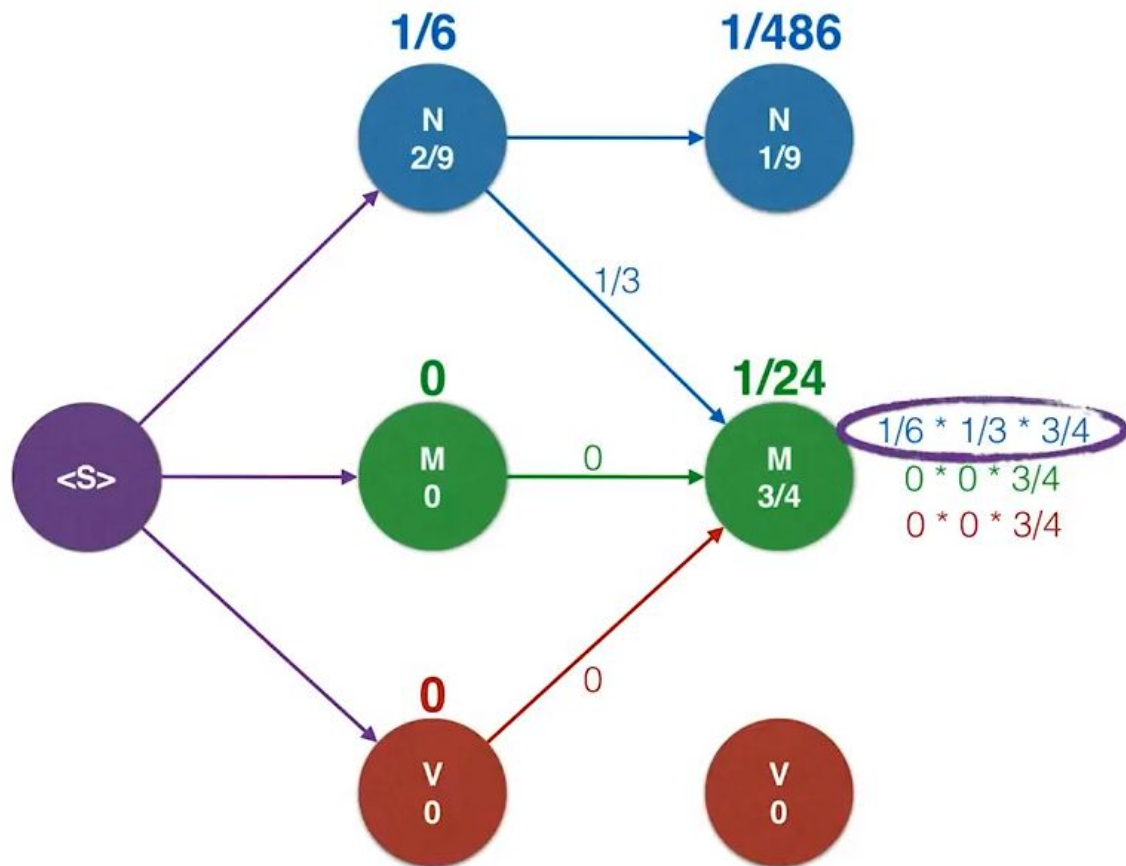
Jane

will

spot

Will

.



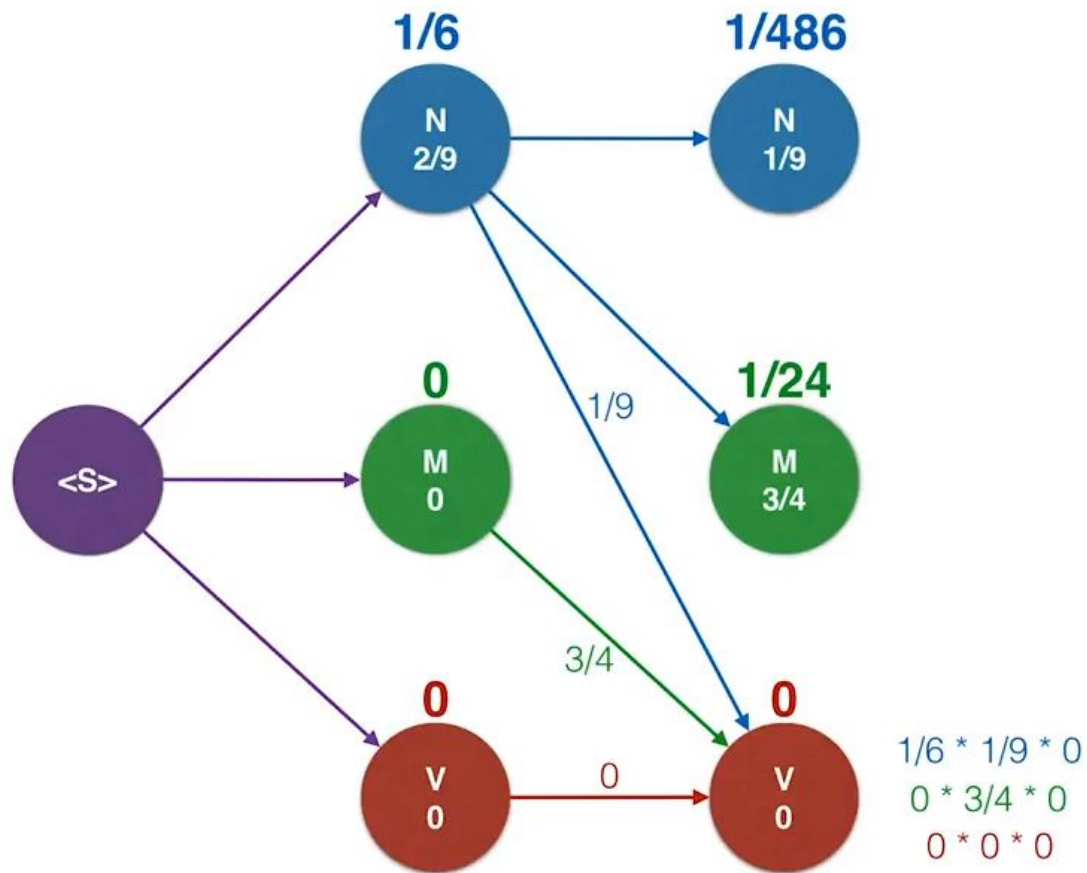
Jane

will

spot

Will

.



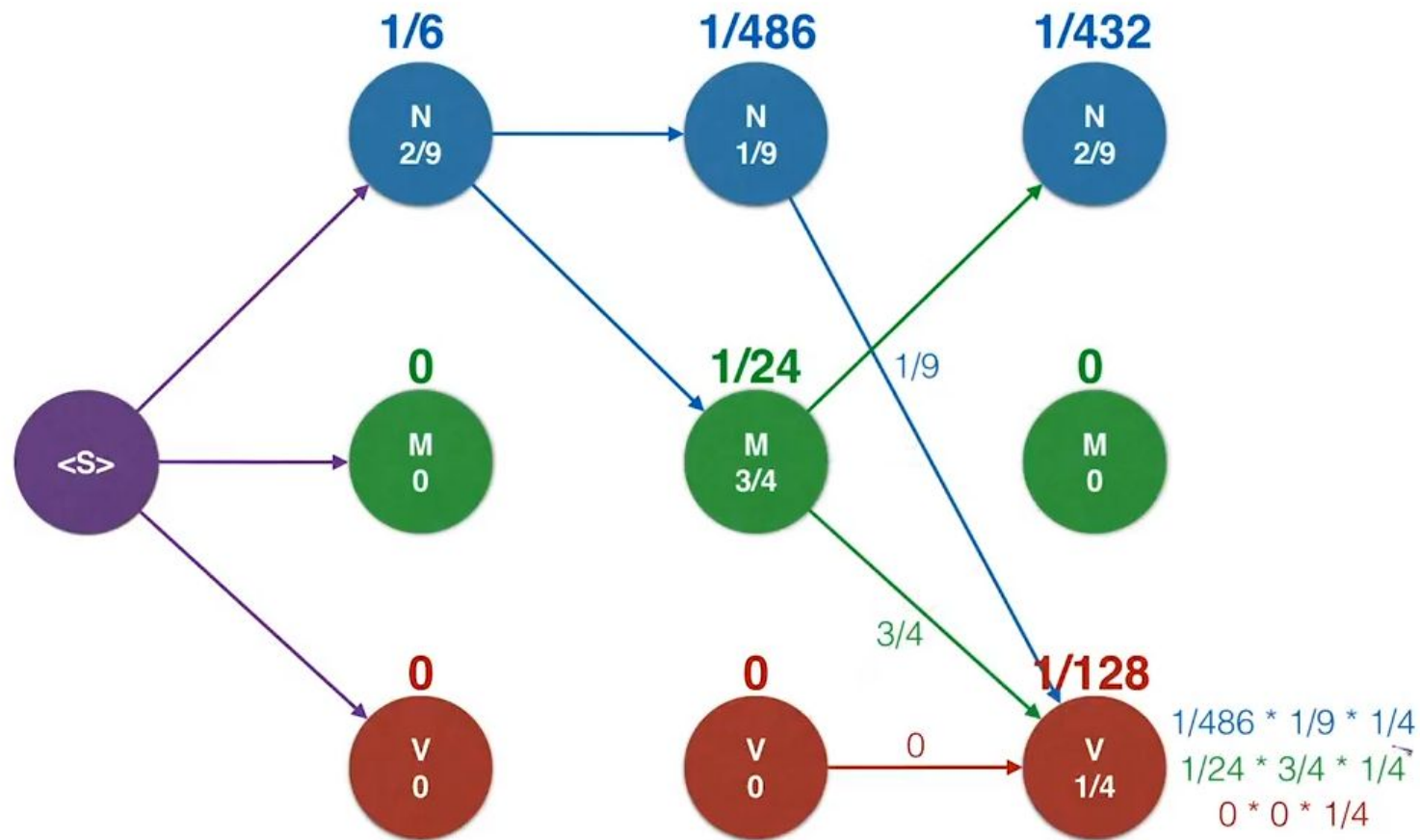
Jane

will

spot

Will

.



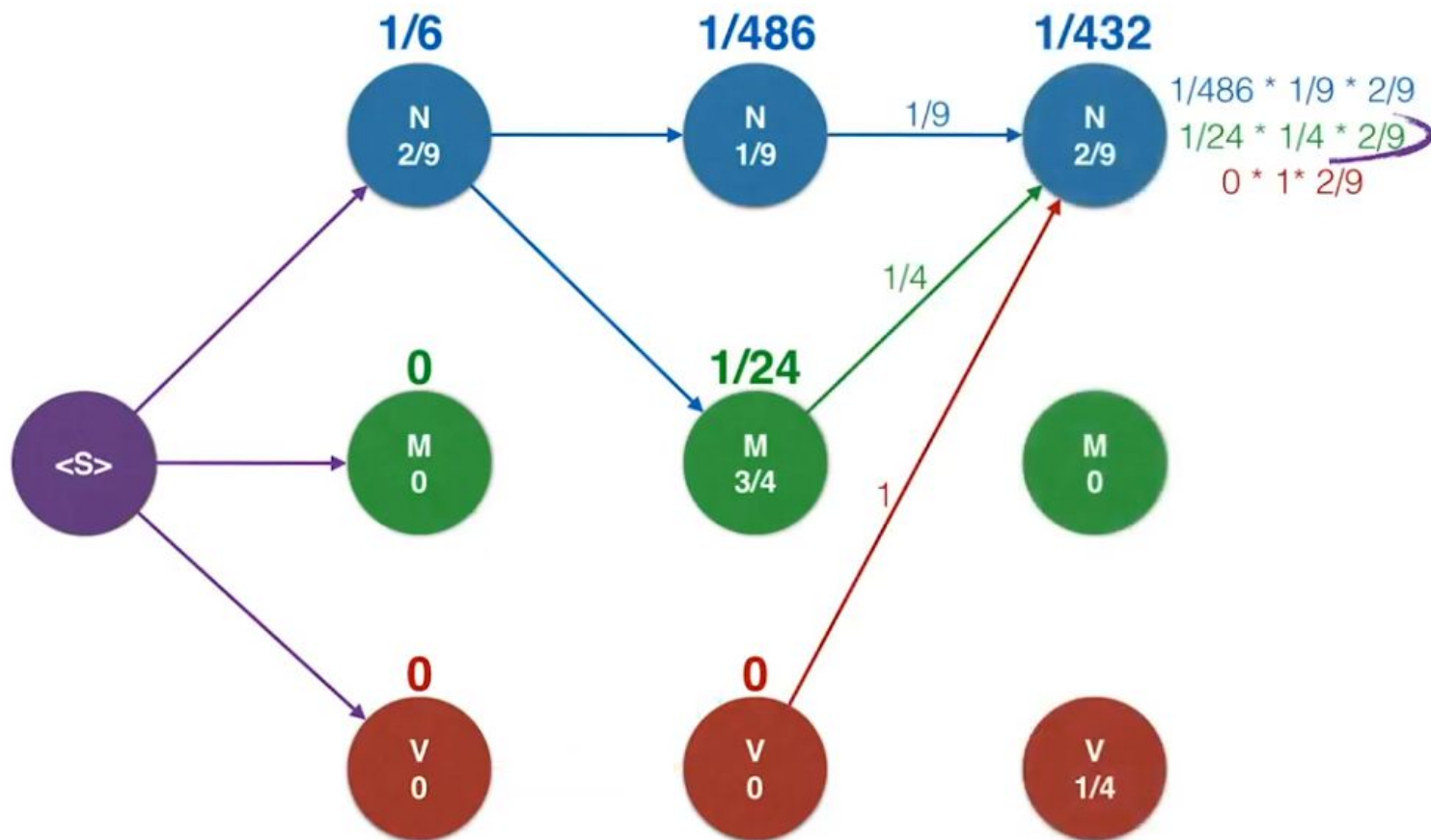
Jane

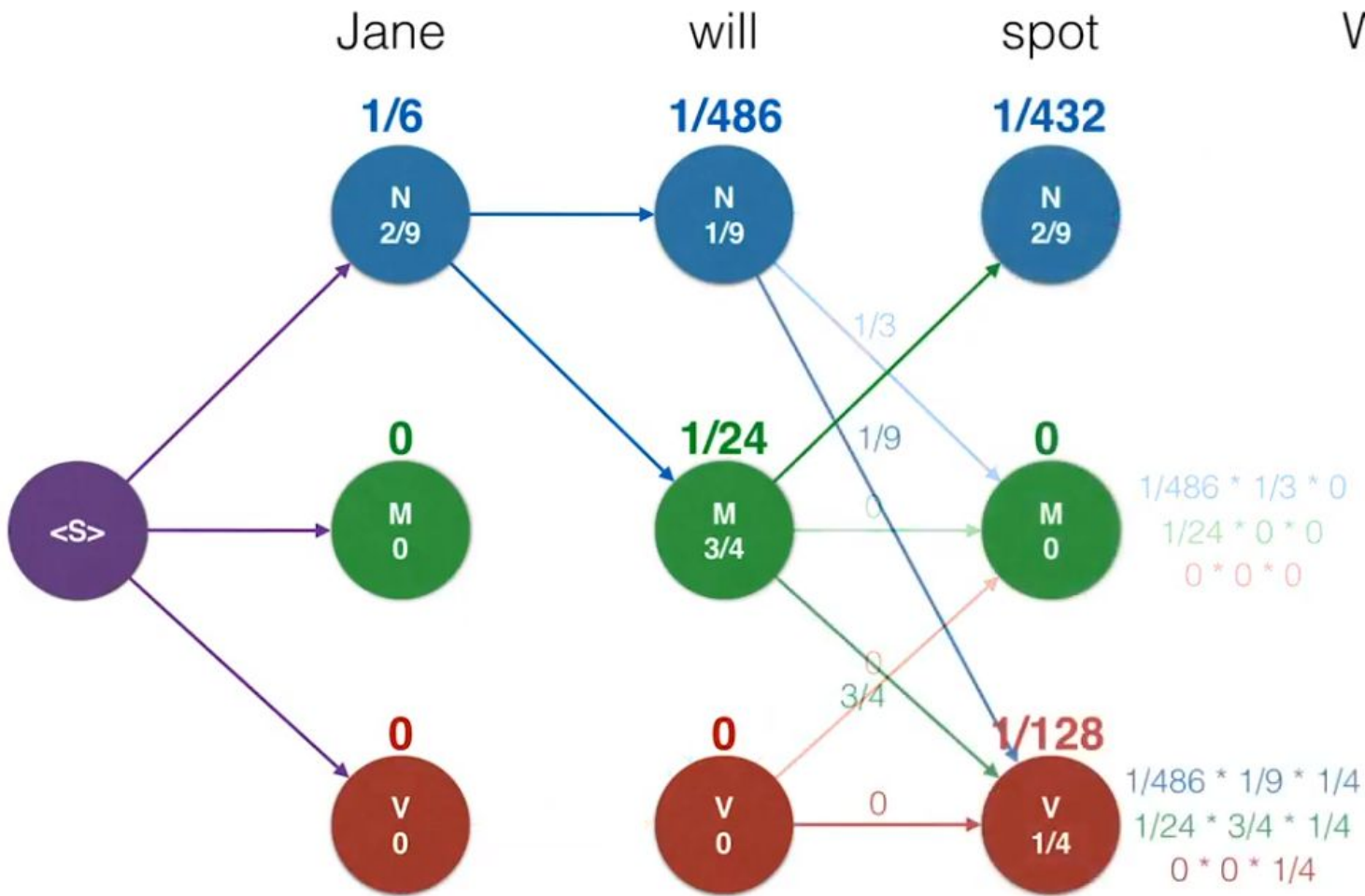
will

spot

Will

.





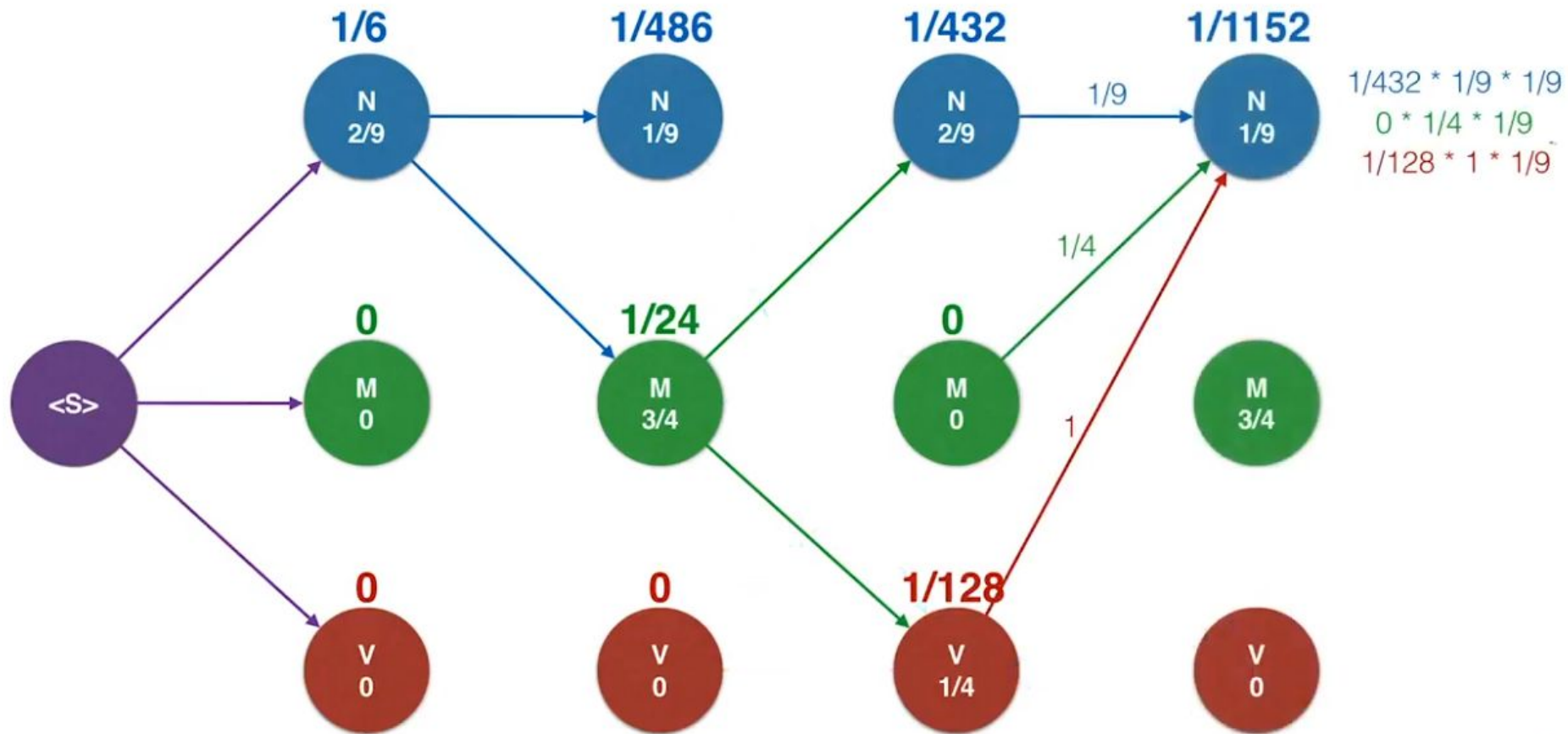
Jane

will

spot

Will

.



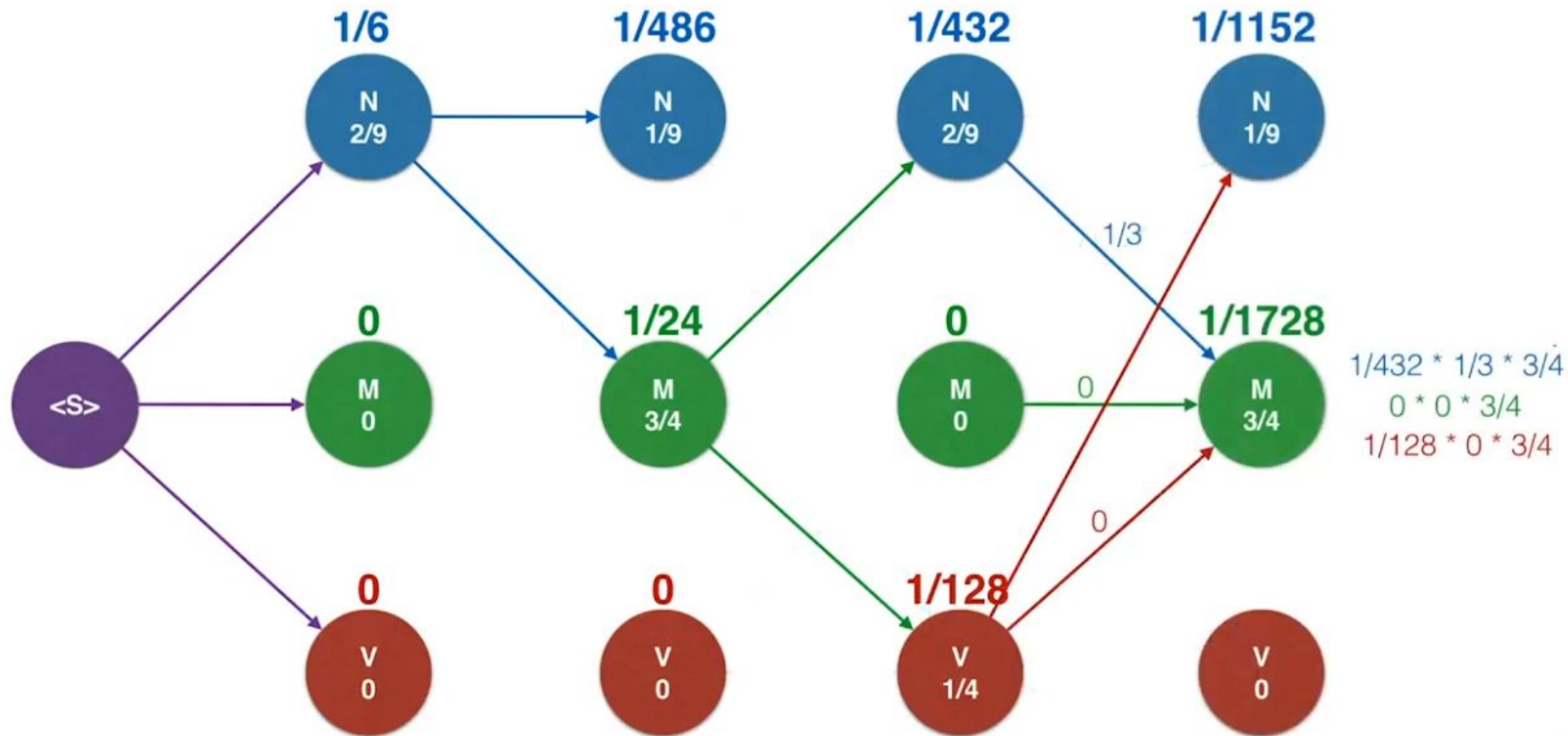


Jane

will

spot

Will



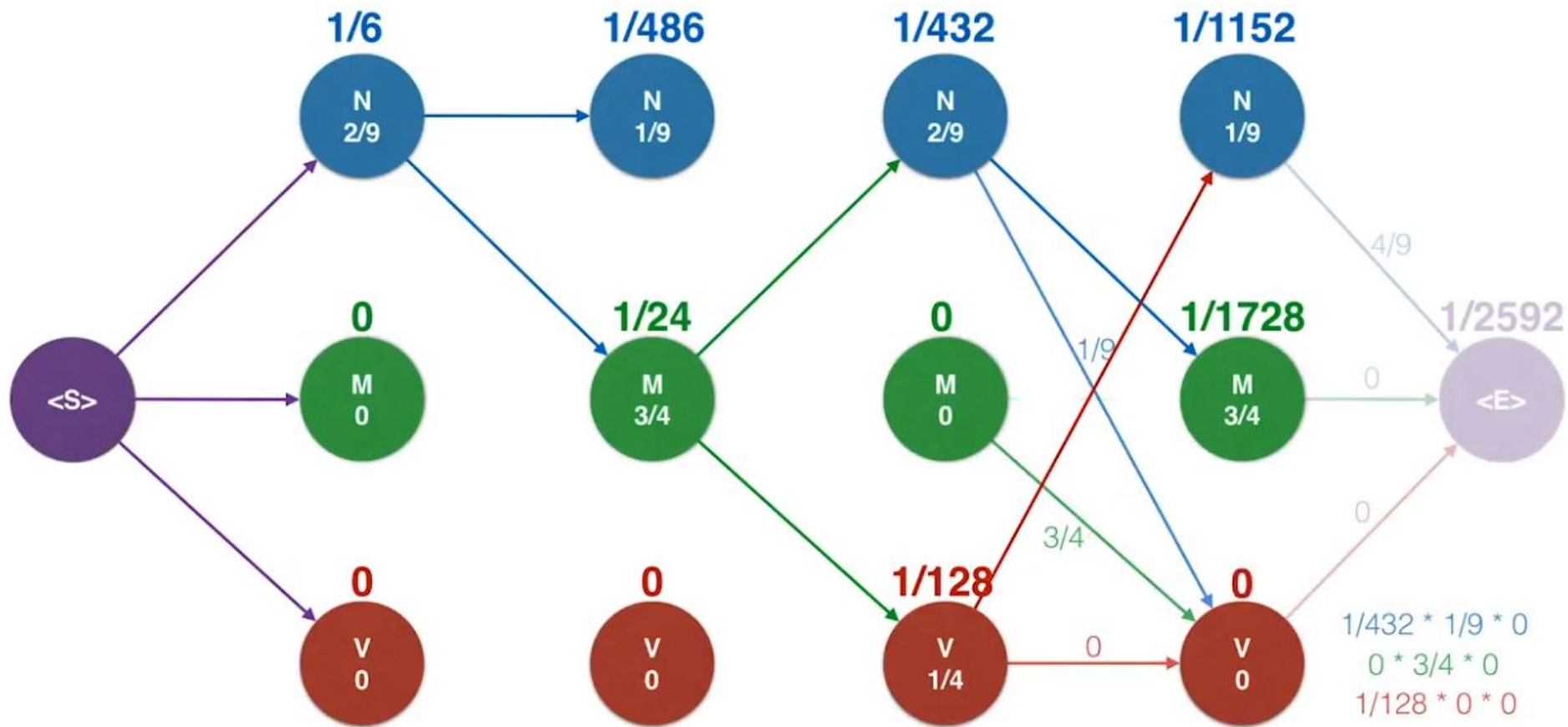
Jane

will

spot

Will

.



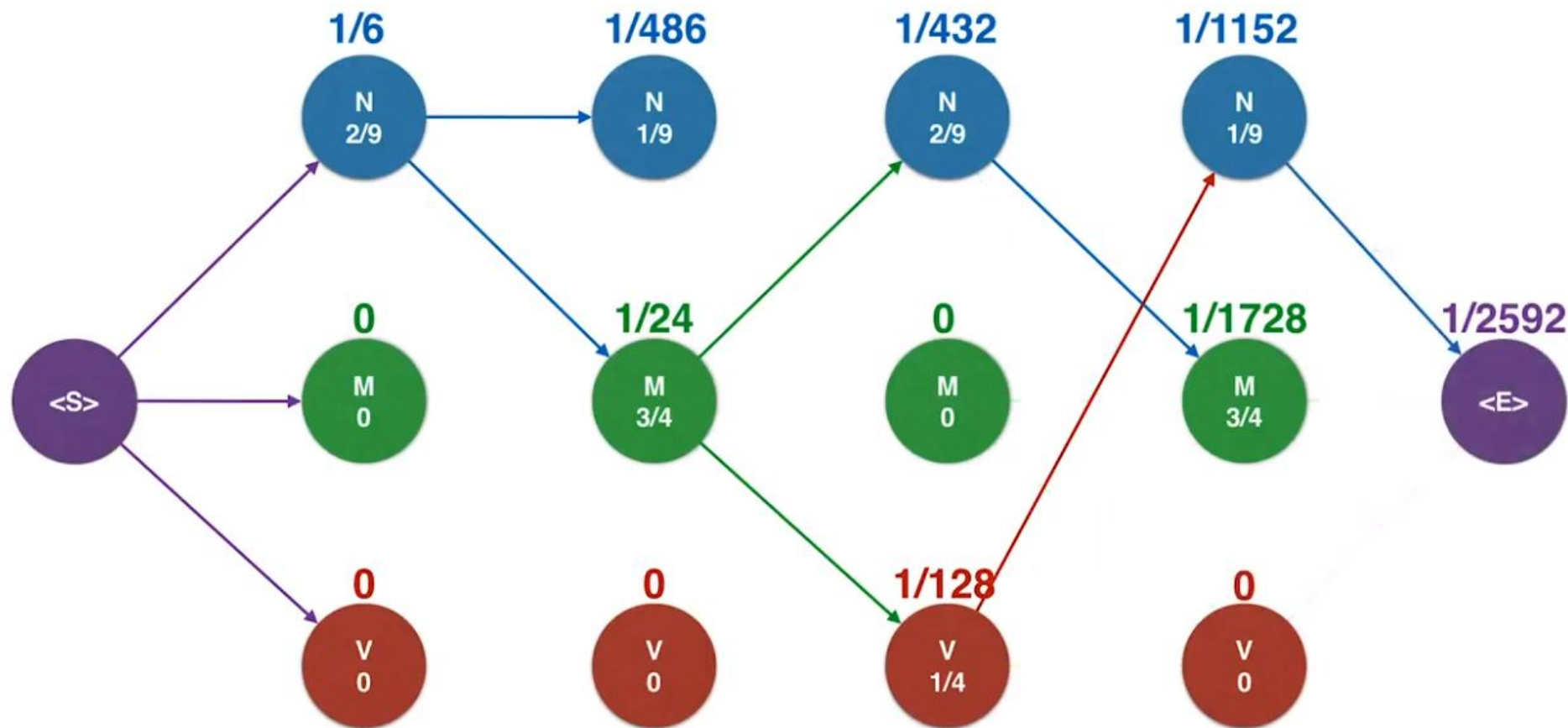
Jane

will

spot

Will

.



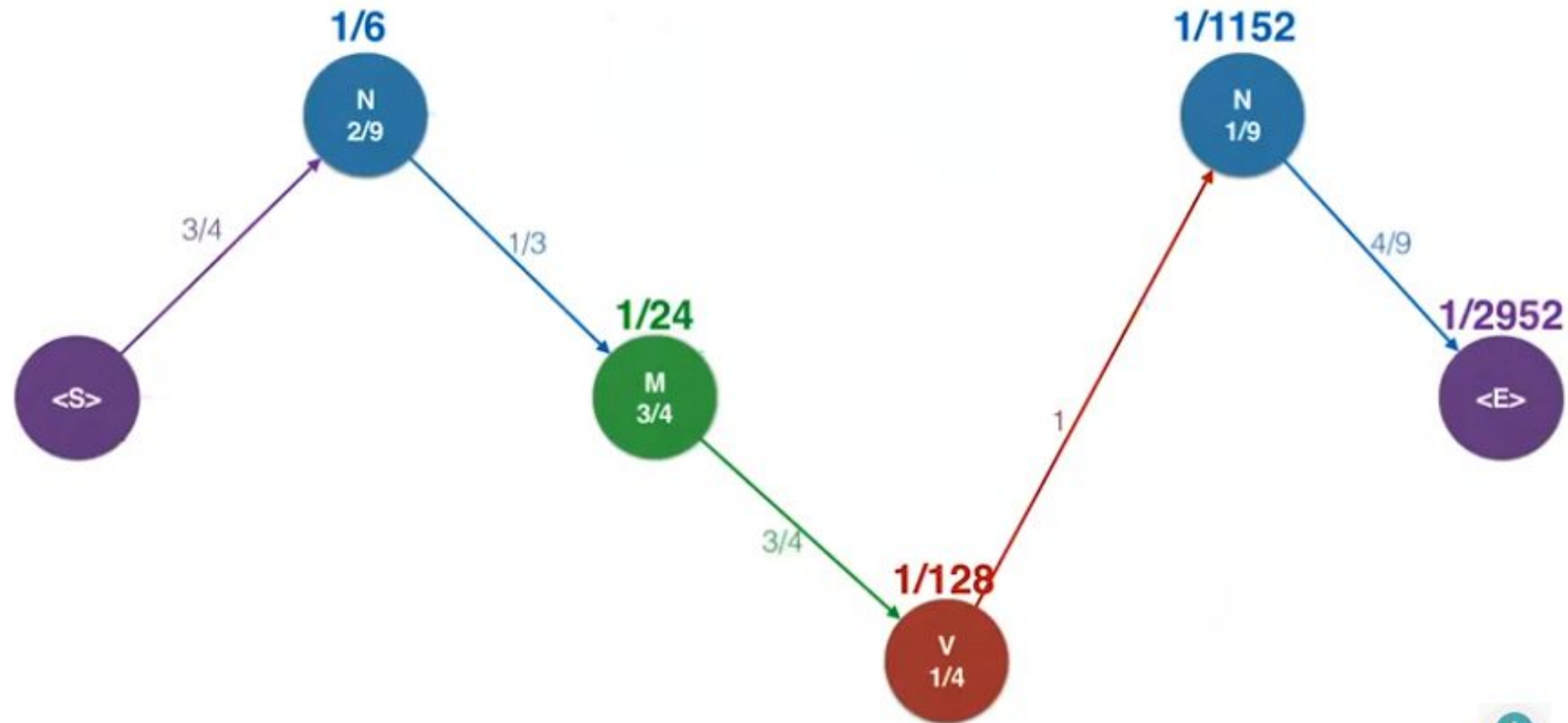
Jane

will

spot

Will

.



# 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:



# 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**  $bestpath$ ,  $bestpathprob$

# 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



# HMM for POS tagging – beam search (k=2)

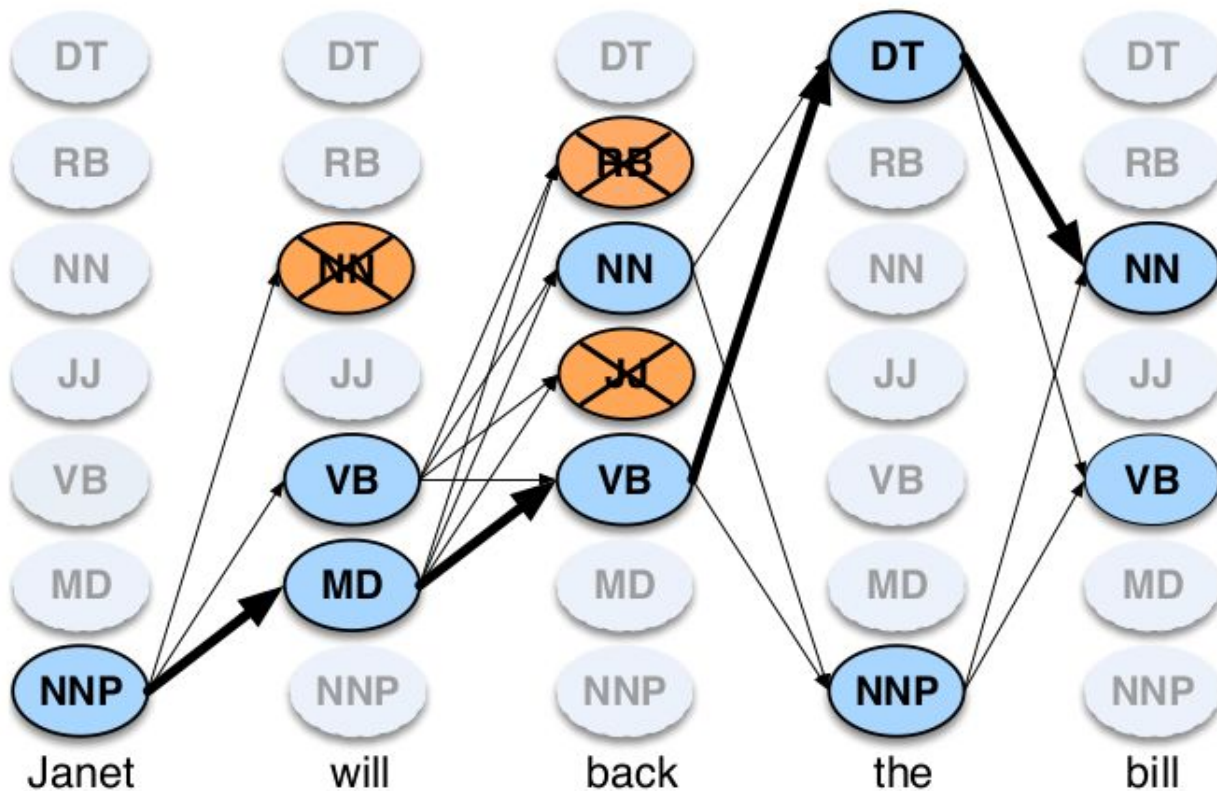


Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

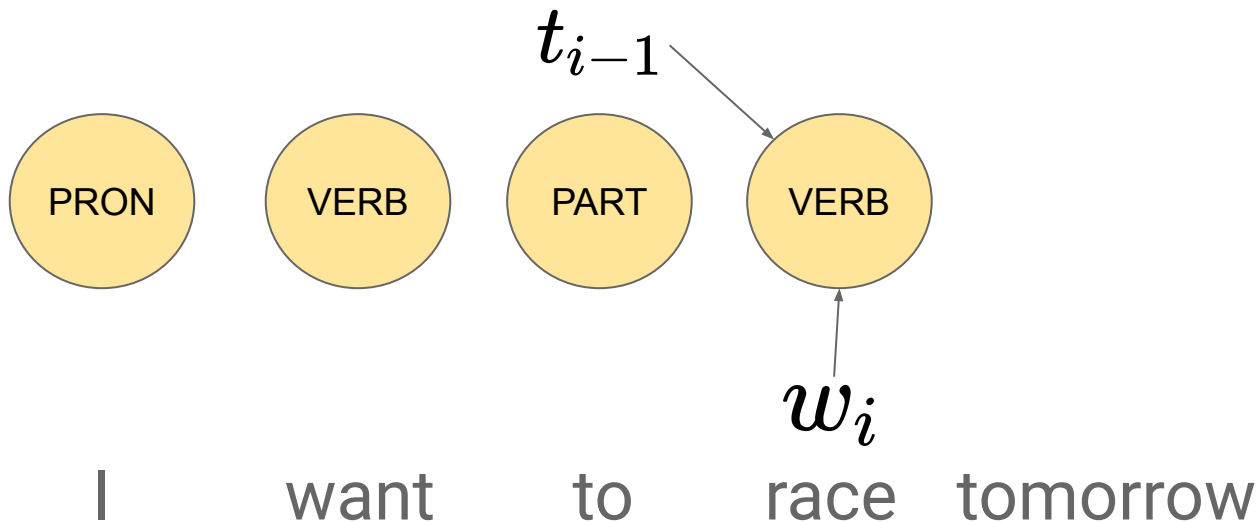


# MEMM for POS tagging

- 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

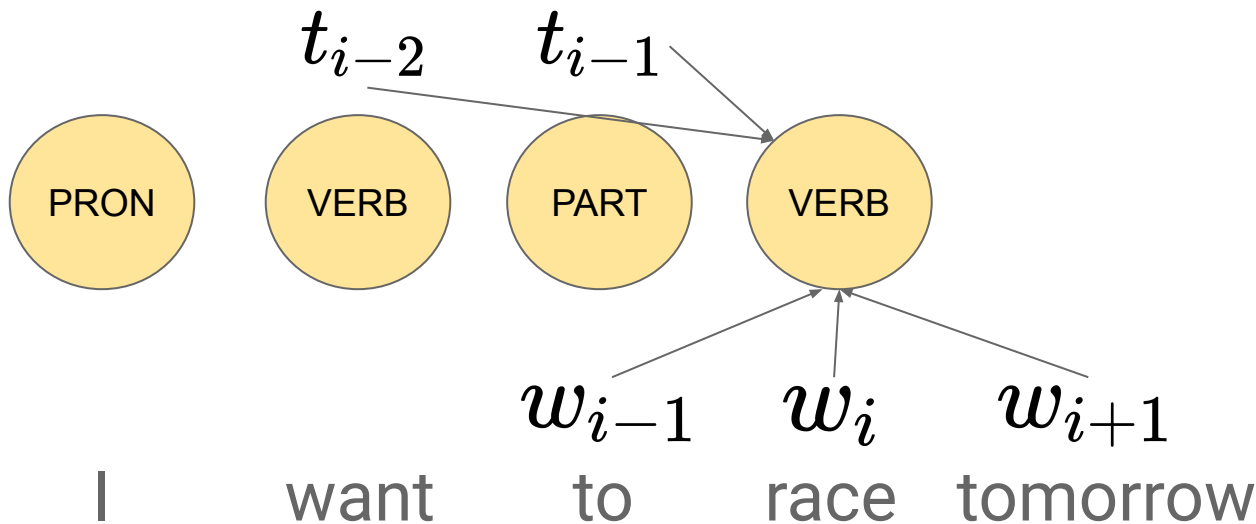
$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T \prod P(t_i | w_i, t_{i-1})\end{aligned}$$

# 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  $\leq 4$ )

$w_i$  contains a particular suffix (from all suffixes of length  $\leq 4$ )

$w_i$  contains a number

$w_i$  contains an upper-case letter

$w_i$  contains a hyphen

$w_i$  is all upper case

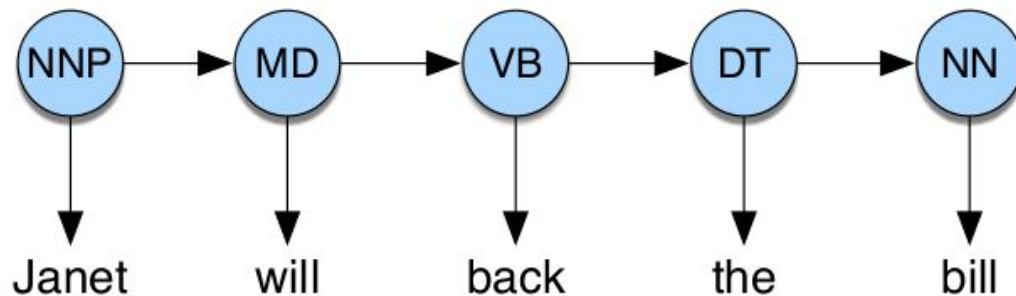
$w_i$ 's word shape

• What else could we use:

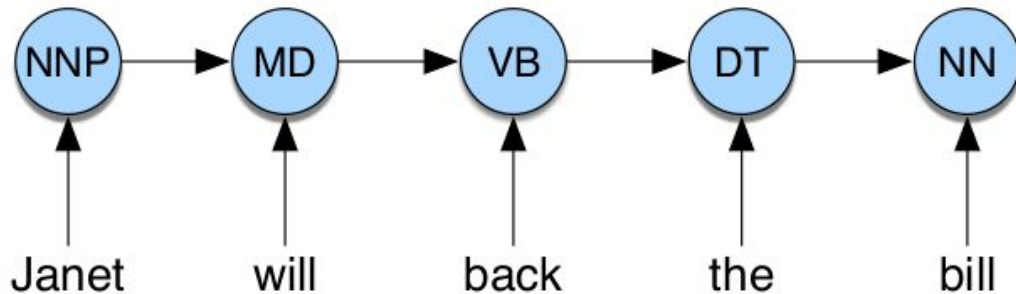
- All sorts of features

# HMM vs MEMM

**HMM**



**MEMM**



# HMM vs MEMM

## MEMM Inference

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

## HMM Inference

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

# HMM vs MEMM

- Versus VITERBI for HMM

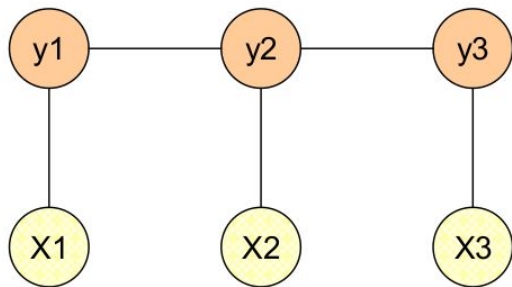
$$v_t(j) = \max_{i=1}^N v_{t-1}(i) P(q_j | q_i) P(o_t | q_j)$$

- Viterbi for MEMM

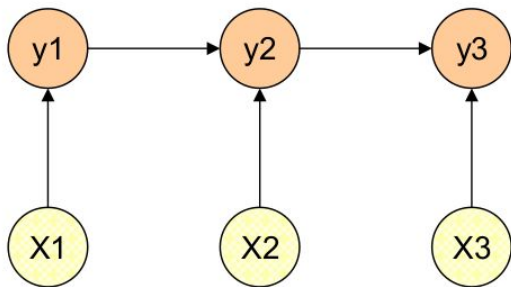
$$v_t(j) = \max_{i=1}^N v_{t-1}(i) P(q_j | q_i, o_t)$$

- $o_t$  could be any feature, not just words

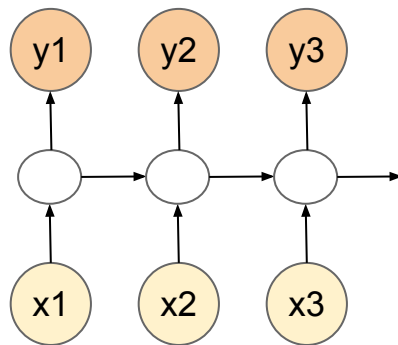
# Other approaches to POS tagging



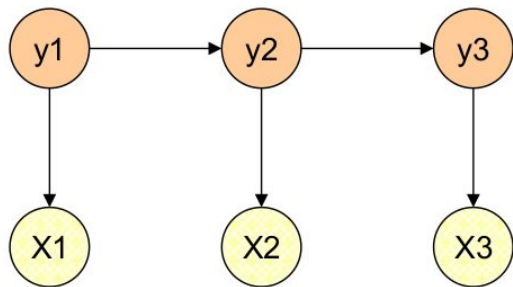
**CRF**



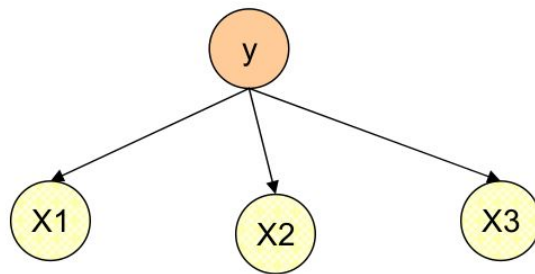
**MEMM**



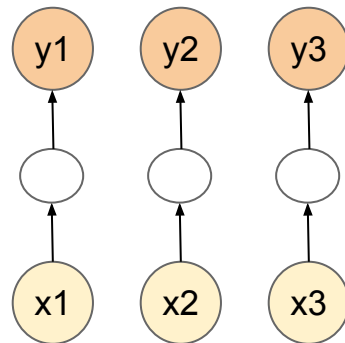
**RNN**



**HMM**



**Naïve Bayes**

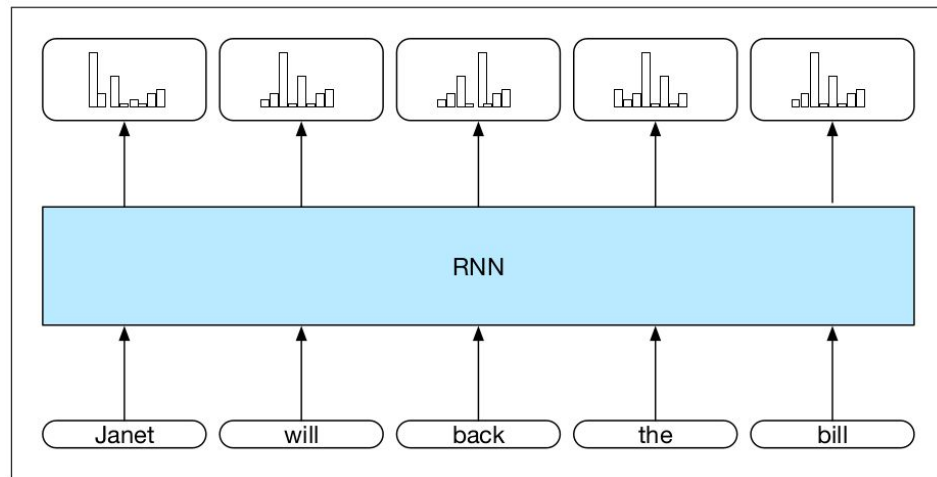


**BERT**



# 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



# RNN for POS tagging

- **Training**

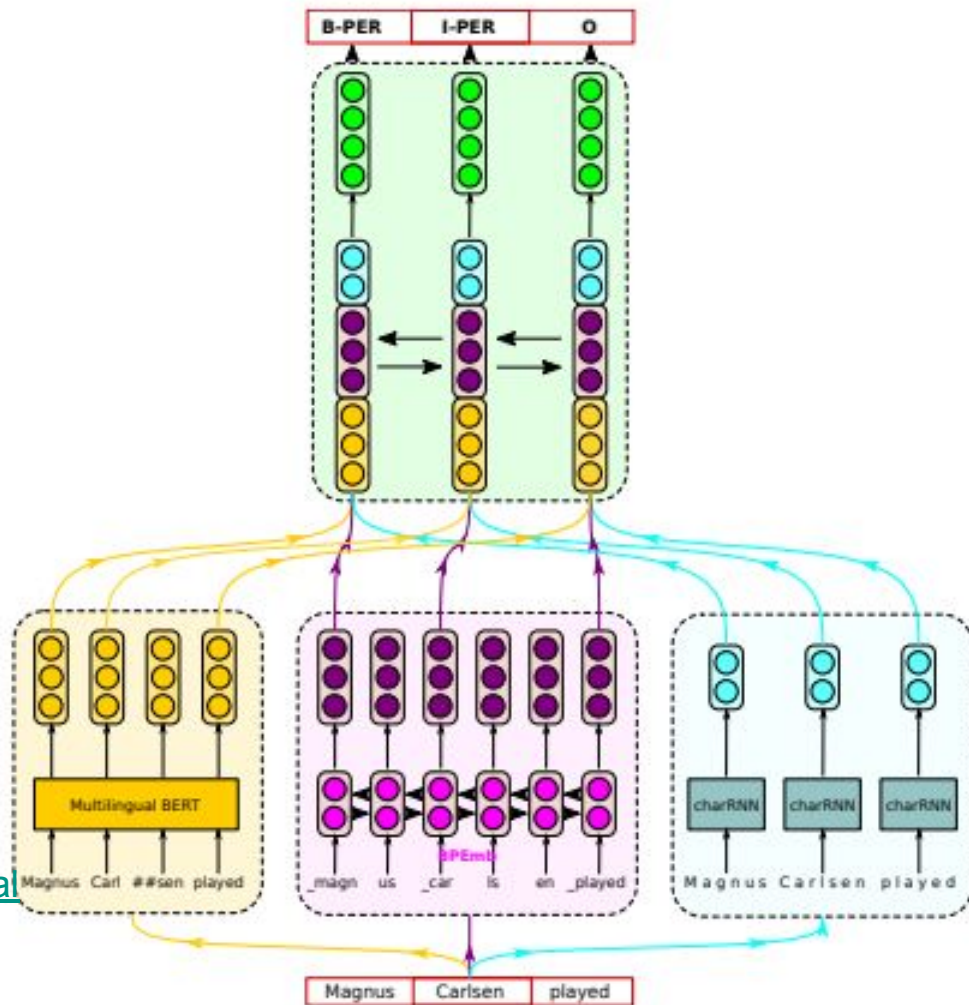
- Cross-entropy loss over the tagset for each word
- Sequence loss is the sum of loss for all words

- **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

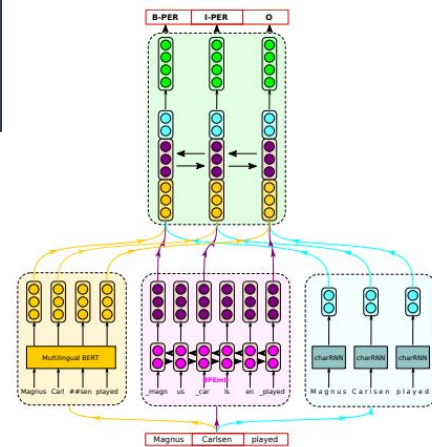


[Sequence Tagging with Contextual and Non-Contextual Subword Representations: A Multilingual Evaluation.](#)

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

Lang.	BiLSTM	Adv.	FastText	BPEmb			BERT		
				BPEmb	+char	+shape	BERT	+char	+char+BPemb
Avg.	96.4	96.6	95.6	95.2	96.4	95.7	95.6	96.3	<b>96.8</b>
bg	98.0	98.5	97.7	97.8	98.5	97.9	98.0	98.5	<b>98.7</b>
cs	98.2	98.8	98.3	98.5	98.9	98.7	98.4	98.8	<b>99.0</b>
da	96.4	96.7	95.3	94.9	96.4	95.9	95.8	96.3	<b>97.2</b>
de	93.4	<b>94.4</b>	90.8	92.7	93.8	93.5	93.7	93.8	<b>94.4</b>
en	95.2	95.8	94.3	94.2	95.5	94.9	95.0	95.5	<b>96.1</b>
es	95.7	96.4	96.3	96.1	96.6	96.0	96.1	96.3	<b>96.8</b>
eu	95.5	94.7	94.6	94.3	<b>96.1</b>	94.8	93.4	95.0	96.0
fa	<b>97.5</b>	<b>97.5</b>	97.1	95.9	97.0	96.0	95.7	96.5	97.3
fi	<b>95.8</b>	95.4	92.8	92.8	94.4	93.5	92.1	93.8	94.3
fr	96.1	<b>96.6</b>	96.0	95.5	96.1	95.8	96.1	96.5	96.5
he	97.0	<b>97.4</b>	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	<b>97.4</b>
hr	<b>96.8</b>	96.3	95.5	93.6	95.4	94.5	96.2	96.6	<b>96.8</b>
id	93.4	<b>94.0</b>	91.9	90.7	93.4	93.0	92.2	93.0	93.5
it	98.0	<b>98.1</b>	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	<b>98.5</b>
pl	97.6	97.6	96.2	95.8	97.1	96.1	96.5	<b>97.7</b>	97.6
pt	97.9	98.1	97.3	96.3	97.7	97.2	97.5	97.8	98.1
sl	96.8	<b>98.1</b>	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	<b>97.4</b>

# SOTA for POS tagging – low res langs.

Lang.	Adv.	FastText	BPEmb +char	MultiBPEmb +char+finetune
Avg.	91.6	90.4	79.3	<b>92.4</b>
el	<b>98.2</b>	97.2	96.5	97.9
et	91.3	89.5	82.1	<b>92.8</b>
ga	<b>91.1</b>	89.2	81.6	91.0
hu	<b>94.0</b>	92.9	83.1	<b>94.0</b>
ro	<b>91.5</b>	88.6	73.9	89.7
ta	83.2	85.2	58.7	<b>88.7</b>