

# Text Mining

March 3, 2017

# Outline

- Language Identification
- Tokenisation
- Part-Of-Speech (POS) tagging
  - Hidden Markov Models - Sequential Taggers
  - Viterbi Algorithm

# Introduction

- Process of determining the natural language in which the contents of the text is written.
- An important research area - focused as early as 1970s.
- Two approaches:
  - non-computational: linguistic knowledge, such as diacritics and symbols, most frequent words used, character combination etc.
  - computational: statistical techniques rather than linguistic knowledge

# Architecture of Language Identifier

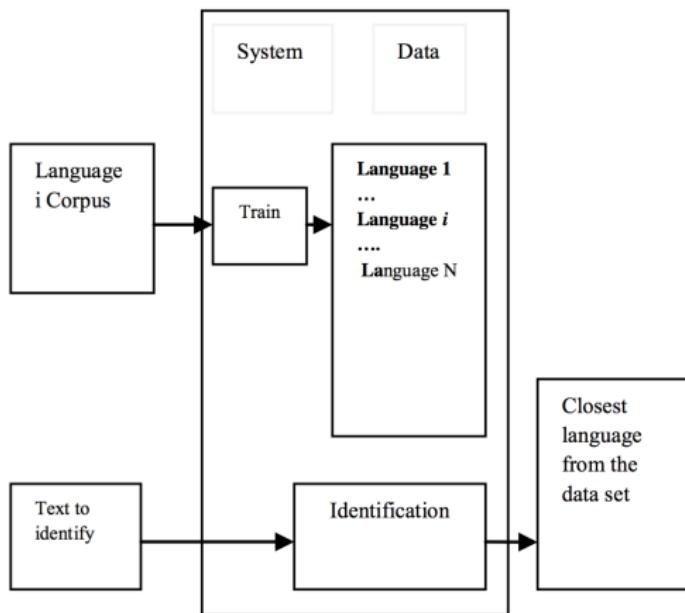


Figure: General architecture of a language identifier [Padró and Padró, 2004]

# N-Gram-based Text Categorization [Cavnar et al., 1994]

- **Zipf's law:** The frequency of any word is inversely proportional to its rank in the frequency table.

Word	Freq. ( $f$ )	Rank ( $r$ )	$f \cdot r$	Word	Freq. ( $f$ )	Rank ( $r$ )	$f \cdot r$
the	3332	1	3332	turned	51	200	10200
and	2972	2	5944	you'll	30	300	9000
a	1775	3	5235	name	21	400	8400
he	877	10	8770	comes	16	500	8000
but	410	20	8400	group	13	600	7800
be	294	30	8820	lead	11	700	7700
there	222	40	8880	friends	10	800	8000
one	172	50	8600	begin	9	900	8100
about	158	60	9480	family	8	1000	8000
more	138	70	9660	brushed	4	2000	8000
never	124	80	9920	sins	2	3000	6000
Oh	116	90	10440	Could	2	4000	8000
two	104	100	10400	Applausive	1	8000	8000

Figure: Empirical evaluation of Zipf's law on Tom Sawyer

- There is always a set of words which dominates most of the other words of the language in terms of frequency of use.

# N-Gram-based Text Categorization [Cavnar et al., 1994]

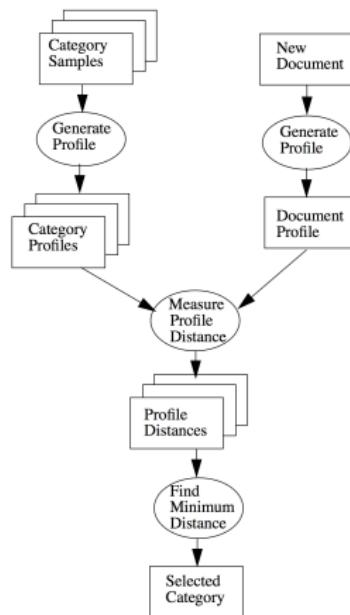


Figure: Dataflow for N-Gram-based text categorization [Cavnar et al., 1994]

# N-Gram-and-Wikipedia Joint Approach [Yang and Liang, 2010]

- Previous methods required large corpus as training data for determining each language.
- Joint approach solved this problem by using a single language corpus i.e., local English corpus for all languages.
- Two steps:
  - implementation of segmentation algorithm which uses n-gram frequency statistics for segmentation into language-specific units
  - determine language of each unit by utilizing different language versions of Wikipedia.
- Wikipedia supports 262 language versions of the documents.
- Joint approach achieves approximately 100% accurate results

# Tokenisation

- Tokenisation is the process of splitting a text into words
- What is a word?
  - string of contiguous alphanumeric characters with space on either side; may include hyphens and apostrophes, but no other punctuation marks [Kučera and Francis, 1967].
- Difficulties: \$22.50; Micro\$oft; C# net; :-)
- Occurrence of *whitespace* - a space or tab is the main clue in English
- Full stop, comma etc can be used as pre-defined delimiters
- For languages that do not use spaces to separate words, supervised sequential tagging methods are used to perform tokenisation during morphological analysis.

# Part-of-Speech Tagging (POS)

- Task of tagging POS tags (Nouns, Verbs, Adjectives, Adverbs, ...) for words
- POS tags provide lot of information about a word
  - knowing whether a word is **noun** or **verb** gives information about neighbouring words
  - nouns are preceded by determiners; adjectives and verbs by nouns
  - provide useful features for **named entity recognition**
- Given a word, we assume it can belong to only of the POS tags.
- POS Tagging problem
  - Given a sentence  $S = w_1 w_2 \dots w_n$  consisting of  $n$  words, determine the corresponding tag sequence  $P = P_1 P_2 \dots P_n$

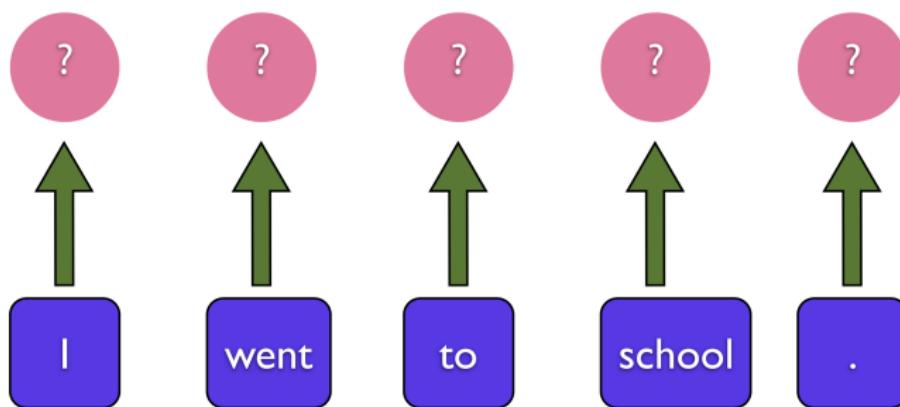
# Part-of-Speech Tagging (POS)

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	<i>and, but, or</i>	SYM	symbol	<i>+, %, &amp;</i>
CD	cardinal number	<i>one, two</i>	TO	"to"	<i>to</i>
DT	determiner	<i>a, the</i>	UH	interjection	<i>ah, oops</i>
EX	existential 'there'	<i>there</i>	VB	verb base form	<i>eat</i>
FW	foreign word	<i>mea culpa</i>	VBD	verb past tense	<i>ate</i>
IN	preposition/sub-conj	<i>of, in, by</i>	VBG	verb gerund	<i>eating</i>
JJ	adjective	<i>yellow</i>	VBN	verb past participle	<i>eaten</i>
JJR	adj., comparative	<i>bigger</i>	VBP	verb non-3sg pres	<i>eat</i>
JJS	adj., superlative	<i>wildest</i>	VBZ	verb 3sg pres	<i>eats</i>
LS	list item marker	<i>I, 2, One</i>	WDT	wh-determiner	<i>which, that</i>
MD	modal	<i>can, should</i>	WP	wh-pronoun	<i>what, who</i>
NN	noun, sing. or mass	<i>llama</i>	WP\$	possessive wh-	<i>whose</i>
NNS	noun, plural	<i>llamas</i>	WRB	wh-adverb	<i>how, where</i>
NNP	proper noun, sing.	<i>IBM</i>	\$	dollar sign	<i>\$</i>
NNPS	proper noun, plural	<i>Carolinias</i>	#	pound sign	<i>#</i>
PDT	predeterminer	<i>all, both</i>	"	left quote	<i>' or "</i>
POS	possessive ending	<i>'s</i>	"	right quote	<i>' or "</i>
PRP	personal pronoun	<i>I, you, he</i>	(	left parenthesis	<i>[, (, {, &lt;</i>
PRP\$	possessive pronoun	<i>your, one's</i>	)	right parenthesis	<i>], ), }, &gt;</i>
RB	adverb	<i>quickly, never</i>	,	comma	<i>,</i>
RBR	adverb, comparative	<i>faster</i>	.	sentence-final punc	<i>. ! ?</i>
RBS	adverb, superlative	<i>fastest</i>	:	mid-sentence punc	<i>: ; ... – -</i>
RP	particle	<i>up, off</i>			

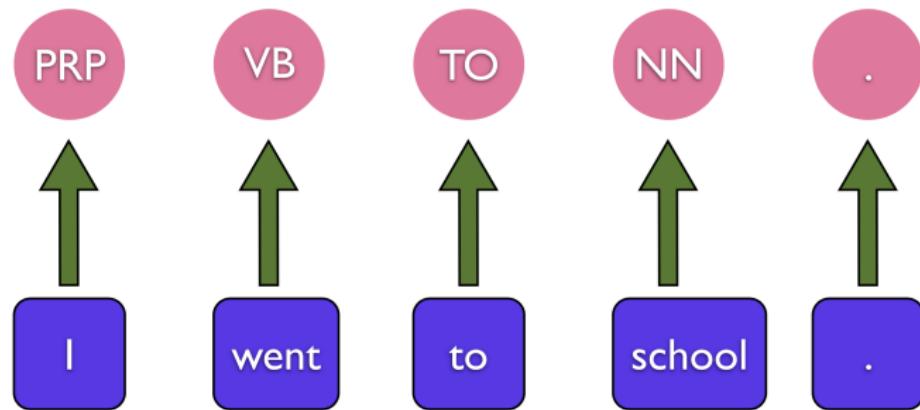
Figure: Penn Treebank POS Tags

# Hidden Markov Model (HMM)

Given a sequence of words (observed states)  
determine a sequence of state transitions (unobserved states)

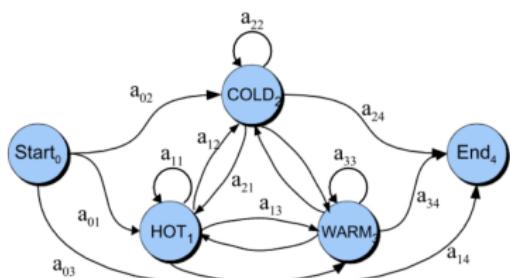


# Hidden Markov Model (HMM)



# Markov Chains

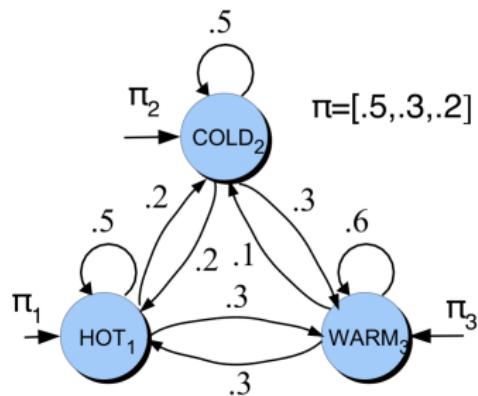
- Probabilistic **graphical model** for representing probabilistic assumptions in a graph.



- $Q = q_1, q_2, \dots, q_N$  : a set of **states**
- $A = a_{01}, a_{02}, \dots, a_{n1}, \dots, a_{nn}$  : a **transition probability matrix A**, each  $a_{ij}$  representing the probability of moving from state  $i$  to state  $j$ , s.t.  $\sum_{j=1}^n a_{ij} = 1 \quad \forall i$
- $q_0, q_{end}$  : a special *start and end state* which are not associated with observations

# Markov Chains

$\pi_1, \pi_2, \dots, \pi_N$  : an **initial probability distribution** over states.  $\pi_i$  is the probability that the Markov chain will start in state  $i$ .

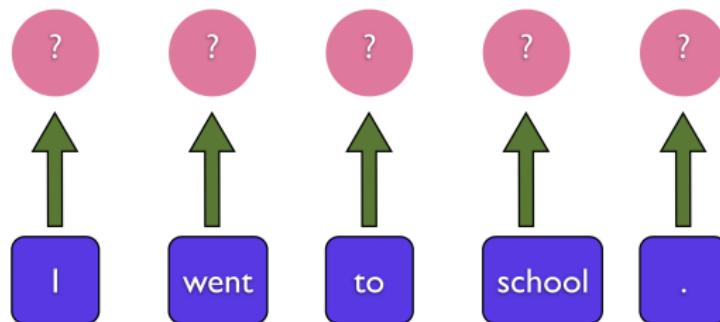


- Markov Assumption:  
 $P(q_i|q_1, q_2, \dots, q_{i-1}) = P(q_i|q_{i-1})$
- $P(\text{cold hot cold hot}) = P(\text{cold}) P(\text{hot}|\text{cold}) P(\text{cold}|\text{hot}) P(\text{hot}|\text{cold}) = 0.3 \times 0.2 \times 0.2 \times 0.2 = 0.0024$

# Hidden Markov Model (HMM)

- Markov chains are useful for observed events
- However, in many cases the events are not observed
  - Example: POS tagging - POS tags are not observed

Given a sequence of words (observed states)  
determine a sequence of state transitions (unobserved states)



- HMMs allow us to model both *observed events* (words that we see) and *hidden events* (POS tags).

# A motivating example

**Urn 1**

# of Red = 30

# of Green = 50

# of Blue = 20

**Urn 2**

# of Red = 10

# of Green = 40

# of Blue = 50

**Urn 3**

# of Red = 60

# of Green = 10

# of Blue = 30

Probability of transition to another Urn after picking a ball:

	$U_1$	$U_2$	$U_3$
$U_1$	0.1	0.4	0.5
$U_2$	0.6	0.2	0.2
$U_3$	0.3	0.4	0.3

# A Motivating Example (contd.)

Given: Transition Probabilities

	$U_1$	$U_2$	$U_3$
$U_1$	0.1	0.4	0.5
$U_2$	0.6	0.2	0.2
$U_3$	0.3	0.4	0.3

Given: Output Probabilities

	R	G	B
$U_1$	0.3	0.5	0.2
$U_2$	0.1	0.4	0.5
$U_3$	0.6	0.1	0.3

Observation: RRGGGBRGR

State Sequence (Urn chosen corresponding to each ball): ?

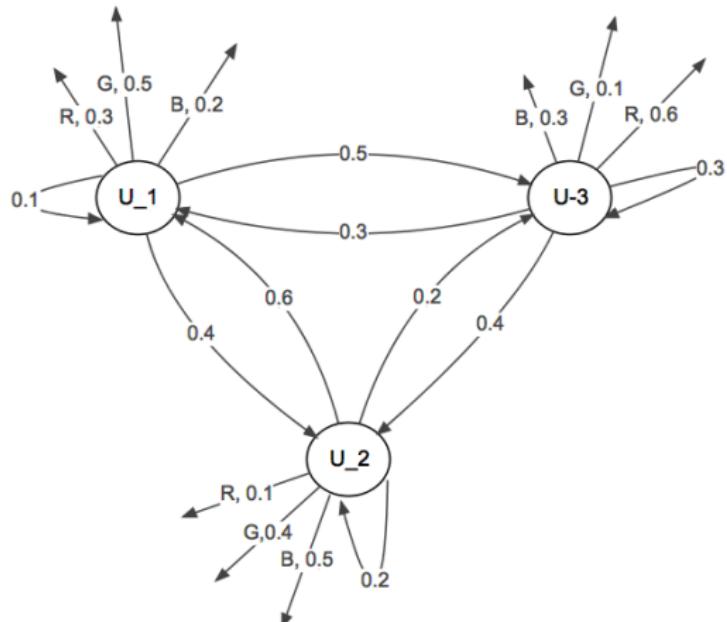
# Diagrammatic Representation - 1

Transition Probabilities

	$U_1$	$U_2$	$U_3$
$U_1$	0.1	0.4	0.5
$U_2$	0.6	0.2	0.2
$U_3$	0.3	0.4	0.3

Output Probabilities

	R	G	B
$U_1$	0.3	0.5	0.2
$U_2$	0.1	0.4	0.5
$U_3$	0.6	0.1	0.3



Observation: RRGGGBRGR

State Sequence (Urn chosen corresponding to each ball): ?

# Diagrammatic Representation - 2

## Transition Probabilities

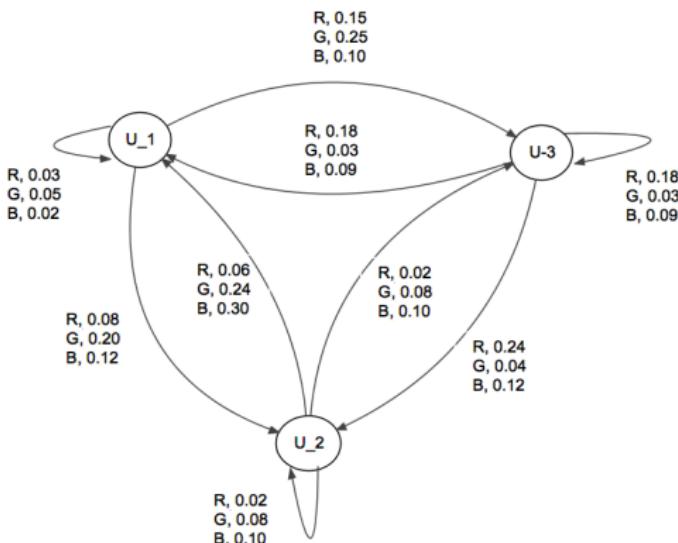
	$U_1$	$U_2$	$U_3$
$U_1$	0.1	0.4	0.5
$U_2$	0.6	0.2	0.2
$U_3$	0.3	0.4	0.3

## Output Probabilities

	R	G	B
$U_1$	0.3	0.5	0.2
$U_2$	0.1	0.4	0.5
$U_3$	0.6	0.1	0.3

Observation: RRGGBRGR

State Sequence (Urn chosen corresponding to each ball): ?



## Example (contd.)

- States Set:  $S = \{U_1, U_2, U_3\}$
- Observation Set:  $V = \{R, G, B\}$
- Observation Sequence:
  - $O = \{O_1, \dots, O_n\}$
- State Sequence:
  - $Q = \{q_1, \dots, q_n\}$
- Initial Probability:  $\epsilon$ 
  - $\epsilon_i = P(q_i = U_i)$

Transition Probabilities (A)

	$U_1$	$U_2$	$U_3$
$U_1$	0.1	0.4	0.5
$U_2$	0.6	0.2	0.2
$U_3$	0.3	0.4	0.3

Output Probabilities (B)

	R	G	B
$U_1$	0.3	0.5	0.2
$U_2$	0.1	0.4	0.5
$U_3$	0.6	0.1	0.3

# Observations and states

	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$	$O_6$	$O_7$	$O_8$
OBS:	R	R	G	G	B	R	G	R
State:	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$	$S_8$

$S_i = U_1/U_2/U_3$ ; A particular state

S: State sequence

O: Observation sequence

$S^*$  = 'best' possible state (urn) sequence

Goal: Maximize  $P(S^* | O)$  by choosing 'best' S

- Goal: Maximize  $P(S|O)$  where  $S$  is the State Sequence and  $O$  is the Observation Sequence
  - $S^* = \text{argmax}_s(P(S|O))$

	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$	$O_6$	$O_7$	$O_8$
OBS:	R	R	G	G	B	R	G	R
State:	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$	$S_8$

$$P(S|O) = P(S_{1-8}|O_{1-8})$$

$$P(S|O) = P(S_1|O)P(S_2|S_1, O)P(S_3|S_2, O) \dots P(S_8|S_{1-7}, O)$$

**Markov Assumption:** a state depends only on the previous state

$$P(S|O) = P(S_1|O)P(S_2|S_1, O)P(S_3|S_2, O) \dots P(S_8|S_7, O)$$

## Baye's Theorem

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

$P(A)$ : Prior  $P(B|A)$ : Likelihood

$$\operatorname{argmax}_s P(S|O) = \operatorname{argmax}_x P(S)P(O|S)$$

## State Transitions Probability

$$P(S) = P(S_{1-8})$$

$$P(S) = P(S_1)P(S_2|S_1)P(S_3|S_{1-2})P(S_4|S_{1-3})\dots P(S_8|S_{1-7})$$

By Markov Assumption (k=1)

$$P(S) = P(S_1)P(S_2|S_1)P(S_3|S_2)P(S_4|S_3)\dots P(S_8|S_7)$$

## Observations Sequence Probability

$$P(O|S) =$$

$$P(O_1|S_{1-8})P(O_2|O_1, S_{1-8})P(O_3|O_{1-2}, S_{1-8}) \dots P(O_8|O_{1-7}, S_{1-8})$$

**Assumption that ball drawn depends only on the Urn Chosen**

$$P(O|S) = P(O_1|S_1)P(O_2|S_2)P(O_3|S_3) \dots P(O_8|S_8)$$

$$P(S|O) = P(S)P(O|S)$$

$$P(S|O) = P(S_1)P(S_2|S_1)P(S_3|S_2)P(S_4|S_3) \dots P(S_8|S_7)P(O_1|S_1)$$

$$P(O_2|S_2)P(O_3|S_3) \dots P(O_8|S_8)$$

	$O_0$	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$	$O_6$	$O_7$	$O_8$	
OBS:	$\epsilon$	R	R	G	G	B	R	G	R	
State:	$S_0$	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$	$S_8$	$S_9$

$$P(S) \cdot P(O|S)$$

$$= [P(O_0|S_0) \cdot P(S_1|S_0)]$$

States  $S_0$  and  $S_9$  is introduced  
as initial and final states

$$[P(O_1|S_1) \cdot P(S_2|S_1)]$$

$$[P(O_2|S_2) \cdot P(S_3|S_2)]$$

$$[P(O_3|S_3) \cdot P(S_4|S_3)]$$

$$[P(O_4|S_4) \cdot P(S_5|S_4)]$$

$$[P(O_5|S_5) \cdot P(S_6|S_5)]$$

$$[P(O_6|S_6) \cdot P(S_7|S_6)]$$

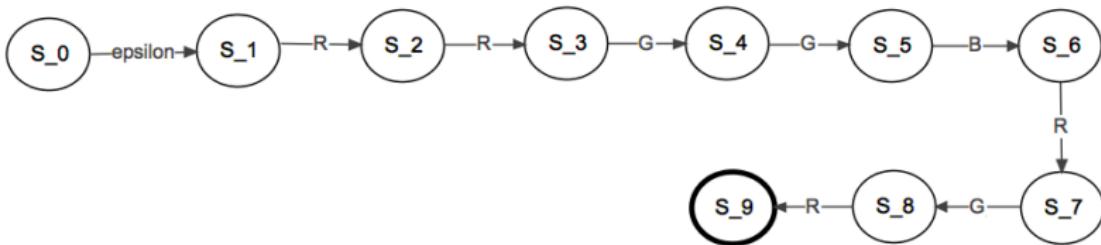
$$[P(O_7|S_7) \cdot P(S_8|S_7)]$$

$$[P(O_8|S_8) \cdot P(S_9|S_8)]$$

After  $S_8$  the next state is  $S_9$   
with probability 1, i.e.,  
 $P(S_9|S_8) == 1$

$O_0$  is  $\epsilon$ -transition

	$O_0$	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$	$O_6$	$O_7$	$O_8$	
OBS:	$\epsilon$	R	R	G	G	B	R	G	R	
State:	$S_0$	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$	$S_8$	$S_9$



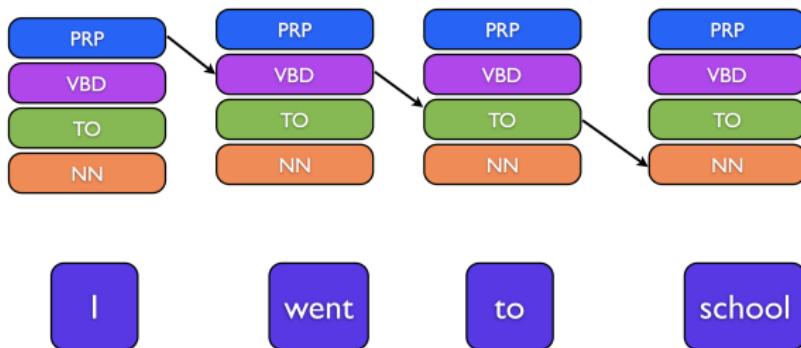
$$P(O_k|S_k) \cdot P(S_{k+1}|S_k) = P(S_k \xrightarrow{o} S_{k+1})$$

# Three problems of HMM

- **Problem 1 (Decoding)**: Given an observation sequence  $O$  and an HMM  $\lambda = (A, B)$ , discover the best hidden state sequence  $S$ .
- **Problem 2 (Computing Likelihood)**: Given an HMM  $\lambda = (A, B)$  and an observation sequence  $O$ , determine the likelihood  $P(O|\lambda)$ .
- **Problem 3 (Learning)** : Given an observation sequence  $O$  and the set of states in the HMM, learn the HMM parameters  $A$  and  $B$ .

- **Problem 1 (Decoding):** Given an observation sequence  $O$  and an HMM  $\lambda = (A, B)$ , discover the best hidden state sequence  $S$ .

## Why is it difficult?

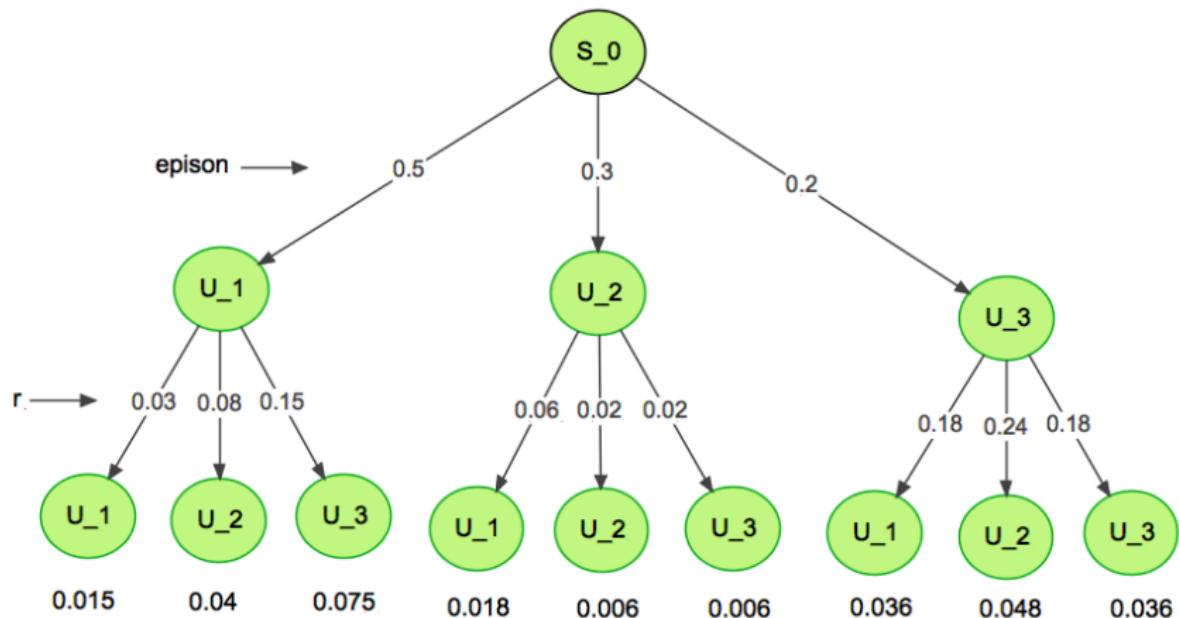


Even if there were only four POS tags, then this is just one of  $4 \times 4 \times 4 \times 4 = 256$  possible state sequences!

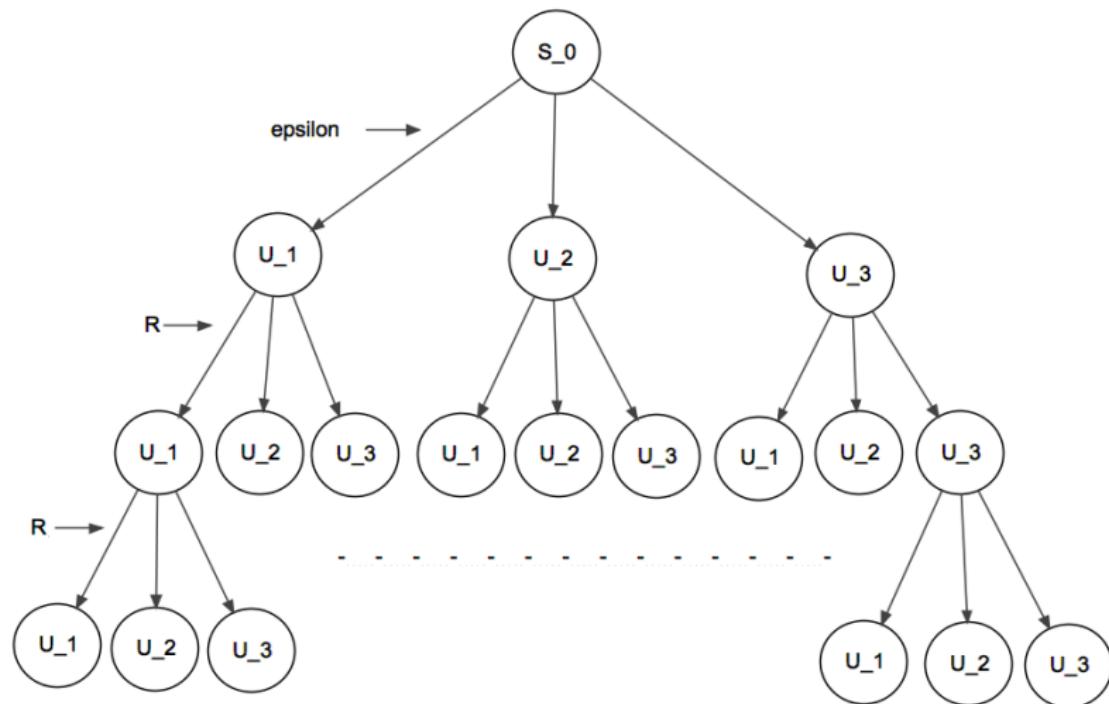
# Quiz 1

- If there are  $T$  number of POS tags, and you have a sentence with  $N$  number of words, how many different possible POS tag sequences can you get?

## Viterbi Algorithm for the Urn problem (first two symbols)



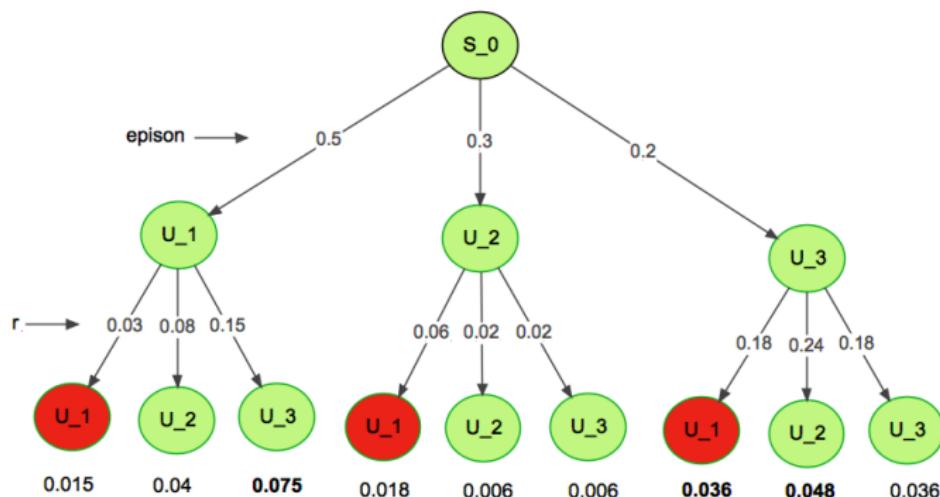
# HMM - Computational Complexity



# HMM - Computational Complexity

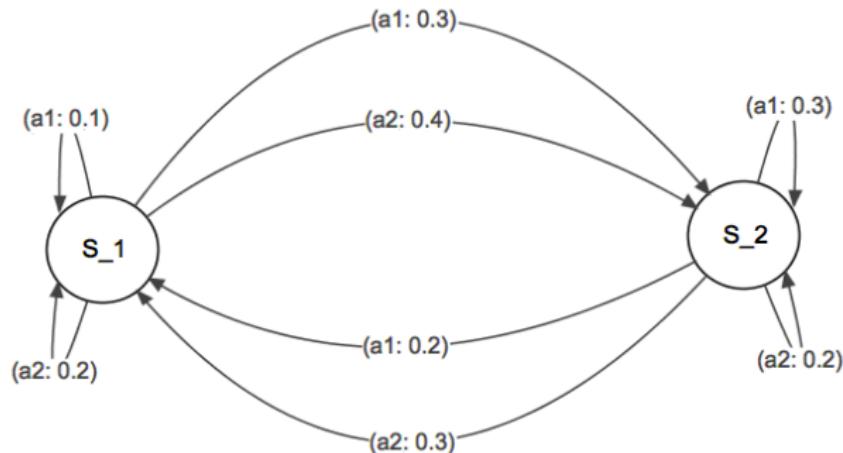
- if the tree is grown in this manner
  - RRGGBRGR - Observation Sequence length = 9 (including epsilon)
  - at each level multiply the node by 3
  - level 1 ( $\epsilon$ ) -  $3^1$ , at level 2 (R) -  $3^2$ , ...at level 9 (R) -  $3^9$  (nodes at leaf)
  - complexity without restriction =  $|S|^{|O|}$   
 $|S|$  = Number of States,  $|O|$  = length of the observation sequence

# Viterbi Algorithm for the Urn problem (first two symbols)

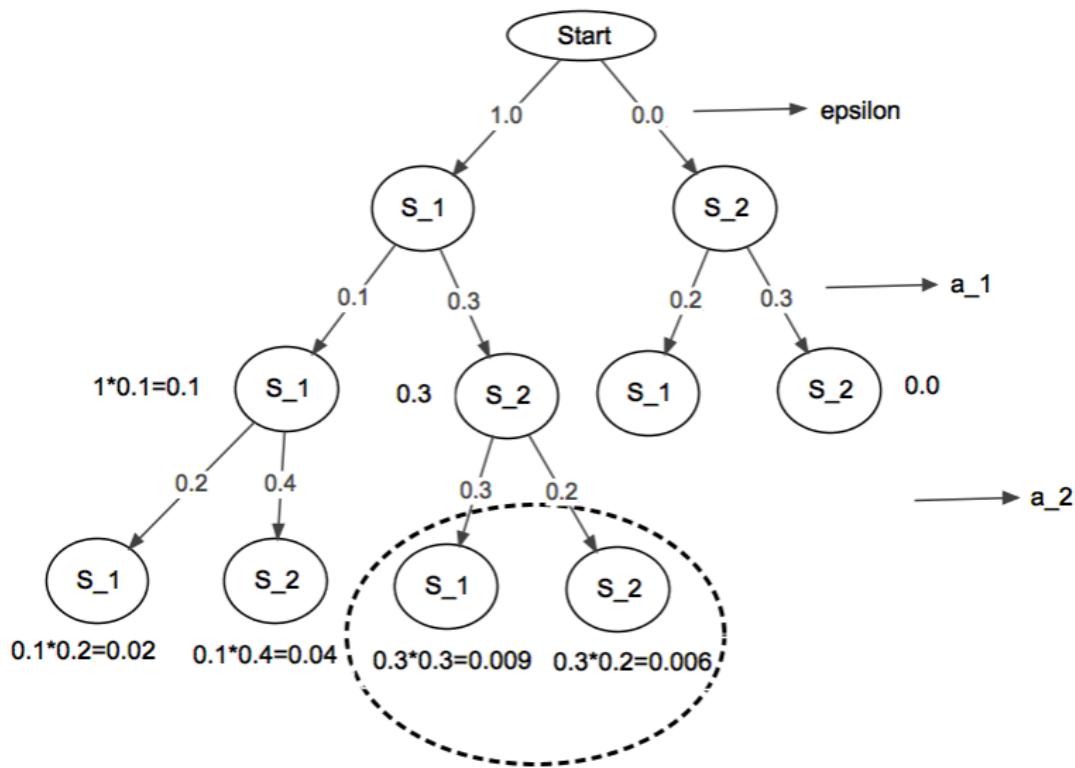


- At every stage, we only keep three nodes
- at the end of observation sequence - we have three nodes (total nodes -  $3 \times 8$ )
- complexity comes down from  $|S|^{|o|}$  to  $|S|.|o|$

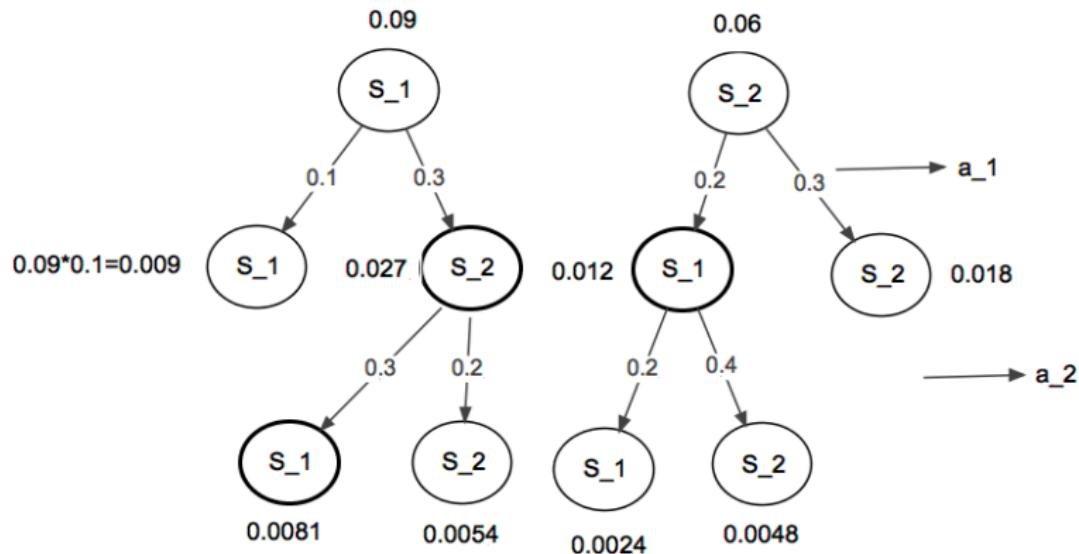
# Probabilistic FSM



# Probabilistic FSM (contd.)



# Probabilistic FSM (contd.)



# Tabular Representation of the Tree

	$\epsilon$	$a_1$	$a_2$	$a_1$	$a_2$
$S_1$	1.0	(1.0*0.1, 0.0*0.2) = (0.1, 0.0)	(0.02, <b>0.09</b> )	(0.009, <b>0.012</b> )	(0.0024, <b>0.0081</b> )
$S_2$	0.0	(1.0*0.3, 0.0*0.3) = (0.3, 0.0)	(0.04, <b>0.06</b> )	(0.027, 0.018)	(0.0048, 0.0054)

- Number of columns = length of observation sequence +1 ( $\epsilon$  )
- Rows - ending state

# HMM - Definition

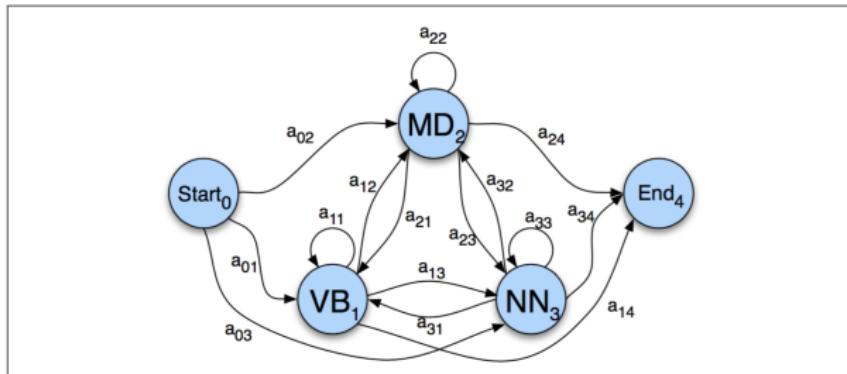
$Q = q_1 q_2 \dots q_N$	a set of <b>states</b>
$A = a_{01} a_{02} \dots a_{n1} \dots a_{nn}$	a <b>transition probability matrix</b> $A$ , each $a_{ij}$ representing the probability of moving from state $i$ to state $j$ , s.t. $\sum_{j=1}^n a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \dots o_N$	a set of <b>observations</b> , each one drawn from a vocabulary $V = v_1, v_2, \dots, v_V$ .
$B = b_i(o_t)$	A set of <b>observation likelihoods</b> : also called <b>emission probabilities</b> , each expressing the probability of an observation $o_t$ being generated from a state $i$ .
$q_0, q_{end}$	a special <b>start and end state</b> which are not associated with observation.

Markov Assumption:  $P(q-1|q_1, \dots, q_{i-1}) = P(q_i|q_{i-1})$

Output Independence Assumption:

$$P(o_i|q_1, \dots, q_i, \dots, q_n, o_1, \dots, o_i, \dots, o_n) = P(o_i|q_i)$$

# HMM - POS Tagging



**Figure:** Markov chain corresponding to the hidden states of HMM. The transition probabilities  $A$  are used to compute the prior probability.

# HMM - POS Tagging

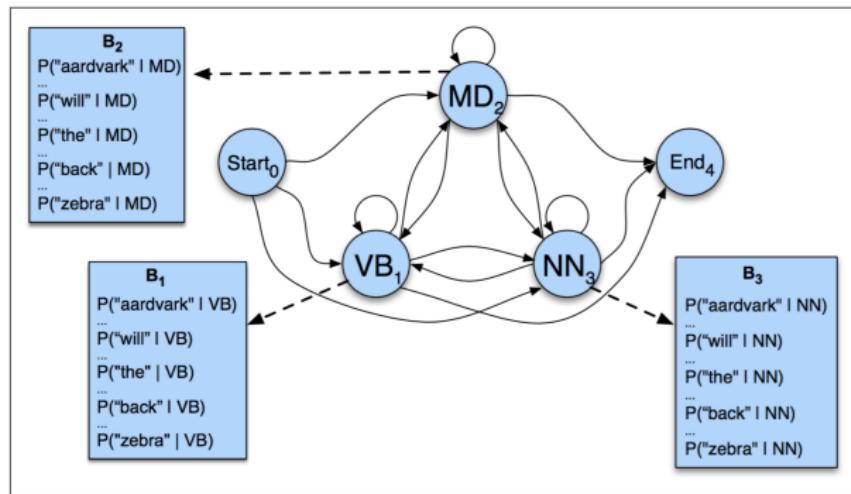


Figure: Observation likelihoods  $B$  for the HMM.

# HMM - POS Tagging

Goal: choose the most probable tag sequence given the observation sequence of  $n$  words  $\hat{w}_1^n$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

Using Bayes' rule

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$$

Simplifying further by dropping the denominator

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(w_1^n | t_1^n) P(t_1^n)$$

# HMM - POS Tagging

HMM makes two further assumptions:

- ① probability of a word depends only on its tag and is independent of neighbouring words and tags

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^n P(w_i | t_i)$$

- ② probability of a word depends only on its tag and is independent of neighbouring words and tags

$$P(t_1^n) \approx \prod_{i=1}^n P(t_i | t_{i-1})$$

Using these simplifications:

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n) \approx \operatorname{argmax}_{t_1^n} \underbrace{\prod_{i=1}^n P(w_i | t_i)}_{\text{emission transition}} \underbrace{P(t_i | t_{i-1})}_{\text{transition}}$$

# Viterbi Algorithm - Pseudocode

```

function VITERBI(observations of len  $T$ ,state-graph) returns best-path

    num-states  $\leftarrow$  NUM-OF-STATES(state-graph)
    Create a path probability matrix viterbi[num-states+2, $T+2$ ]
    viterbi[0,0]  $\leftarrow$  1.0
    for each time step  $t$  from 1 to  $T$  do
        for each state  $s$  from 1 to num-states do
            viterbi[s,t]  $\leftarrow$   $\max_{1 \leq s' \leq \text{num-states}}$  viterbi[ $s',t-1$ ] *  $a_{s',s}$  *  $b_s(o_t)$ 
            backpointer[s,t]  $\leftarrow$   $\operatorname{argmax}_{1 \leq s' \leq \text{num-states}}$  viterbi[ $s',t-1$ ] *  $a_{s',s}$ 
    Backtrace from highest probability state in final column of viterbi[] and return path

```

**Figure 6.10** Viterbi algorithm for finding optimal sequence of tags. Given an observation sequence and an HMM  $\lambda = (A, B)$ , the algorithm returns the state-path through the HMM which assigns maximum likelihood to the observation sequence. Note that states 0 and  $N+1$  are non-emitting *start* and *end* states.

# POS Tagging - Example [Jurafsky and James, 2000]

- Janet will back the bill
- Janet/NNP will/MD back/VB the/DT bill/NN

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

# POS Tagging - Example

- Janet will back the bill
- Janet/NNP will/MD back/VB the/DT bill/NN

	<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	0.000097	0
<b>NN</b>	0	0.000200	0.000223	0.000006	0.002337
<b>RB</b>	0	0	0.010446	0	0
<b>DT</b>	0	0	0	0.506099	0

Figure: Observation likelihoods computed from WSJ corpus (without smoothing)

# POS Tagging - Example

- Janet will back the bill
- Janet/NNP will/MD back/VB the/DT bill/NN

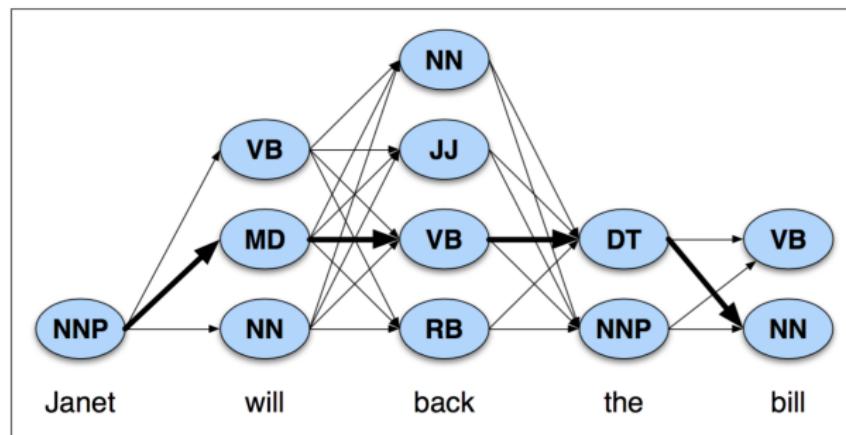
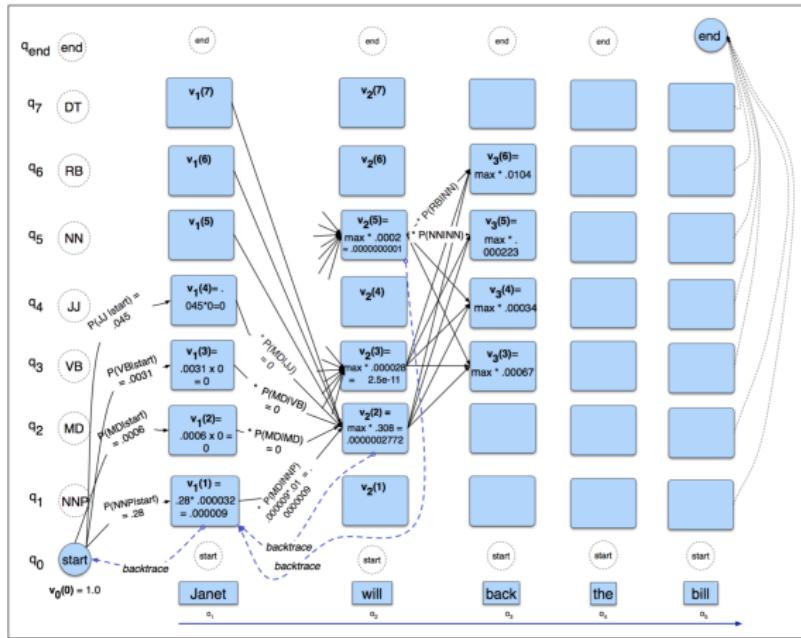


Figure: Schematic diagram showing possible POS tag sequences

# POS Tagging - Example

- Janet will back the bill
- Janet/NNP will/MD back/VB the/DT bill/NN



-  Cavnar, W. B., Trenkle, J. M., et al. (1994).  
N-gram-based text categorization.  
*Ann Arbor MI*, 48113(2):161–175.
-  Jurafsky, D. and James, H. (2000).  
Speech and language processing an introduction to natural language processing, computational linguistics, and speech.
-  Kučera, H. and Francis, W. N. (1967).  
*Computational analysis of present-day American English*.  
Dartmouth Publishing Group.
-  Padró, M. and Padró, L. (2004).  
Comparing methods for language identification.  
*Procesamiento del lenguaje natural*, 33:155–162.
-  Yang, X. and Liang, W. (2010).  
An n-gram-and-wikipedia joint approach to natural language identification.  
In *Universal Communication Symposium (IUCS), 2010 4th International*, pages 332–339. IEEE.