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
- Founder of NLP AI startup Semafind

About

Who researchers neuro-symbolic reasoning at Imperial College London?

Searching for: **neuro-symbolic reasoning Imperial College London**

Generating answers for you…

According to the web search results, some of the researchers who work on neuro-symbolic reasoning at Imperial College London are:

- **Nuri Cingillioglu** [1], who is a guest lecturer at the Department of Computing and works on neuro-symbolic systems that combine low-level visual input with high-level symbolic reasoning [1] [2].
- **Alessandra Russo** [3], who is a professor of applied computational logic and works on logic-based machine learning and neuro-symbolic AI [2] [3].
- **Lukas Dickens** [3], who is a research associate and works on formalizing coherence and consistency applied to transfer learning in neuro-symbolic autoencoders [3].

I hope this answers your question. 😊

Learn more: 1. ceur-ws.org 2. proceedings.kr.org 3. imperial.ac.uk

[1]: https://ceur-ws.org/Vol-2986/paper3.pdf "pix2rule: End-to-end Neuro-symbolic Rule Learning"
[2]: https://proceedings.kr.org/2022/44/kr2022-0044-aspis-et-al.pdf "Embed2Sym - Scalable Neuro-Symbolic Reasoning via Clustered Embeddings - KR"
[3]: https://sites.google.com/view/nesy20/home "NeSy'20/21 @ IJCLR - Google"
[4]: https://www.imperial.ac.uk/people/a.russo "Home - Professor Alessandra Russo - Imperial College London"
[5]: https://www.doc.ic.ac.uk/~nuric/pages/about.html "About | nuric"
[6]: https://www.imperial.ac.uk/engineering/departments/computing/current-students/courses/50009/ "50009 | Faculty of Engineering | Imperial College London"

Learn more: 1. ceur-ws.org 2. proceedings.kr.org 3. imperial.ac.uk 4. ceur-ws.org +5 more

POS tagging

Introduction

I saw the boy on the hill with a telescope.

I	saw	the	boy	on	the	hill	with	a	telesc ope	.
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** l
  - Pre-processing can be based on POS
    - E.g. select **adjectives** for sentiment analysis
  - For neural syntactic and **semantic parsing**

- POS tagging libraries: 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 $W = w_1, w_2, \ldots, w_n$
  - Estimate the **sequence** of POS tags $T = t_1, t_2, \ldots, t_n$
  - 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):

$$P(T) \approx P(t_1)P(t_2|t_1)P(t_3|t_2)\ldots P(t_n|t_{n-1})$$

- Each word depends only on its tag (not on previous words)

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

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

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

t_0

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

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

	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

t_0

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

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

	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

$$P(\boxed{w_i}|\boxed{t_i})$$

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

	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

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(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

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 $P(T|W)$

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

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:

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

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

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

Emission probabilities

Transition probabilities

Probabilistic POS tagging

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

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

t_0

	PROPN	VERB	PART	NOUN	PRON	DET
<s> (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

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

	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	[illegible]	[illegible]	[illegible]	[illegible]	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 **λ** and a sequence of observations $O = o_1 o_2 \dots o_T$ (test case), find the most probable sequence of states $Q = q_1 q_2 \dots q_T$

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

$$\approx argmax \prod_{i=1}^{N} \underbrace{P(w_i|t_i)}_{\text{emission}} \underbrace{P(t_i|t_{i-1})}_{\text{transition}}$$

HMM for POS tagging

- **Viterbi**

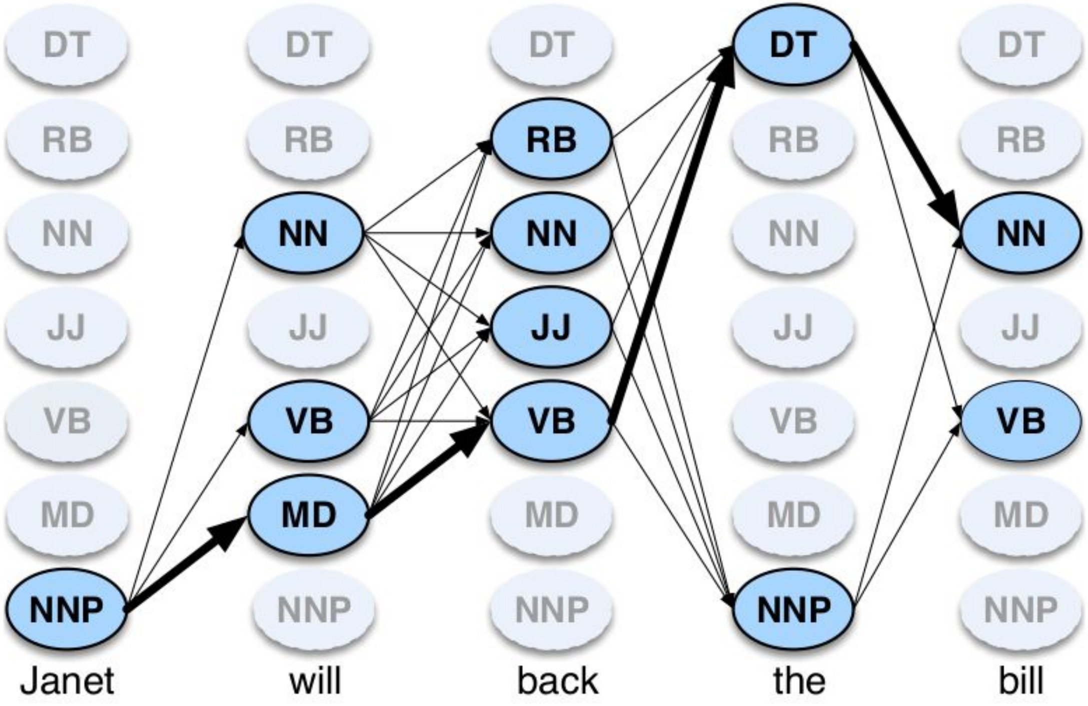


Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

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) $q\_1.....q\_t{-}1$

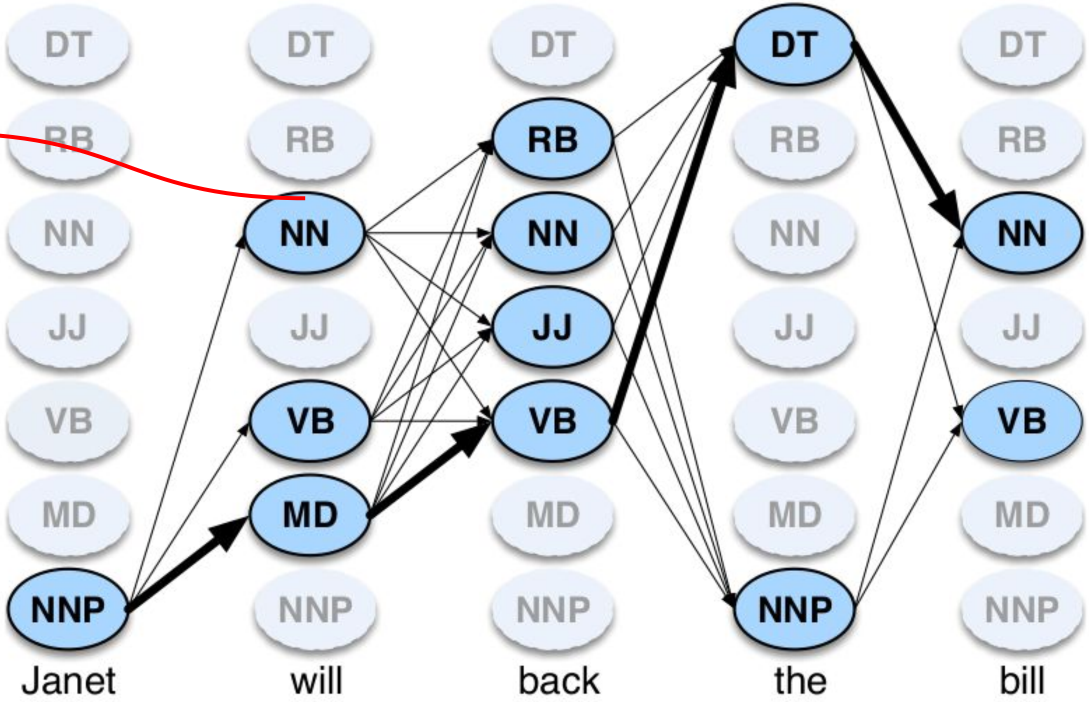


Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

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 $q_1, \ldots, q_{t-1}$

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:

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

  - 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 **previous Viterbi path** probability from the previous time step
a_{ij}	the **transition** probability from previous state q_i to current state q_j
$b_j(o_t)$	the **emission** 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 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

HMM for POS tagging

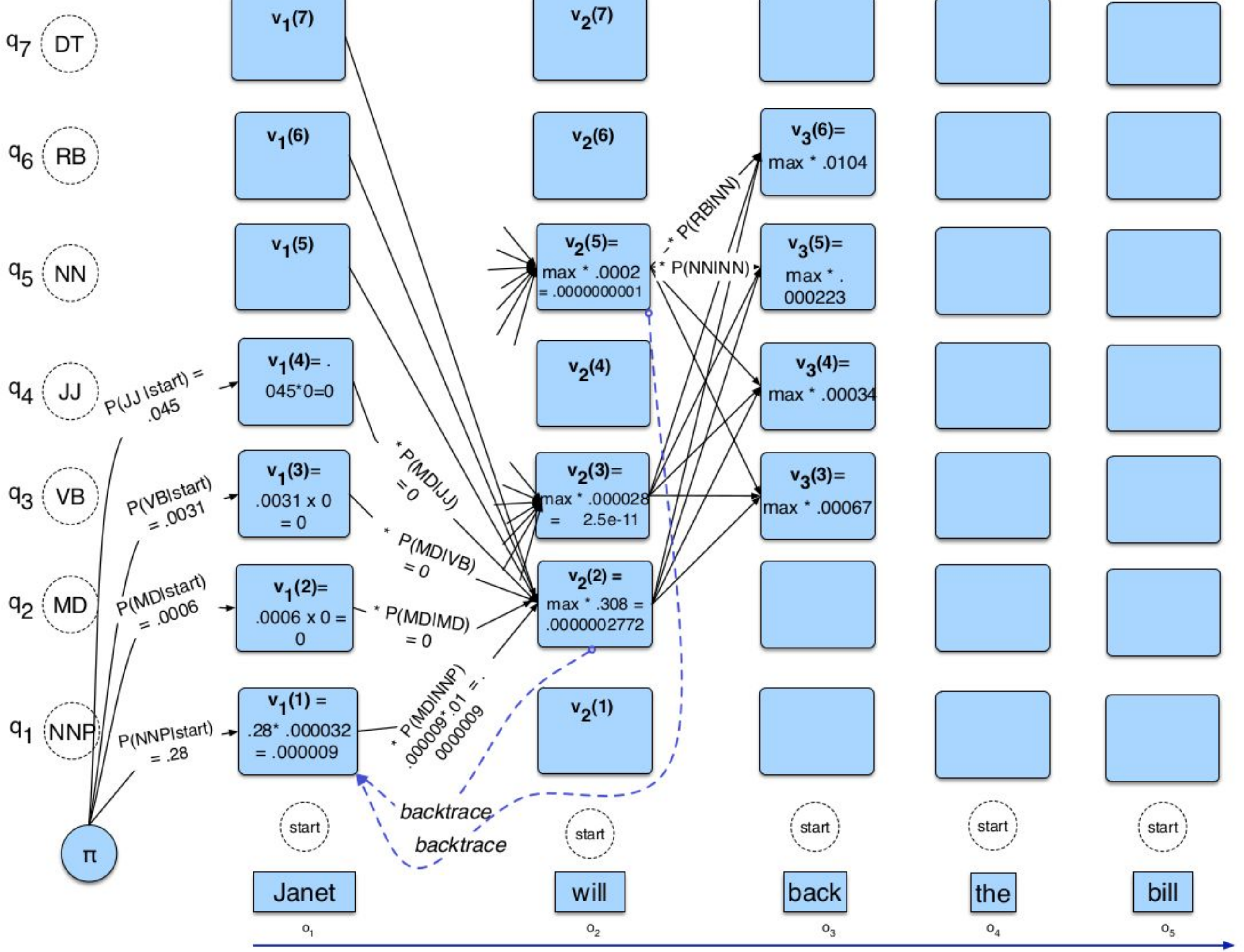
Counts should be smoothed

- 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

HMM



q7 DT
q6 RB
q5 NN
q4 JJ
q3 VB
q2 MD
q1 NNP
π
P(JJ |start) = .045
P(VB|start) = .0031
P(MD|start) = .0006
P(NNP|start) = .28
v1(7)
v1(6)
v1(5)
v1(4)= . 045*0=0
v1(3)= .0031 x 0 = 0
v1(2)= .0006 x 0 = 0
v1(1) = .28* .000032 = .000009
* P(MD|JJ) = 0
* P(MD|VB) = 0
* P(MD|MD) = 0
* P(MD|NNP) .000009*.01 = .0000009
v2(7)
v2(6)
v2(5)= max * .0002 = .0000000001
v2(4)
v2(3)= max * .000028 = 2.5e-11
v2(2) = max * .308 = .0000002772
v2(1)
* P(RB|NN)
* P(NN|NN)
v3(6)= max * .0104
v3(5)= max * . 000223
v3(4)= max * .00034
v3(3)= max * .00067
backtrace
backtrace
start
Janet
will
back
the
bill
o1
o2
o3
o4
o5

HMM

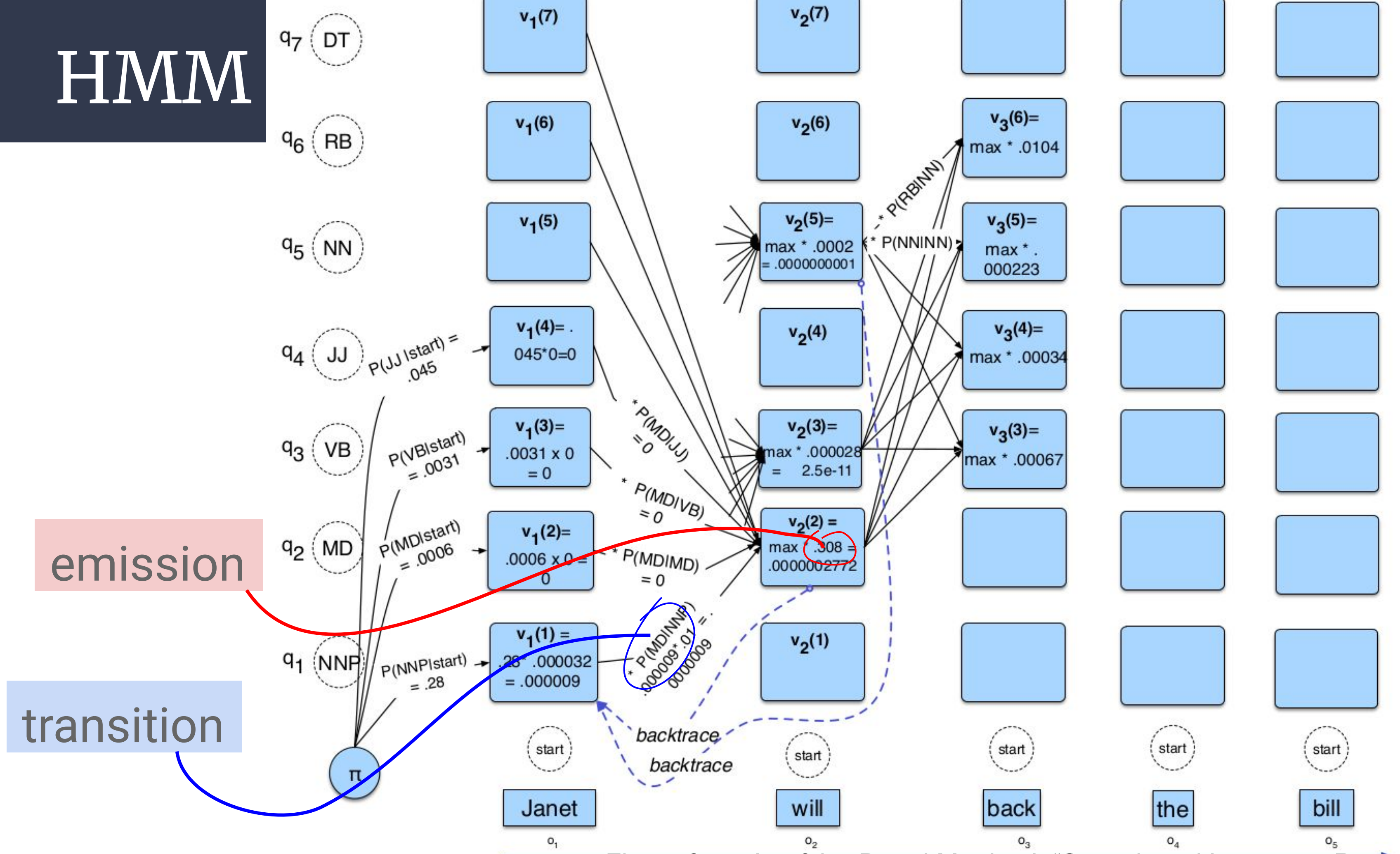


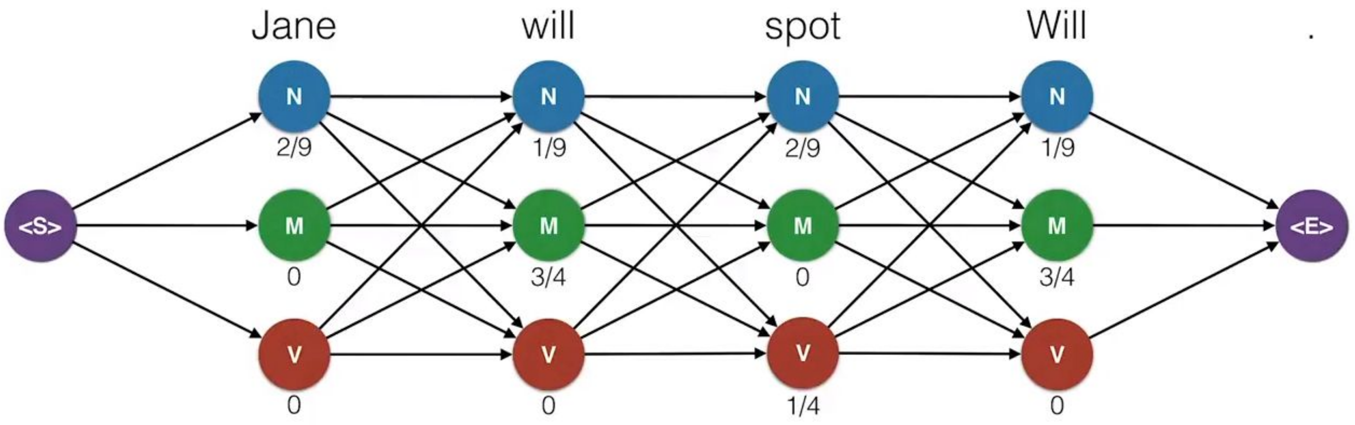
Figure from Jurafsky, D and Martin, J, "Speech and Language Processing," 2018

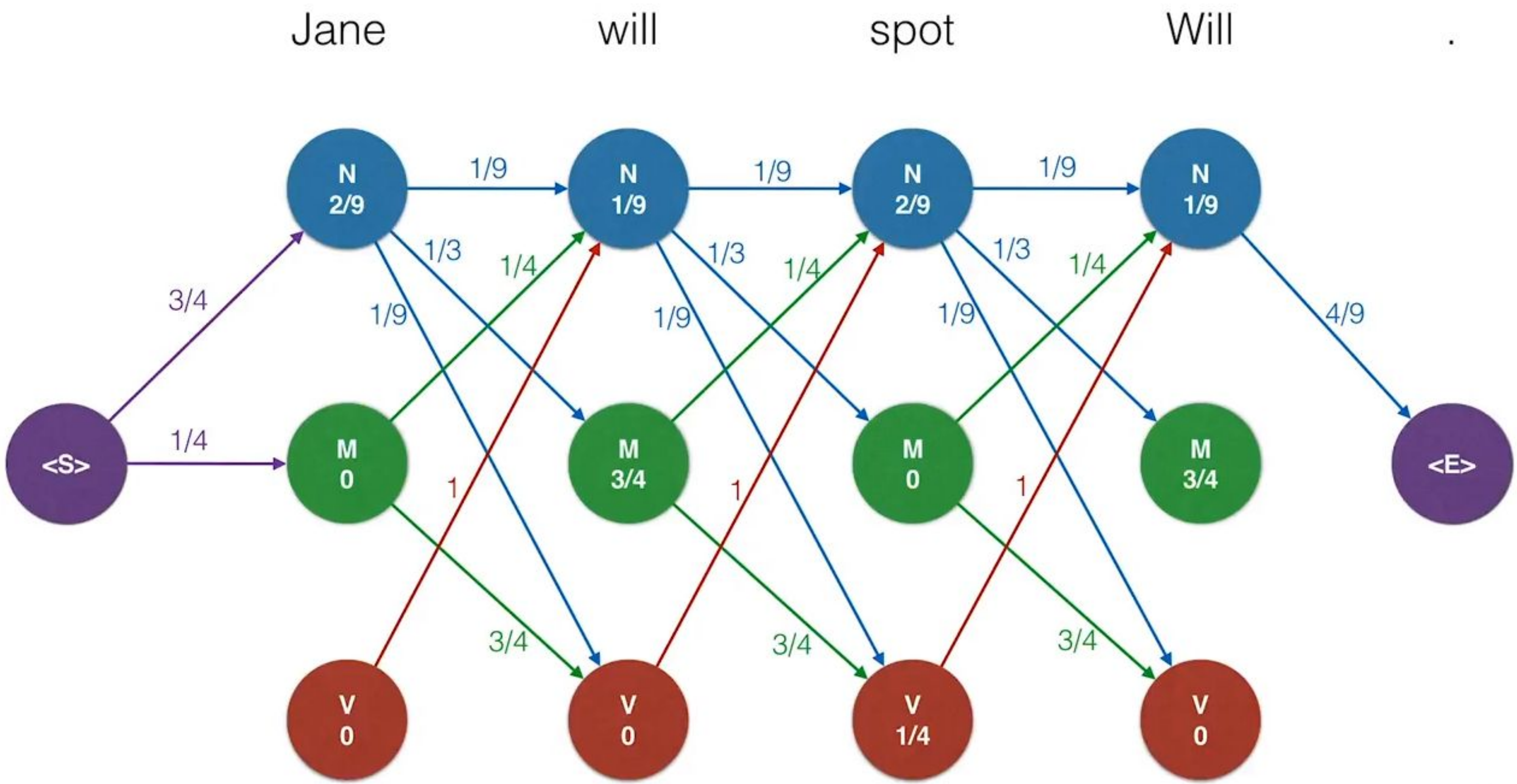
HMM for POS tagging

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

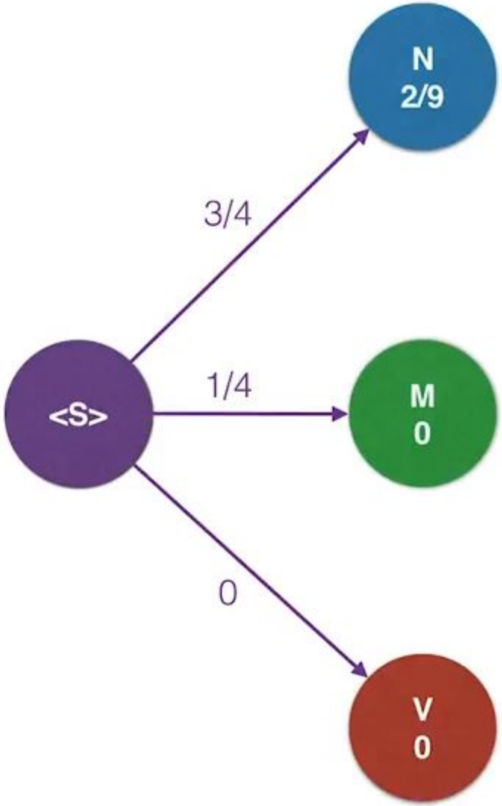
	N	M	V	<E>
<S>	3/4	1/4	0	0
N	1/9	1/3	1/9	4/9
M	1/4	0	3/4	0
V	1	0	0	0



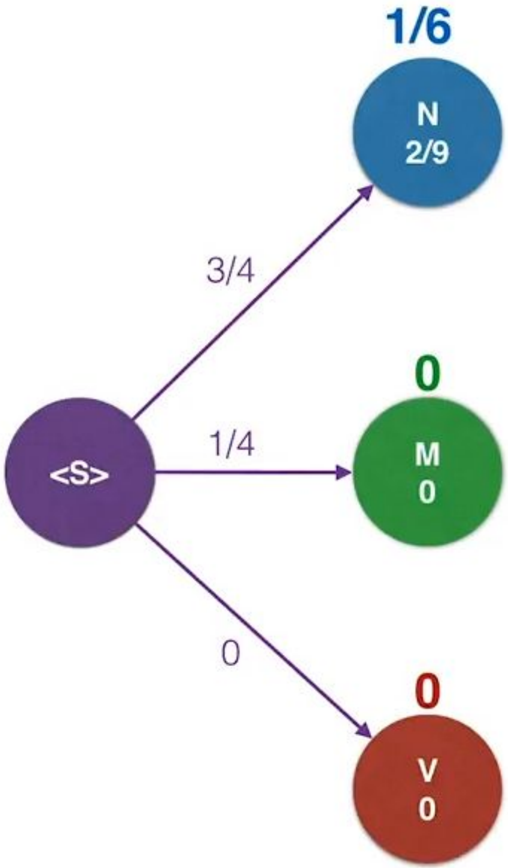


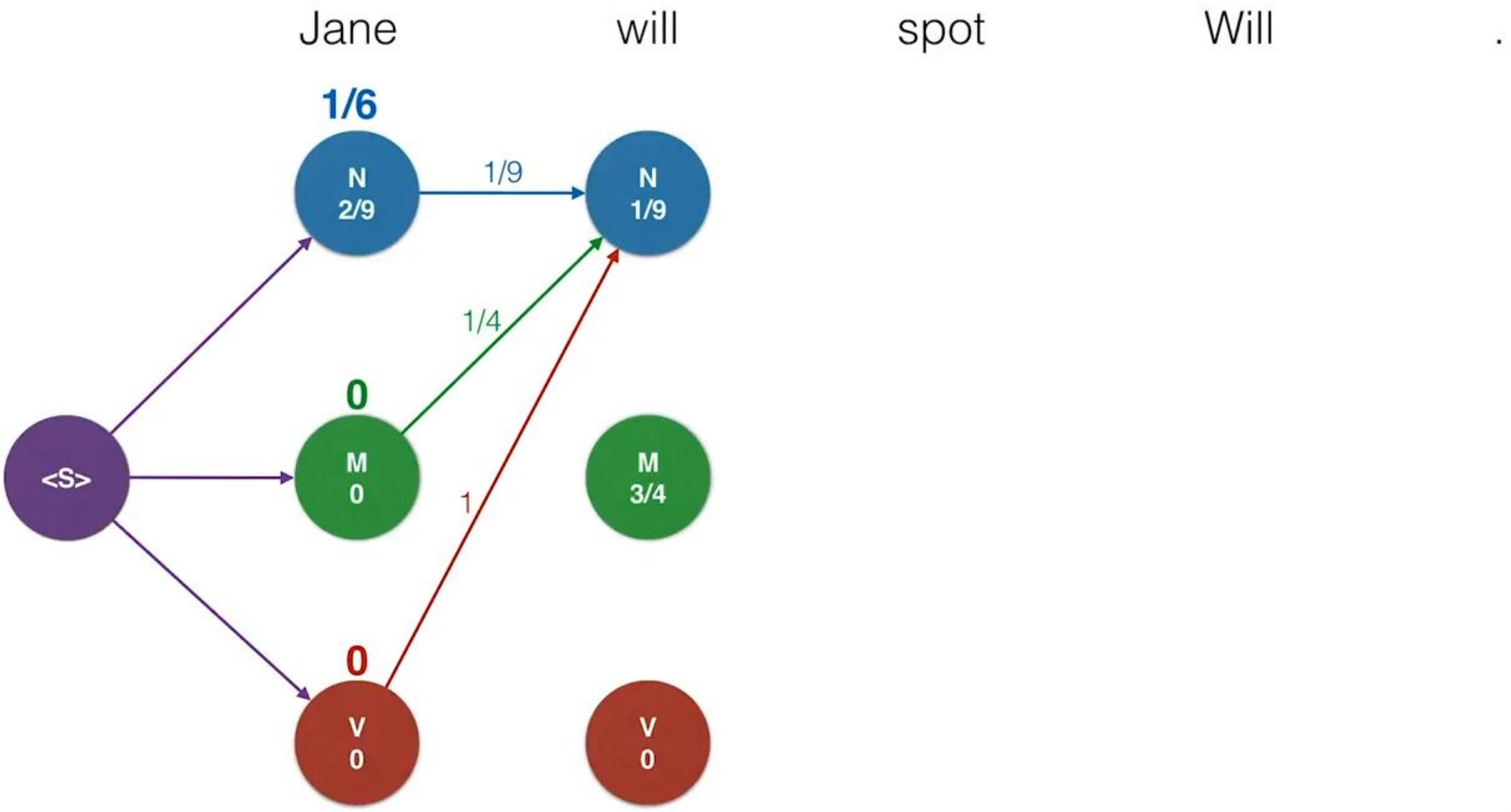
Jane
will
spot
Will
.
<S>
<E>
N 2/9
M 0
V 0
N 1/9
M 3/4
V 0
N 2/9
M 0
V 1/4
N 1/9
M 3/4
V 0
3/4
1/4
1/9
1/3
1/9
1/4
1
3/4
1/9
1/3
1/9
1/4
1
3/4
1/9
1/3
1/9
1/4
1
3/4
4/9

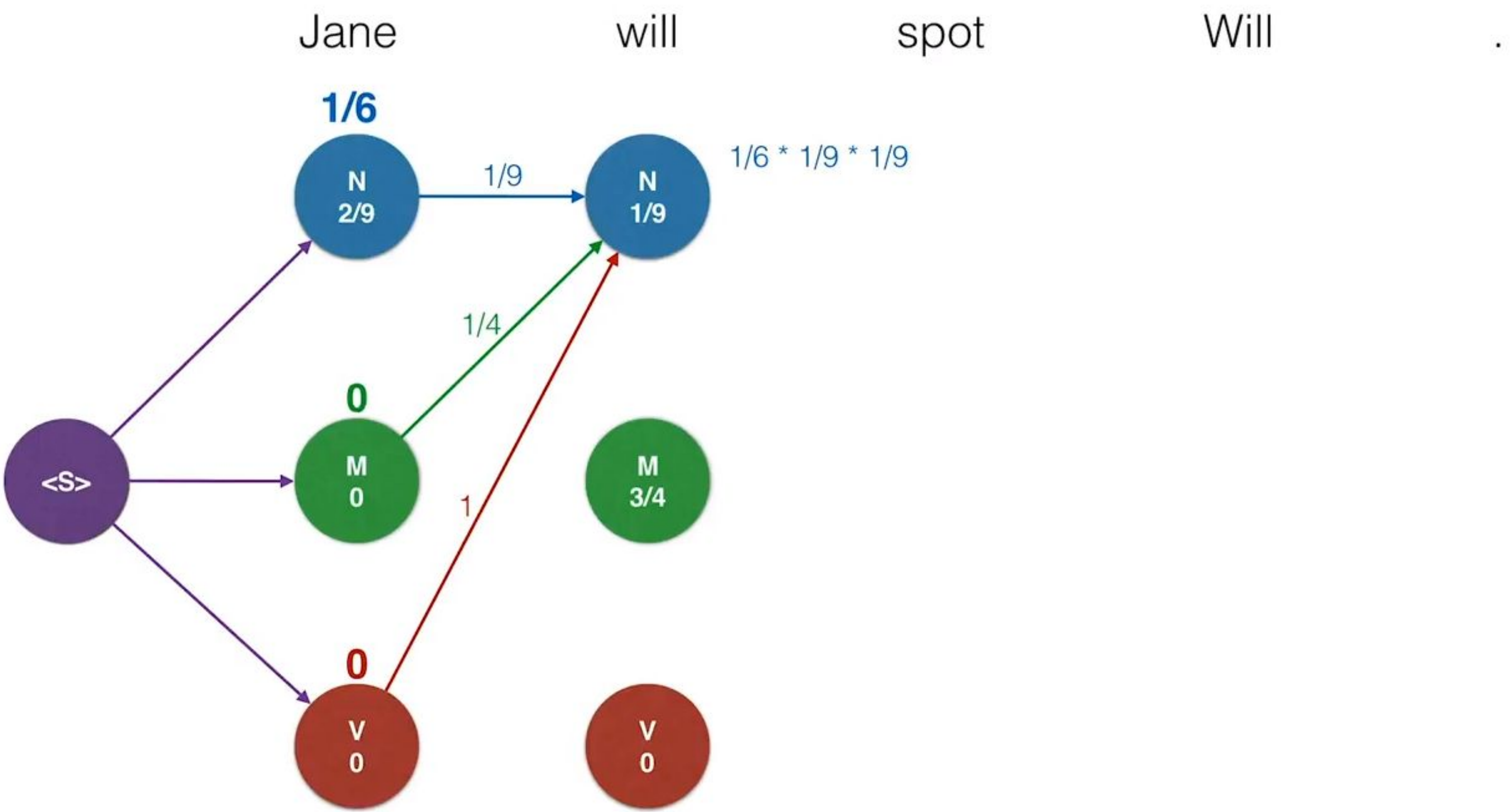
Jane will spot Will .

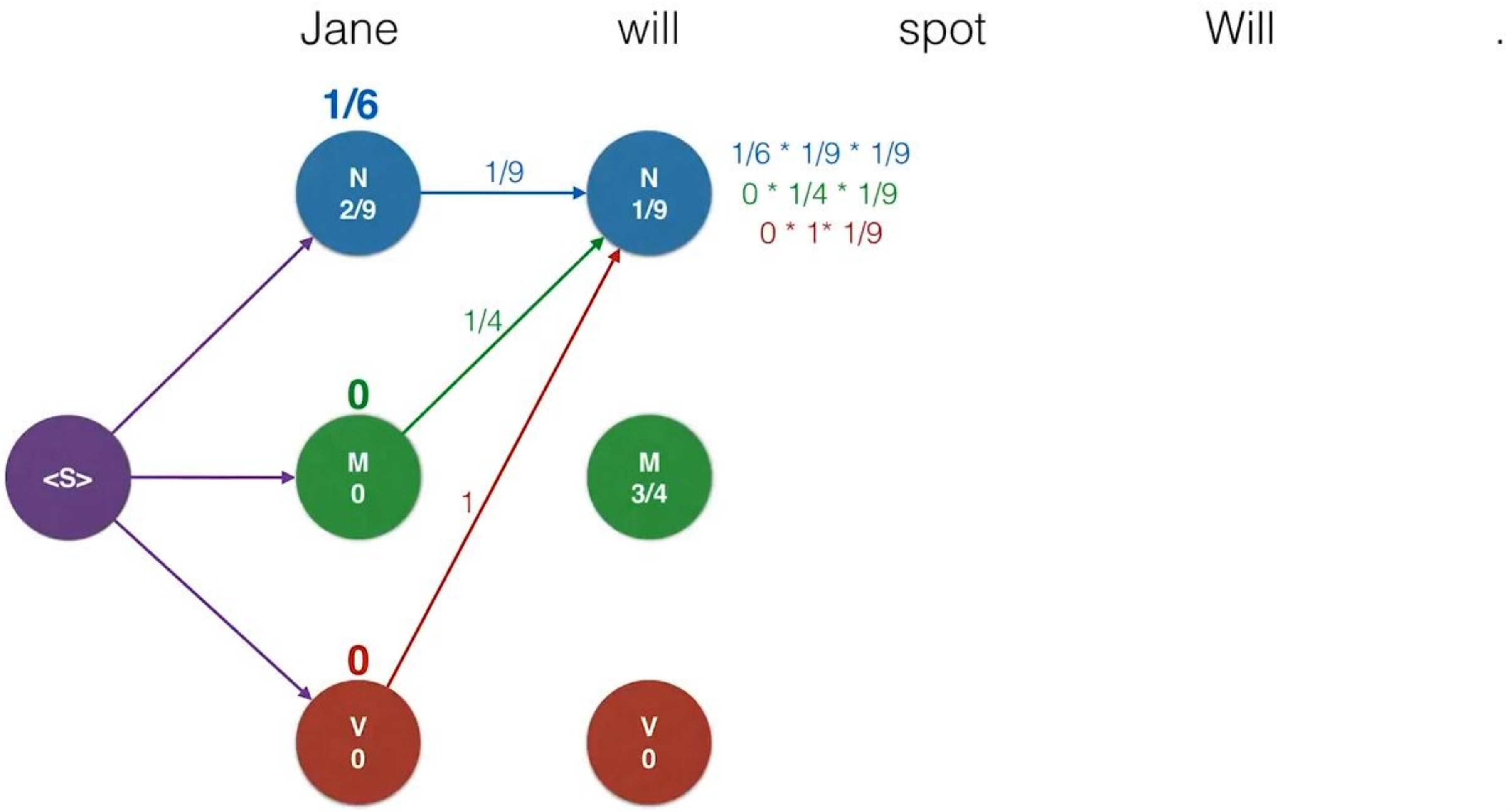


Jane will spot Will .

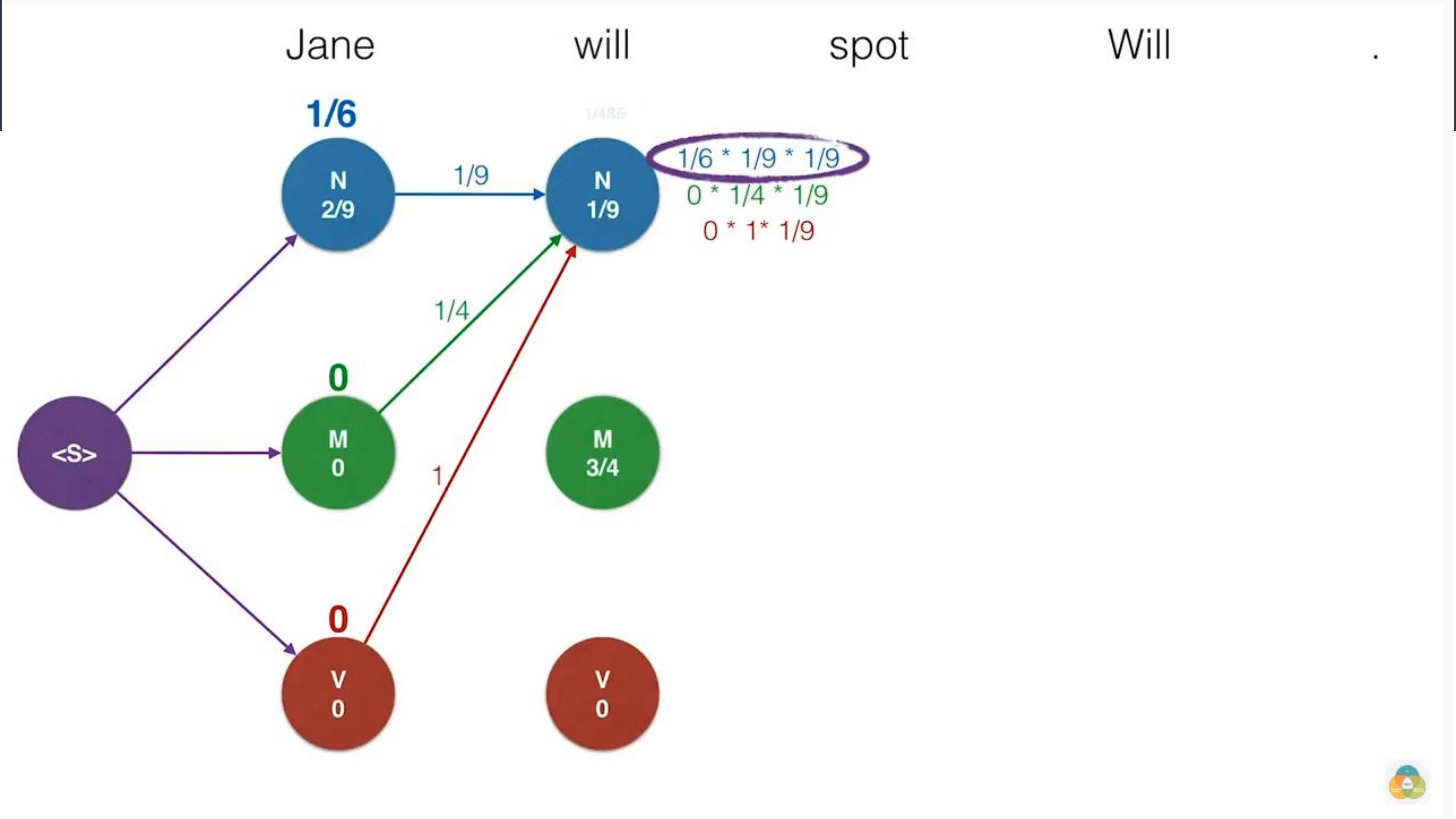


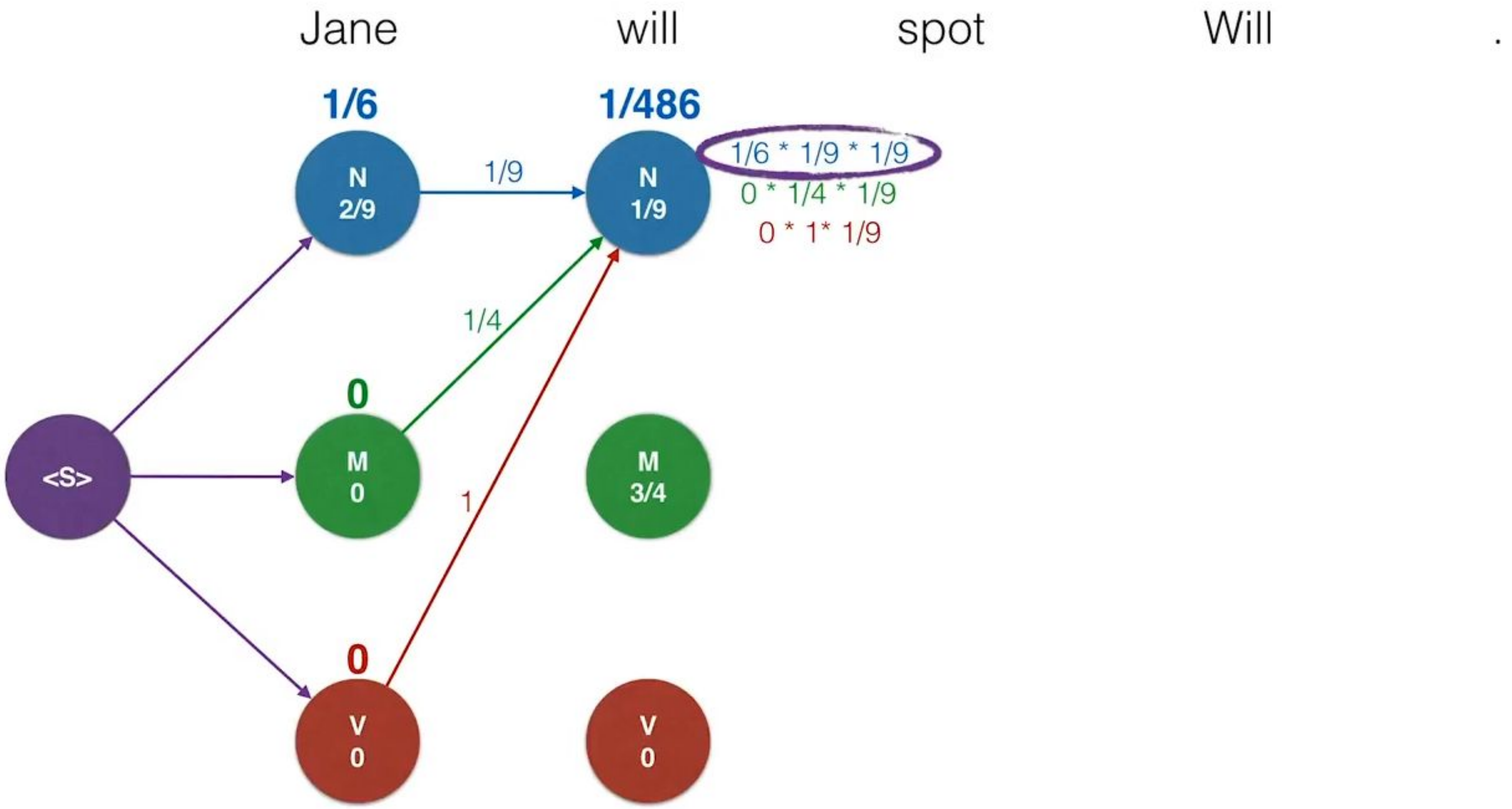

Jane
will
spot
Will
.
1/6
N
2/9
1/9
N
1/9
1/4
0
<S>
M
0
1
M
3/4
0
V
0
V
0


Jane
will
spot
Will
.
1/6
N
2/9
1/9
N
1/9
1/6 * 1/9 * 1/9
1/4
0
<S>
M
0
1
M
3/4
0
V
0
V
0

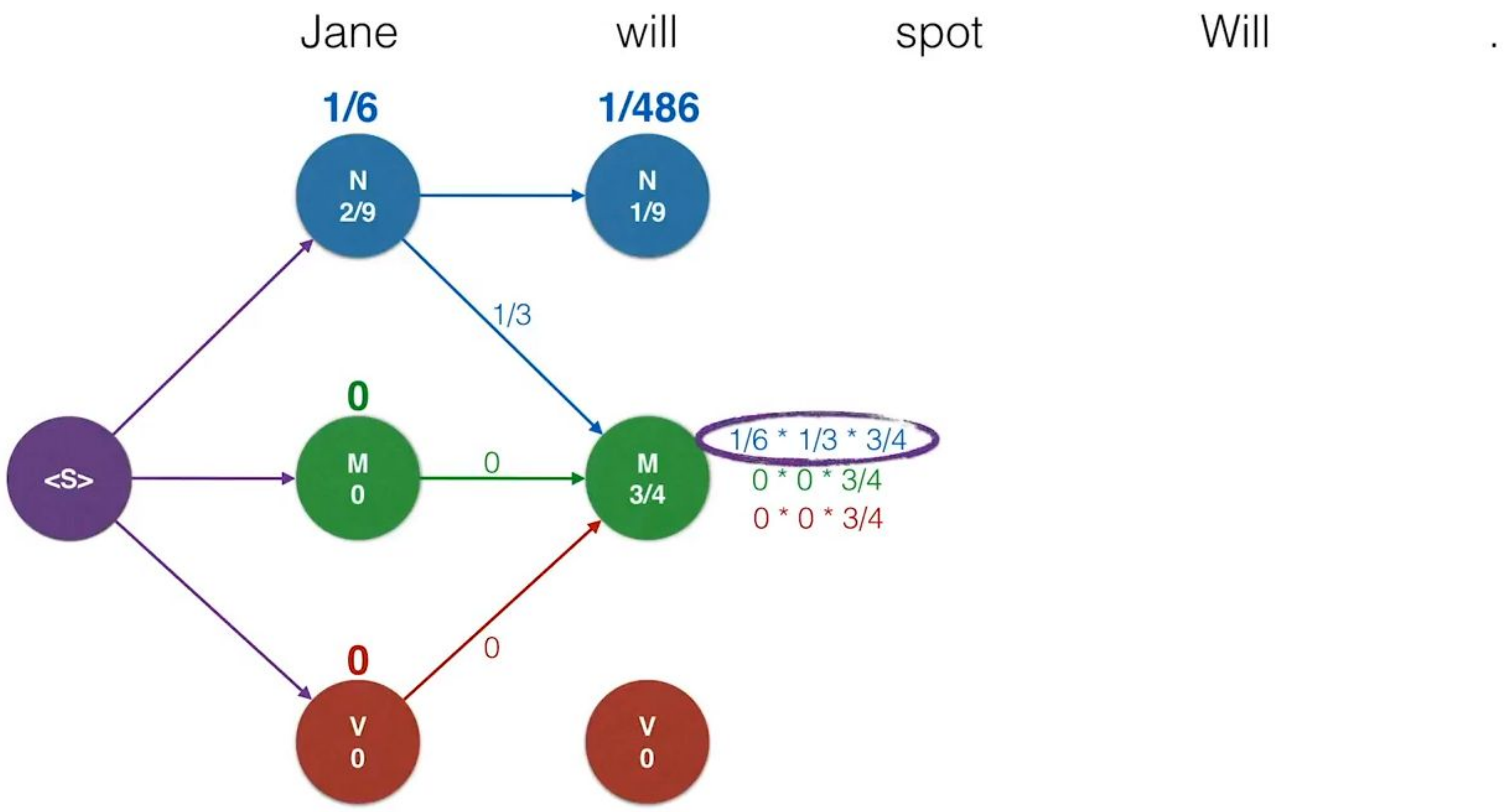


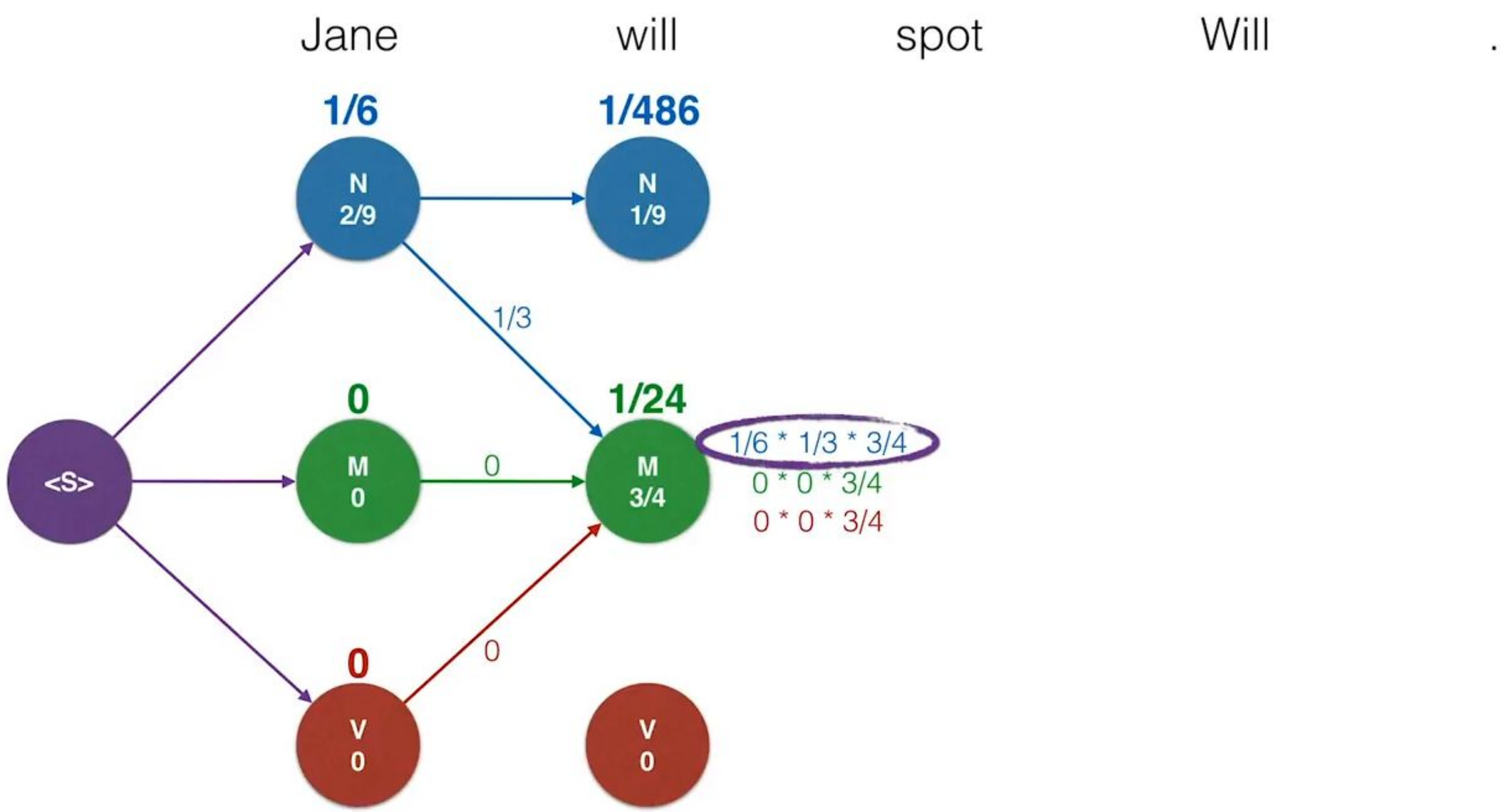
Jane
will
spot
Will
.
1/6
N
2/9
1/9
N
1/9
1/6 * 1/9 * 1/9
0 * 1/4 * 1/9
0 * 1* 1/9
1/4
0
<S>
M
0
1
M
3/4
0
V
0
V
0

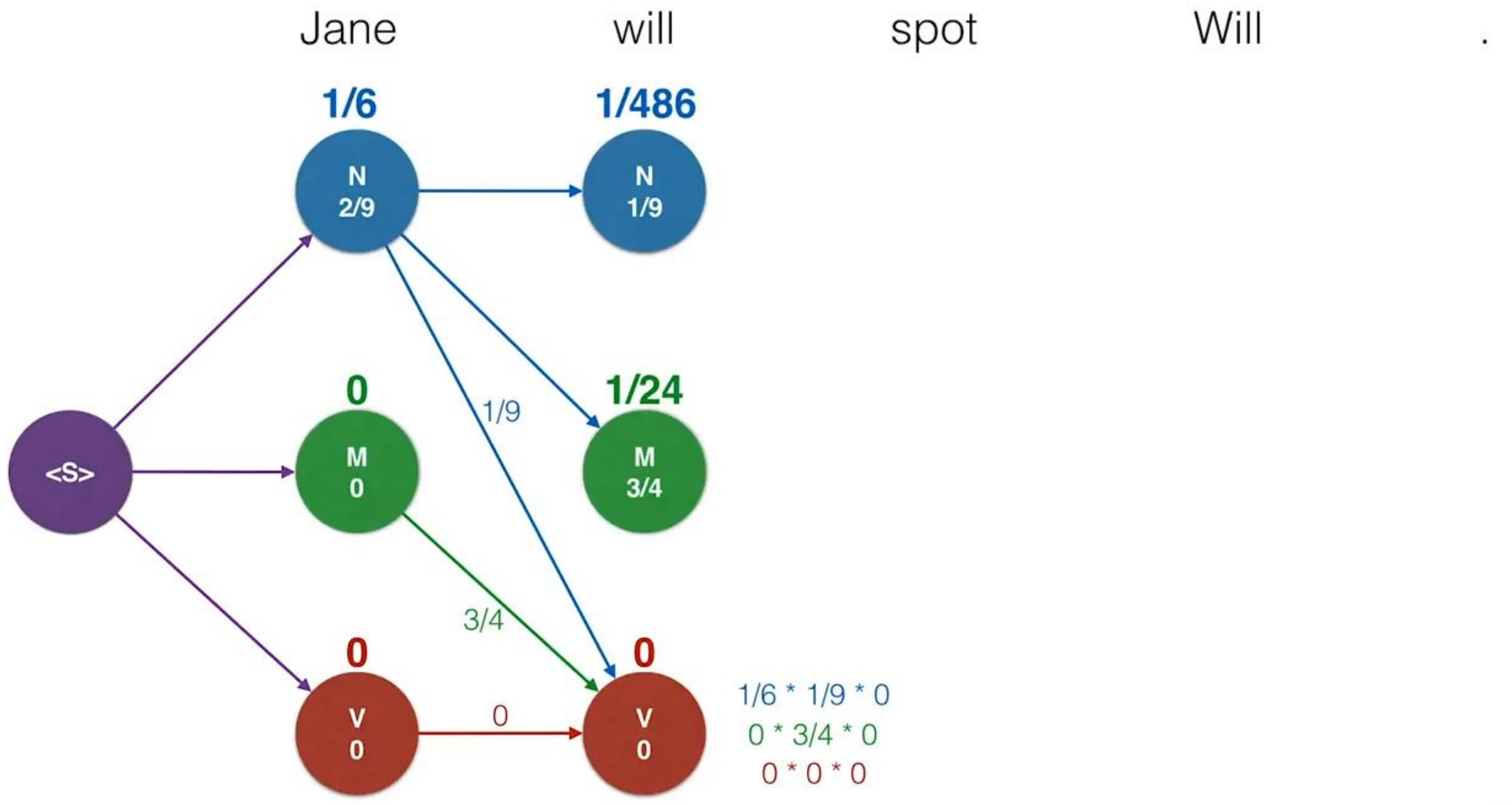

Jane
will
spot
Will
.
1/6
N
2/9
1/9
N
1/9
1/6 * 1/9 * 1/9
0 * 1/4 * 1/9
0 * 1* 1/9
1/4
0
M
0
M
3/4
<S>
1
0
V
0
V
0



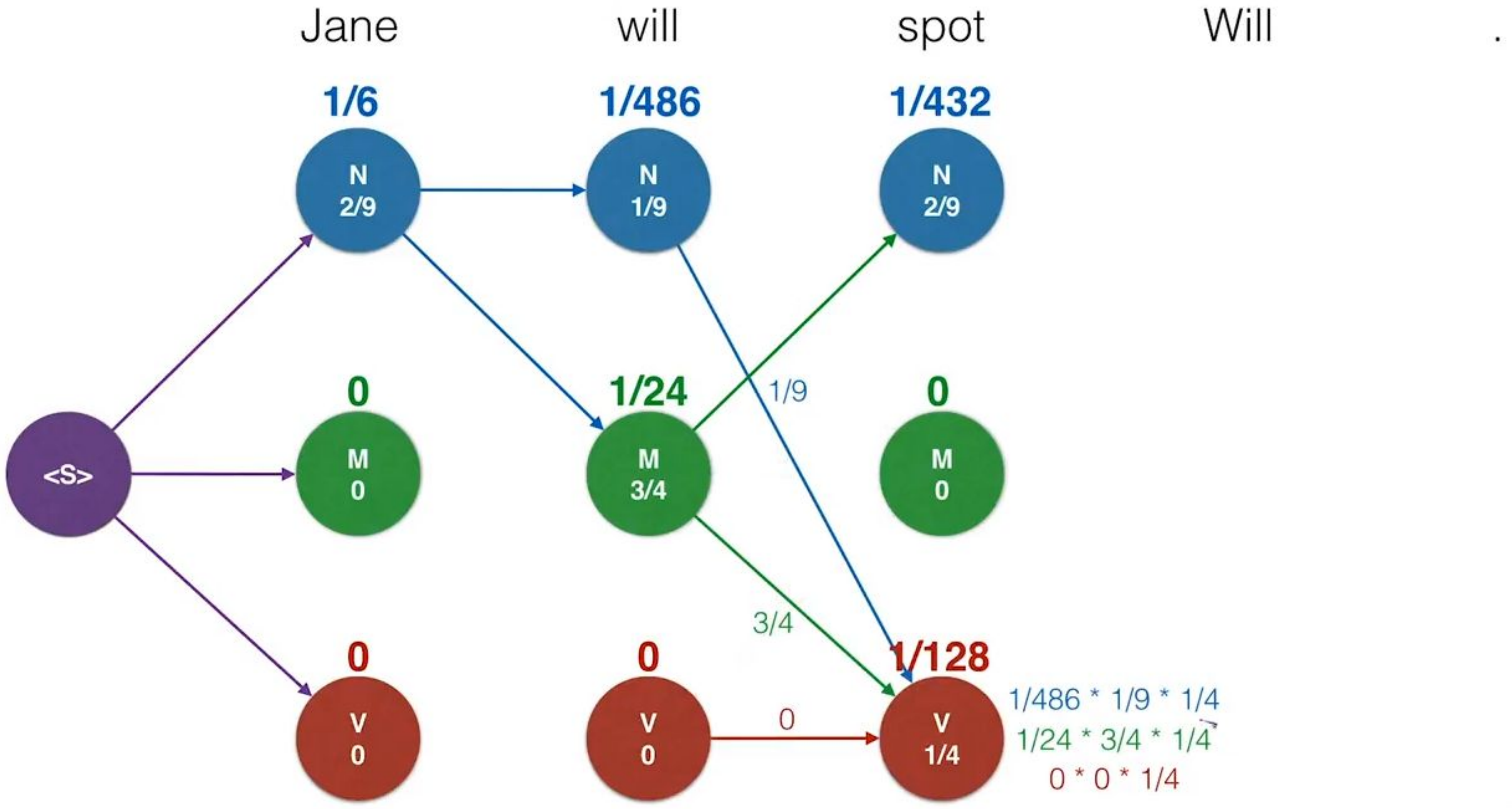
Jane
will
spot
Will
.
1/6
1/486
1/6 * 1/9 * 1/9
0 * 1/4 * 1/9
0 * 1* 1/9
1/9
N
2/9
N
1/9
1/4
0
<S>
M
0
M
3/4
1
0
V
0
V
0

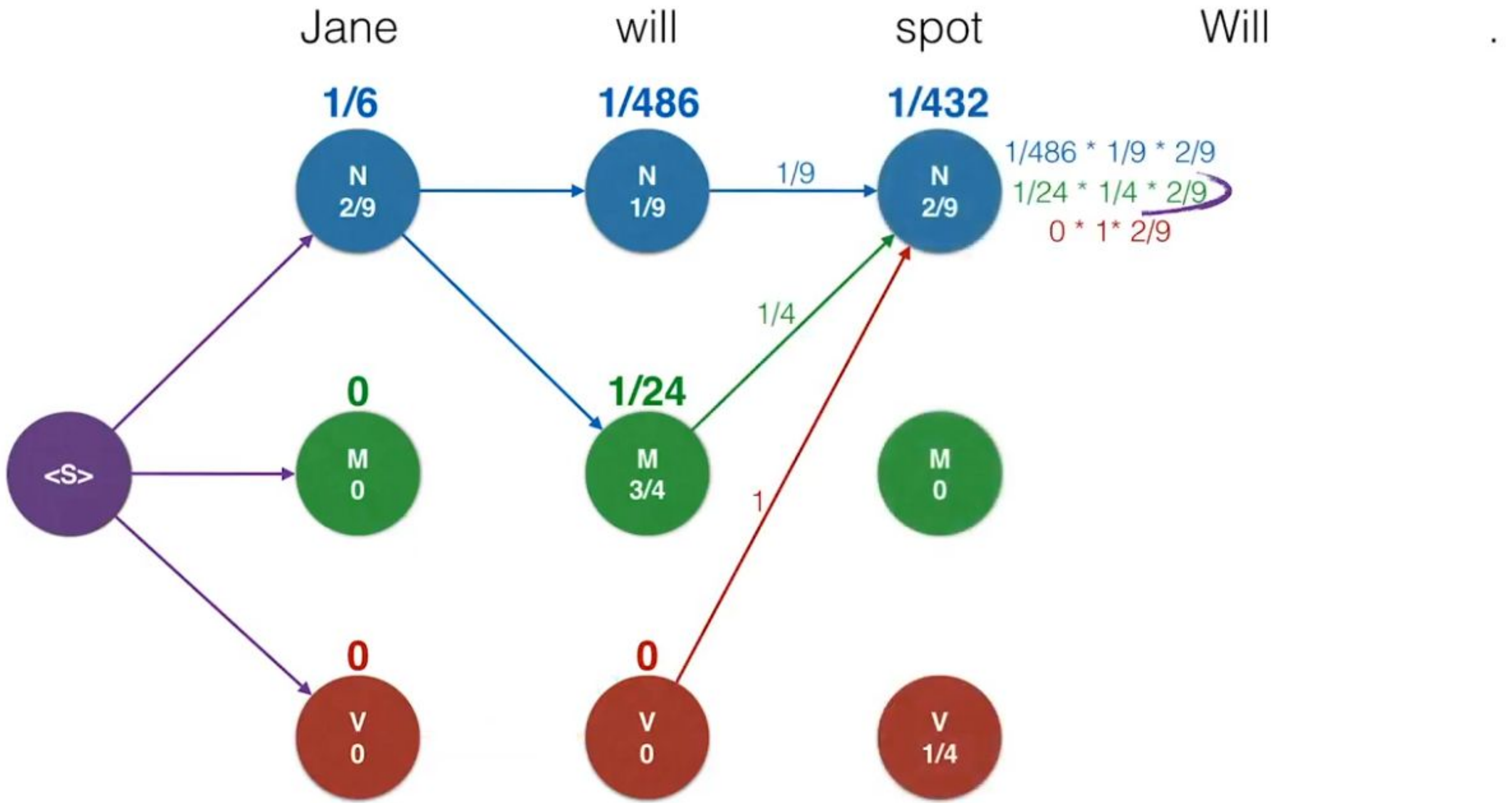

Jane
will
spot
Will
.
1/6
1/486
N
2/9
N
1/9
1/3
0
<S>
M
0
0
M
3/4
1/6 * 1/3 * 3/4
0 * 0 * 3/4
0 * 0 * 3/4
0
0
V
0
V
0


Jane
will
spot
Will
.
1/6
1/486
N
2/9
N
1/9
1/3
0
1/24
<S>
M
0
0
M
3/4
1/6 * 1/3 * 3/4
0 * 0 * 3/4
0 * 0 * 3/4
0
0
V
0
V
0

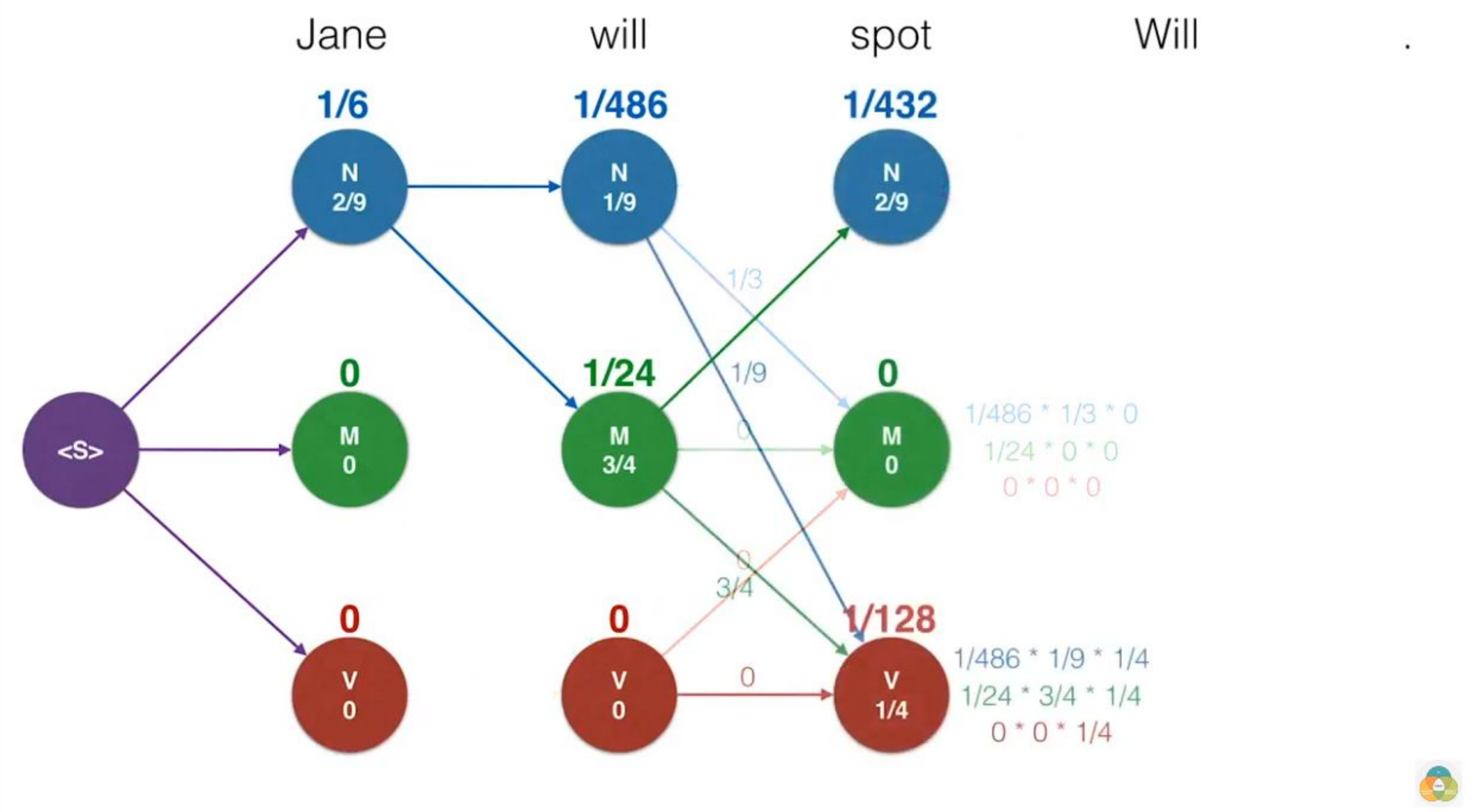


Jane
will
spot
Will
.
1/6
1/486
N
2/9
N
1/9
0
1/24
1/9
<S>
M
0
M
3/4
3/4
0
0
V
0
0
V
0
1/6 * 1/9 * 0
0 * 3/4 * 0
0 * 0 * 0

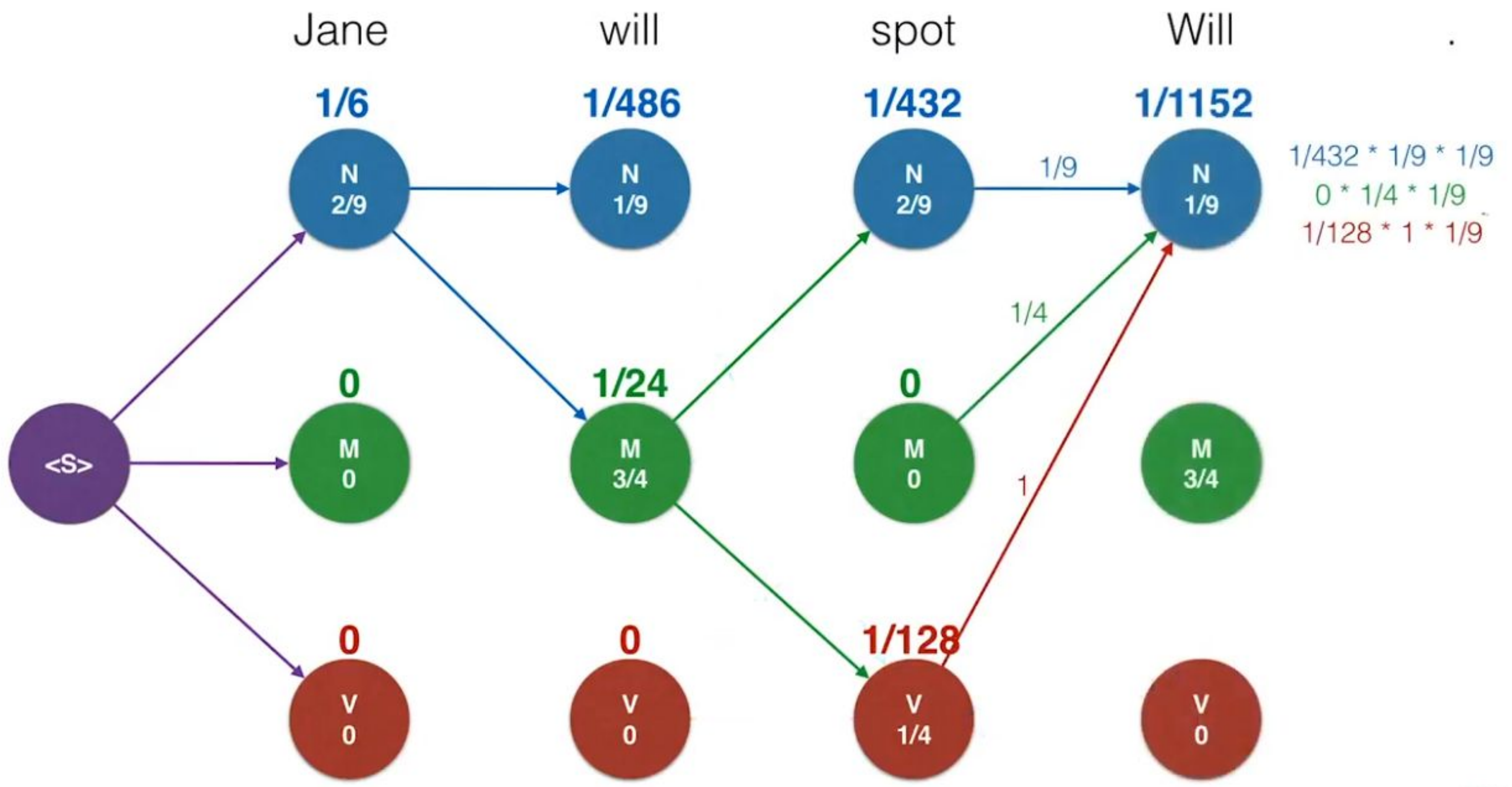

Jane
will
spot
Will
.
1/6
1/486
1/432
N
2/9
N
1/9
N
2/9
0
1/24
1/9
0
<S>
M
0
M
3/4
M
0
3/4
0
0
1/128
V
0
V
0
0
V
1/4
1/486 * 1/9 * 1/4
1/24 * 3/4 * 1/4
0 * 0 * 1/4

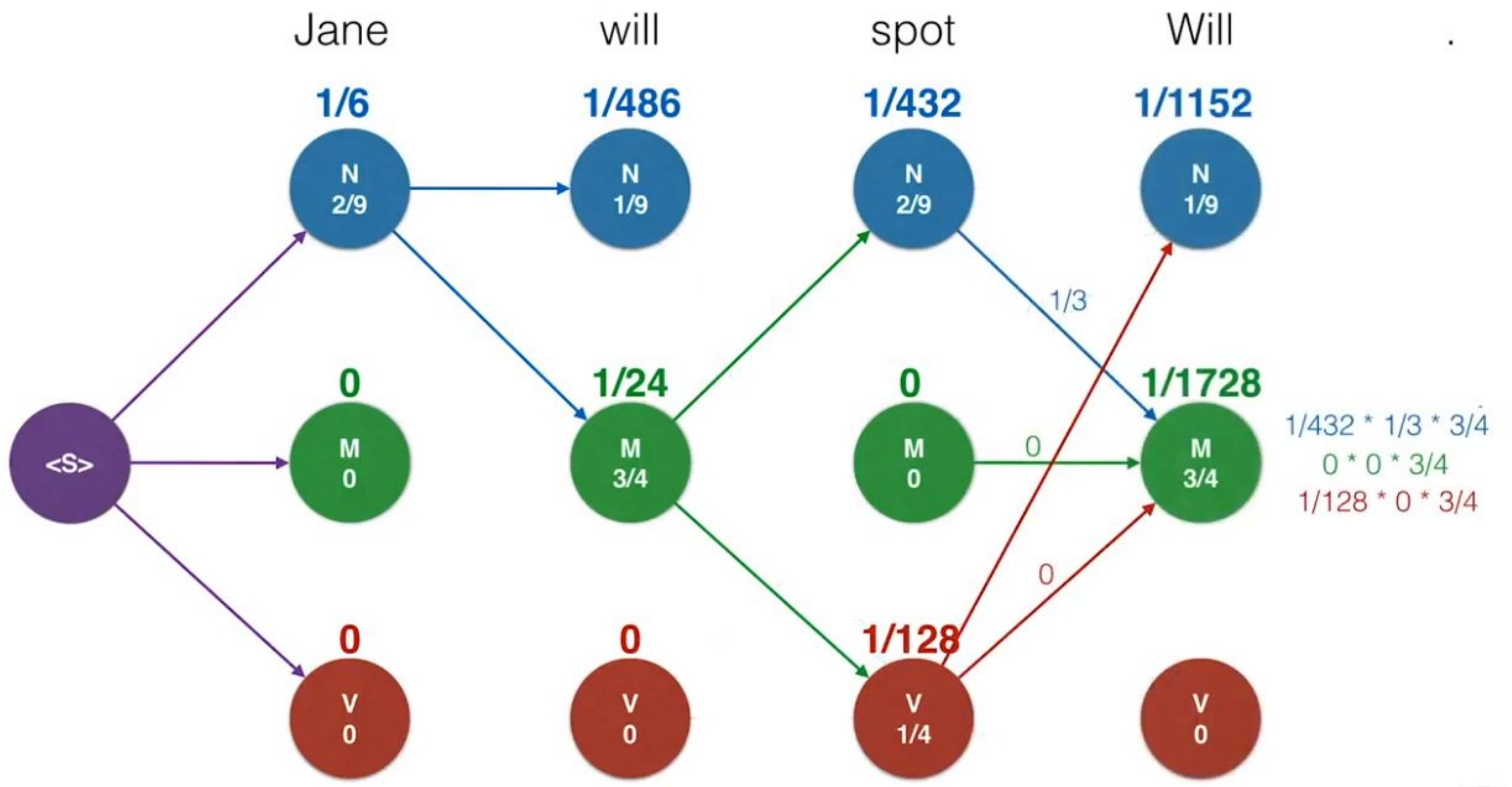


Jane
will
spot
Will
.
1/6
1/486
1/432
1/486 * 1/9 * 2/9
1/24 * 1/4 * 2/9
0 * 1* 2/9
N
2/9
N
1/9
N
2/9
1/9
1/4
0
1/24
<S>
M
0
M
3/4
M
0
1
0
0
V
0
V
0
V
1/4

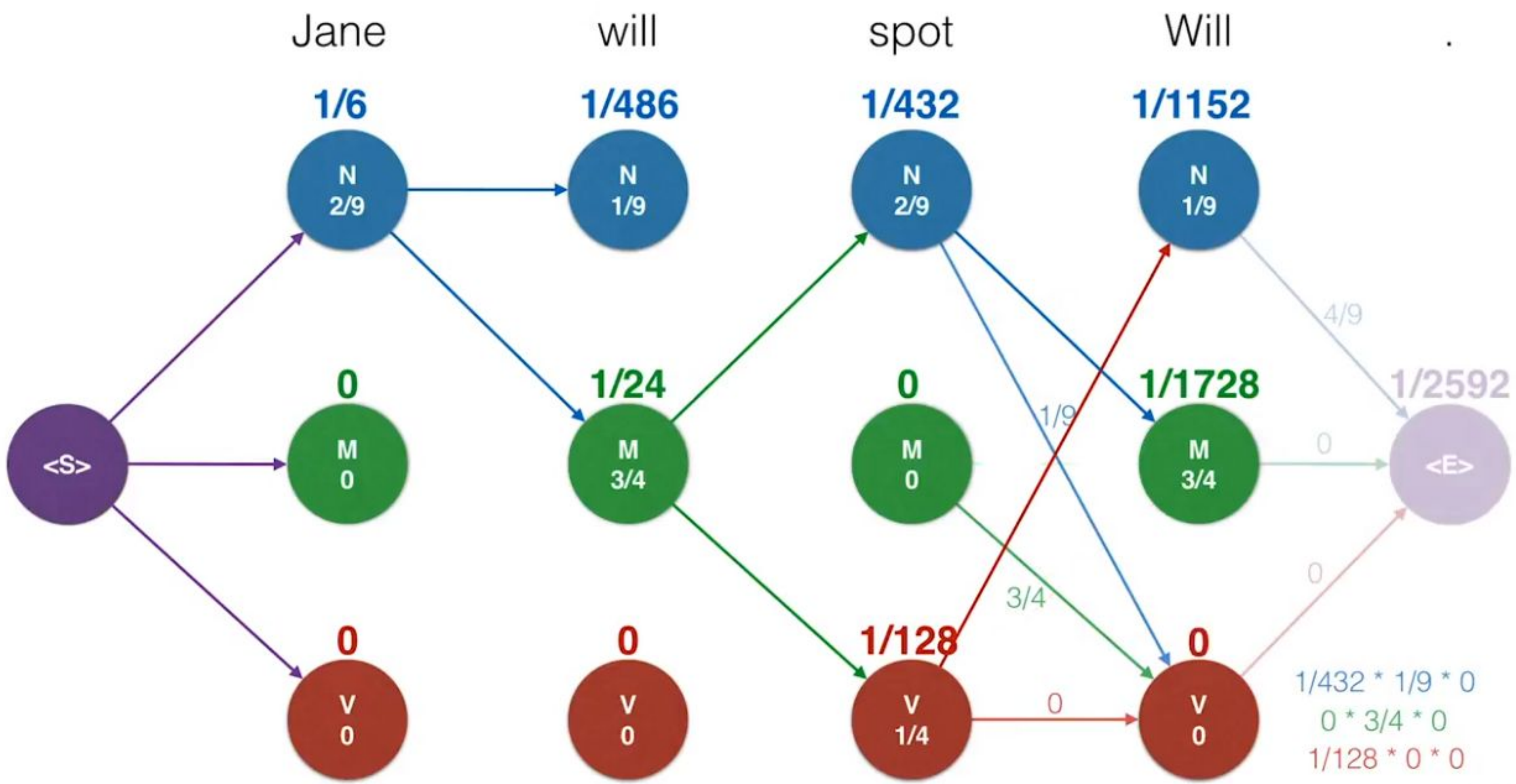


Jane
will
spot
Will
.
1/6
1/486
1/432
N 2/9
N 1/9
N 2/9
1/3
1/9
0
1/24
0
<S>
M 0
M 3/4
M 0
1/486 * 1/3 * 0
1/24 * 0 * 0
0 * 0 * 0
0
3/4
0
0
1/128
V 0
V 0
V 1/4
1/486 * 1/9 * 1/4
1/24 * 3/4 * 1/4
0 * 0 * 1/4

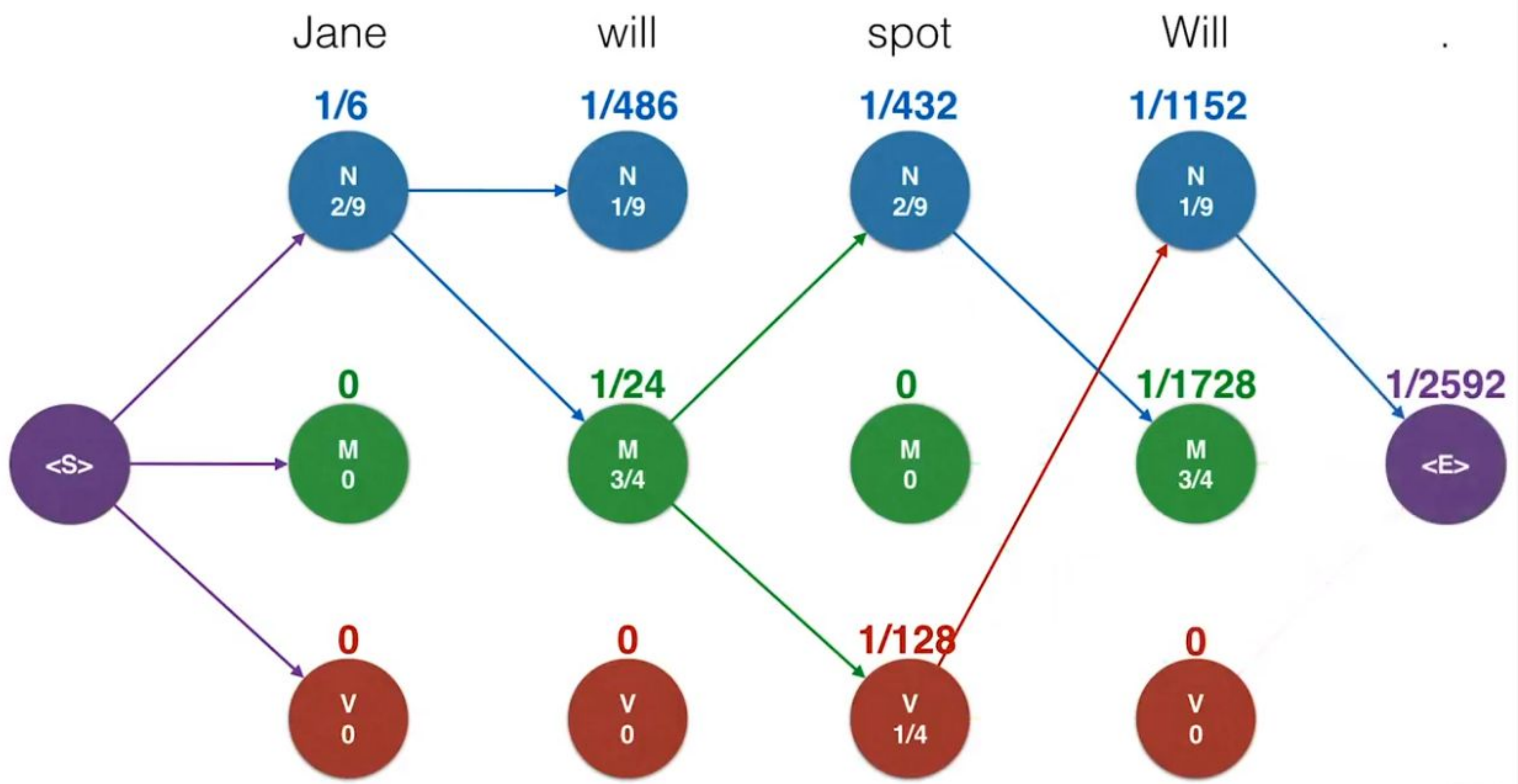

Jane
will
spot
Will
.
1/6
1/486
1/432
1/1152
N 2/9
N 1/9
N 2/9
N 1/9
1/9
1/432 * 1/9 * 1/9
0 * 1/4 * 1/9
1/128 * 1 * 1/9
1/4
0
1/24
0
<S>
M 0
M 3/4
M 0
M 3/4
1
0
0
1/128
V 0
V 0
V 1/4
V 0

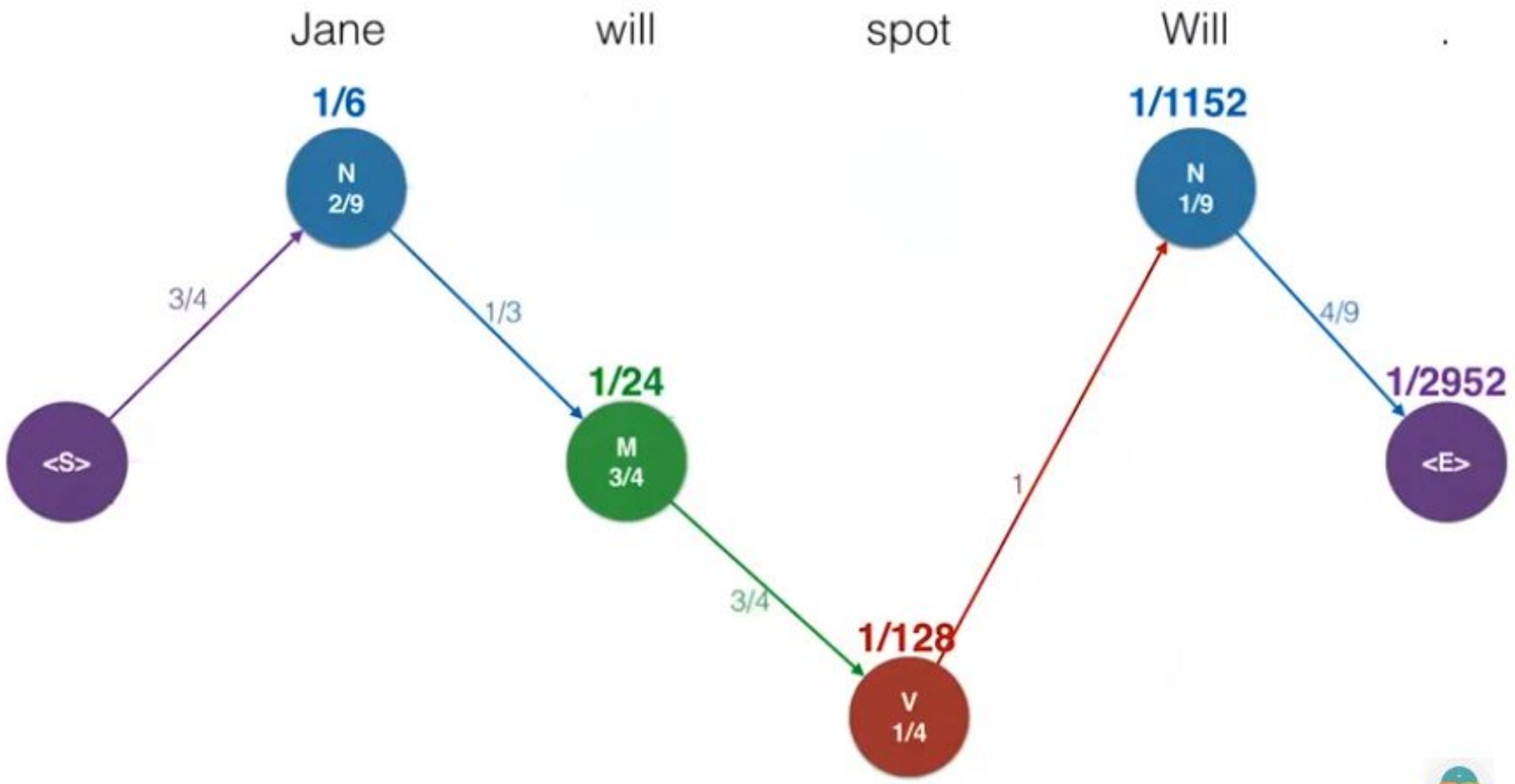


Jane
will
spot
Will
.
1/6
1/486
1/432
1/1152
N 2/9
N 1/9
N 2/9
N 1/9
1/3
0
1/24
0
1/1728
<S>
M 0
M 3/4
M 0
M 3/4
0
1/432 * 1/3 * 3/4
0 * 0 * 3/4
1/128 * 0 * 3/4
0
0
0
1/128
V 0
V 0
V 1/4
V 0



Jane
will
spot
Will
.
1/6
1/486
1/432
1/1152
N
2/9
N
1/9
N
2/9
N
1/9
0
1/24
0
1/1728
1/2592
<S>
M
0
M
3/4
M
0
M
3/4
<E>
4/9
1/9
0
3/4
0
0
1/128
0
0
V
0
V
0
V
1/4
V
0
1/432 * 1/9 * 0
0 * 3/4 * 0
1/128 * 0 * 0


Jane
will
spot
Will
.
1/6
1/486
1/432
1/1152
N
2/9
N
1/9
N
2/9
N
1/9
0
1/24
0
1/1728
1/2592
<S>
M
0
M
3/4
M
0
M
3/4
<E>
0
0
1/128
0
V
0
V
0
V
1/4
V
0


Jane
will
spot
Will
.
1/6
N
2/9
1/1152
N
1/9
3/4
1/3
4/9
1/24
M
3/4
1/2952
<S>
<E>
1
3/4
1/128
V
1/4

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:

https://www.youtube.com/watch?v=mHEKZ8jv2SY

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

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)

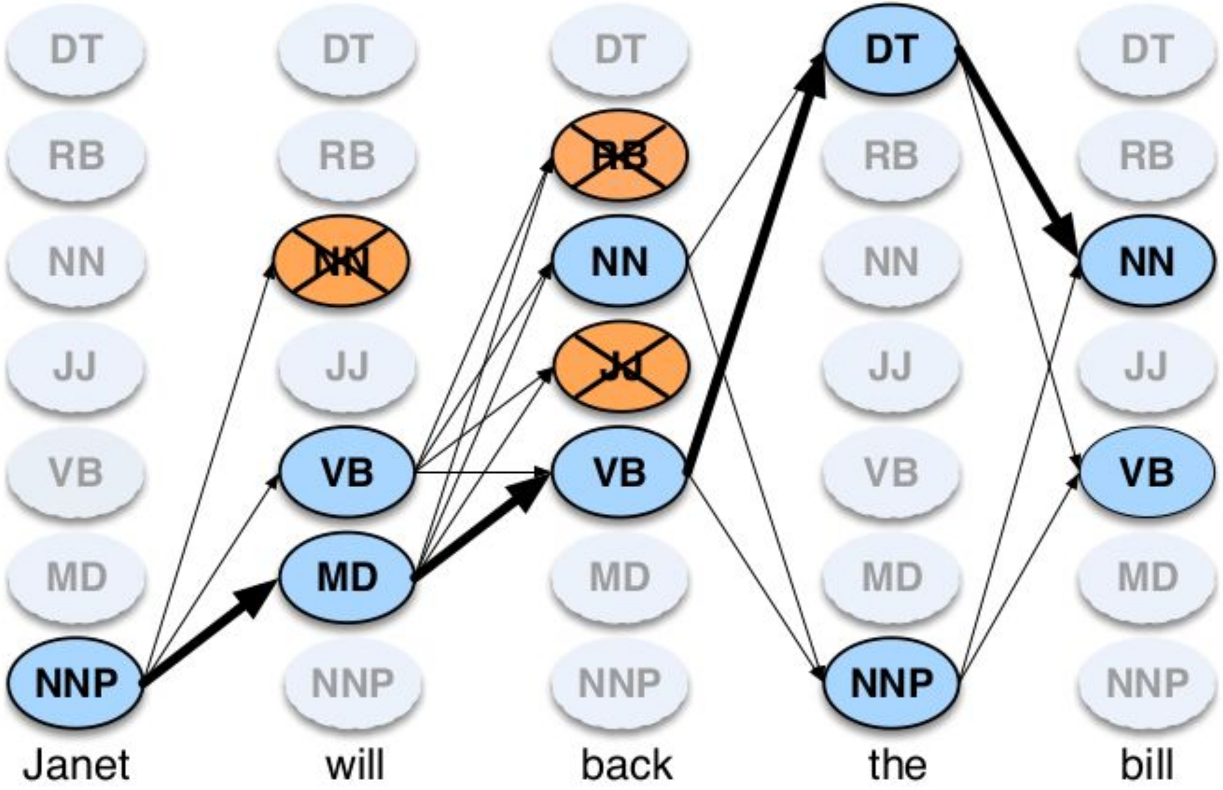


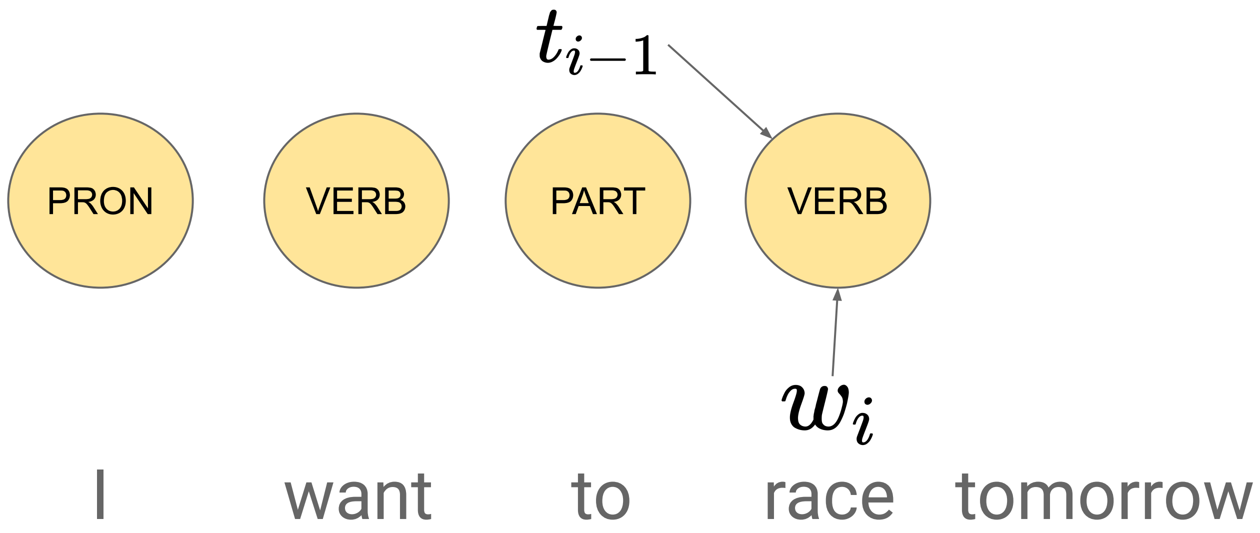
Figure from Jurafsky, D and Martin, J, “Speech and Language Processing,” 2018

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

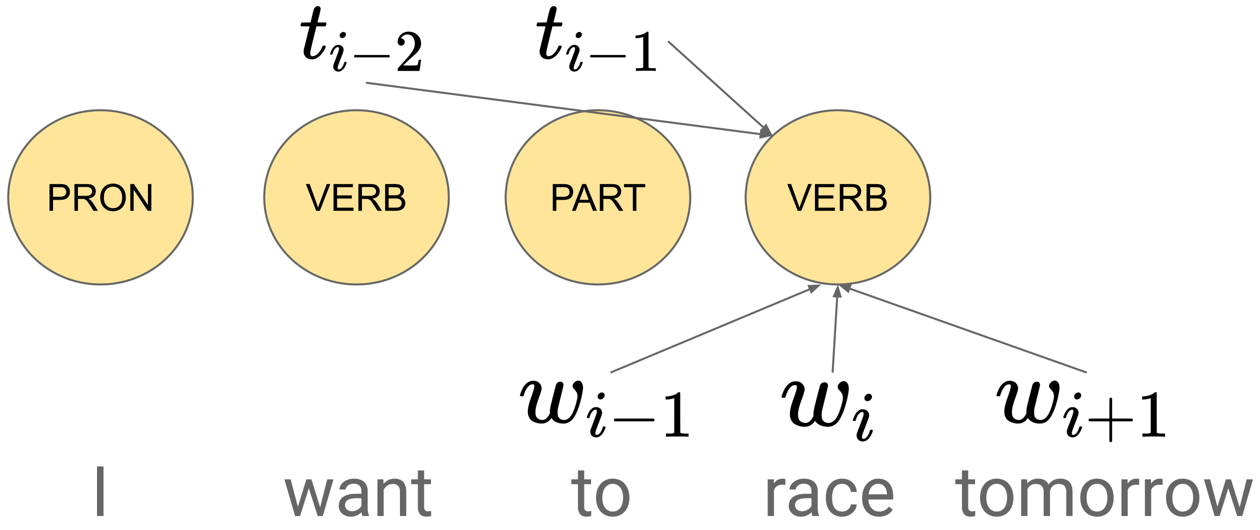
$$\hat{T} = argmax_T P(T|W)$$
$$= argmax_T \prod P(t_i|w_i, t_{i-1})$$

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

- What else could we use?
  - All sorts of features

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

HMM vs MEMM

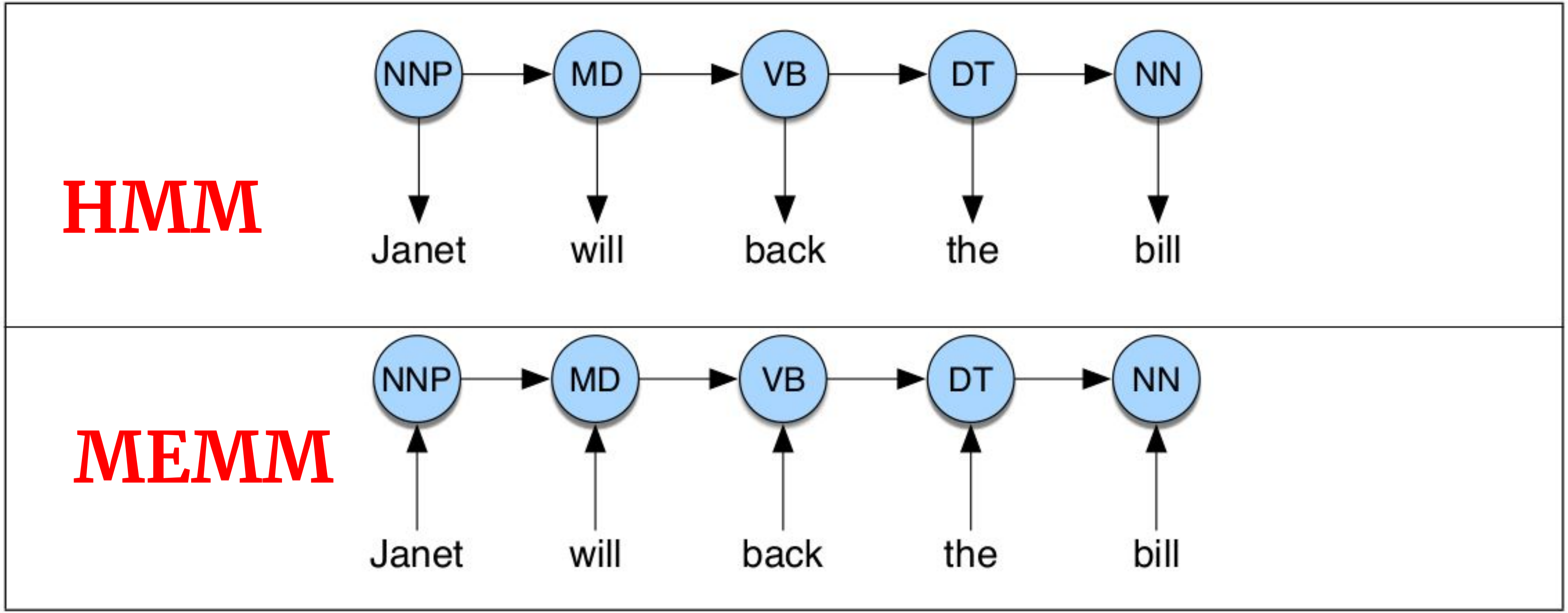


Figure from Jurafsky, D and Martin, J, “Speech and Language Processing,” 2018

HMM vs MEMM

MEMM Inference

$$\hat{T} = argmax_T P(T|W)$$
$$= \boxed{argmax_T \prod_i P(t_i|t_{i-1}, w_i)}$$

HMM Inference

$$\hat{T} = argmax_T P(T|W)$$
$$\hat{T} = argmax_T P(W|T)P(T)$$
$$= \boxed{argmax_T \prod_i P(w_i|t_i)p(t_i|t_{i-1})}$$

HMM vs MEMM

- Versus VITERBI for HMM

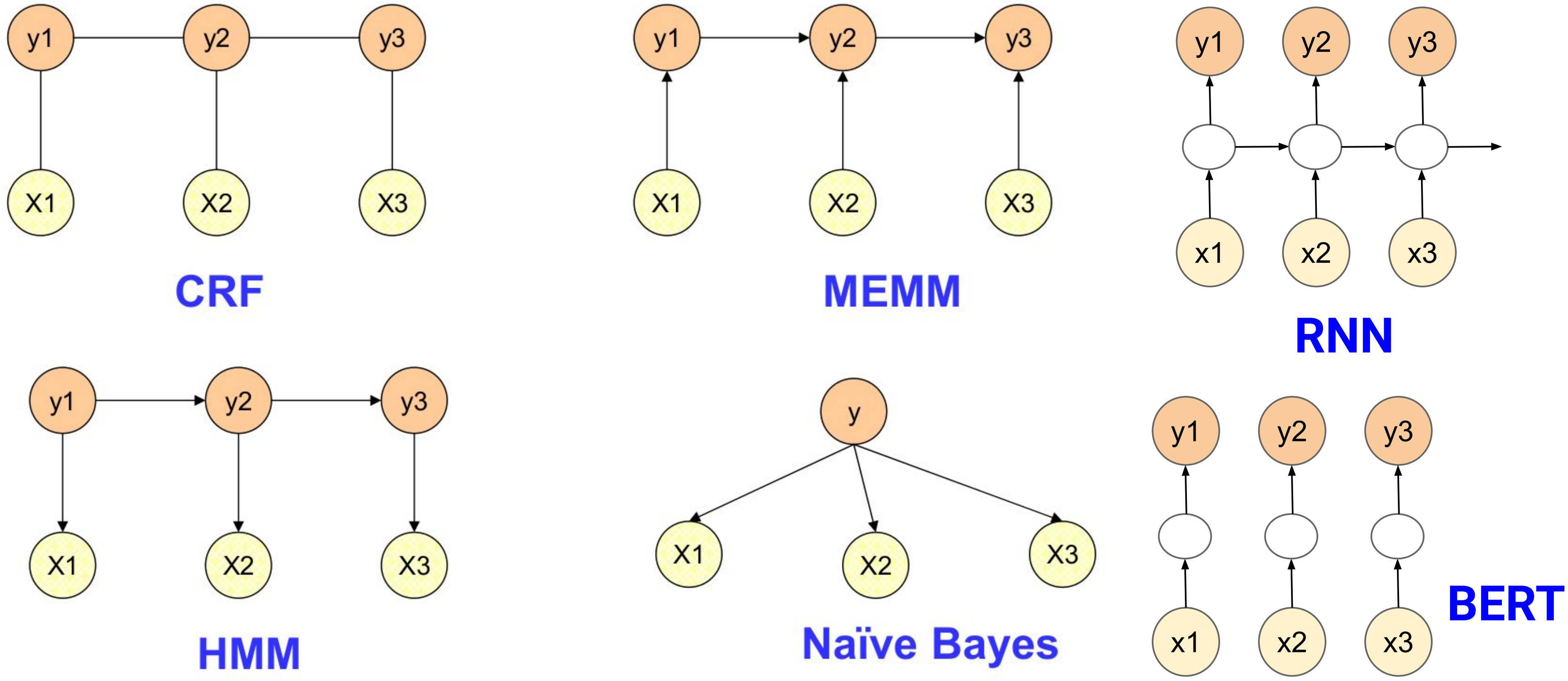
$$v_t(j) = max_{i=1}^{N} v_{t-1}(i) P(q_j|q_i) P(o_t|q_j)$$

- 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

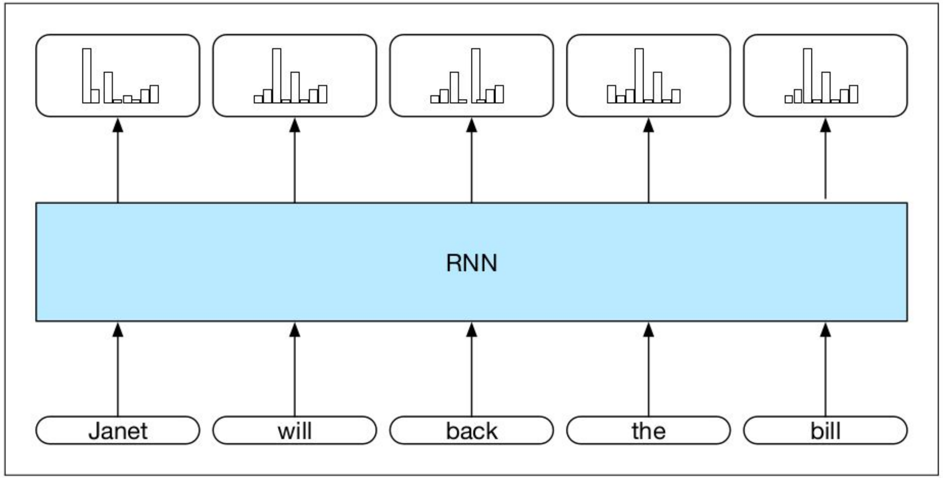


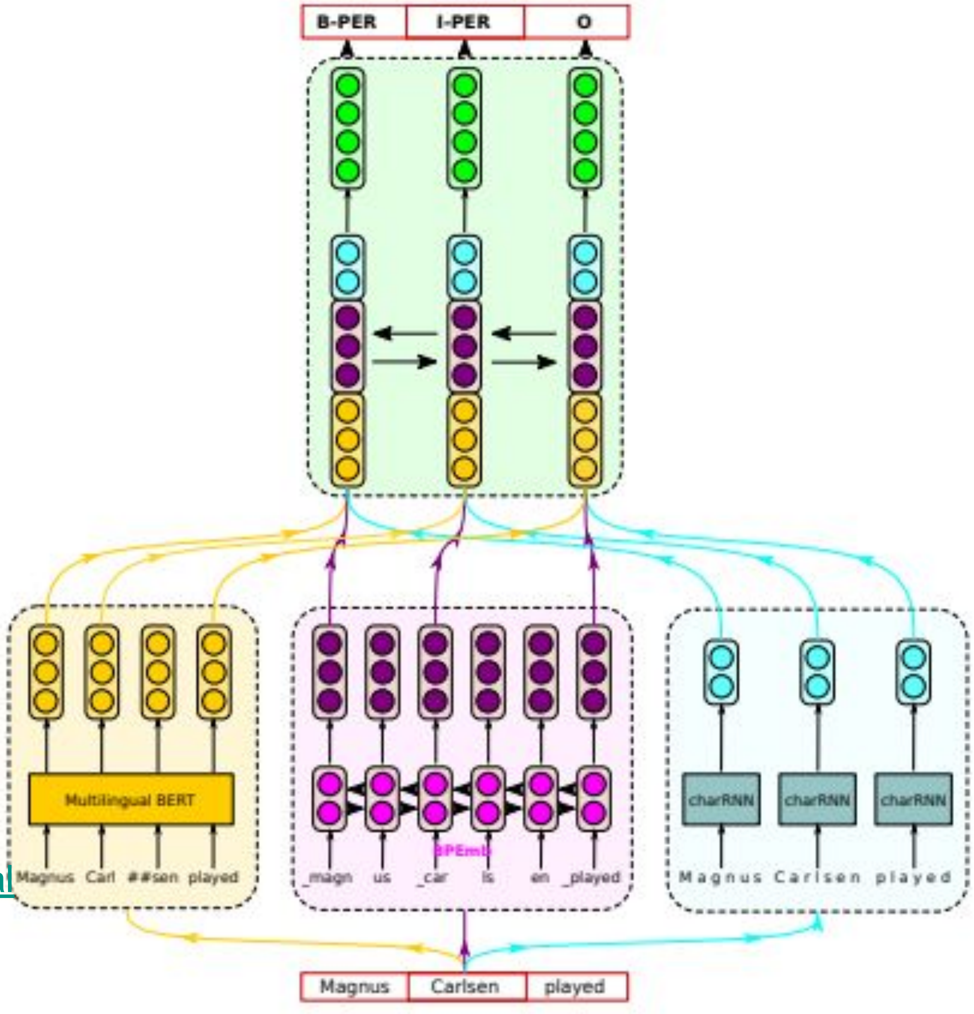
Figure from Jurafsky, D and Martin, J, “Speech and Language Processing,” 2018

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

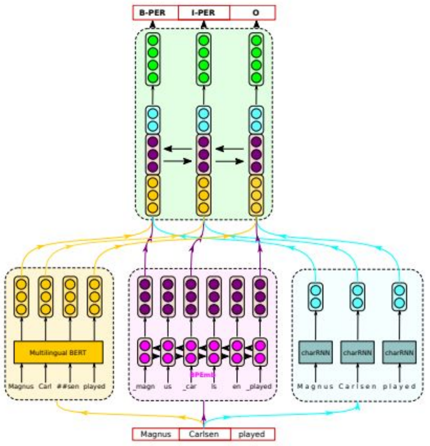
SOTA

- 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

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

SOTA for POS tagging – low res langs.

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