Part of Speech Tagging

FPT Education
FPT UNIVERSITY

- Outline
  - What is part of speech tagging?
  - Markov chains
  - Hidden Markov models
  - Viterbi algorithm
  - Example

What is part of speech?

FPT Education
FPT UNIVERSITY

Why	not	learn	something	?
adverb	adverb	verb	noun	punctuation mark, sentence closer

Part of speech (POS) tagging

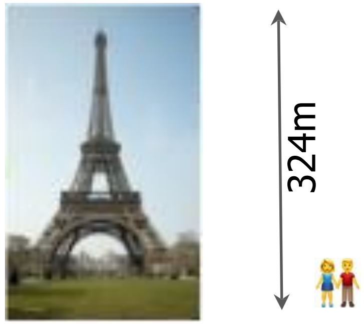
FPT Education
FPT UNIVERSITY

- Part of speech tags:

lexical term	tag	example
noun	NN	something , nothing
verb	VB	learn, study
determiner	DT	the, a
w-adverb	WRB	why, where
...	...	

Why not learn something ?
WRB RB VB NN .

- Applications:
  - Named entities
  - Co-reference resolution
  - Speech recognition



Co-reference resolution

Speech recognition

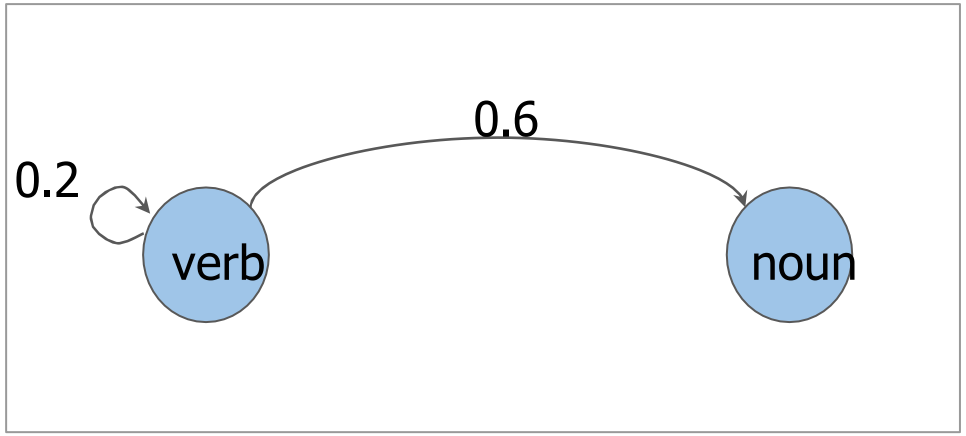
Markov Chains

- Example
  - whether the following word in the sentence is a noun, a verb, or some other parts of speech

Why not learn ...

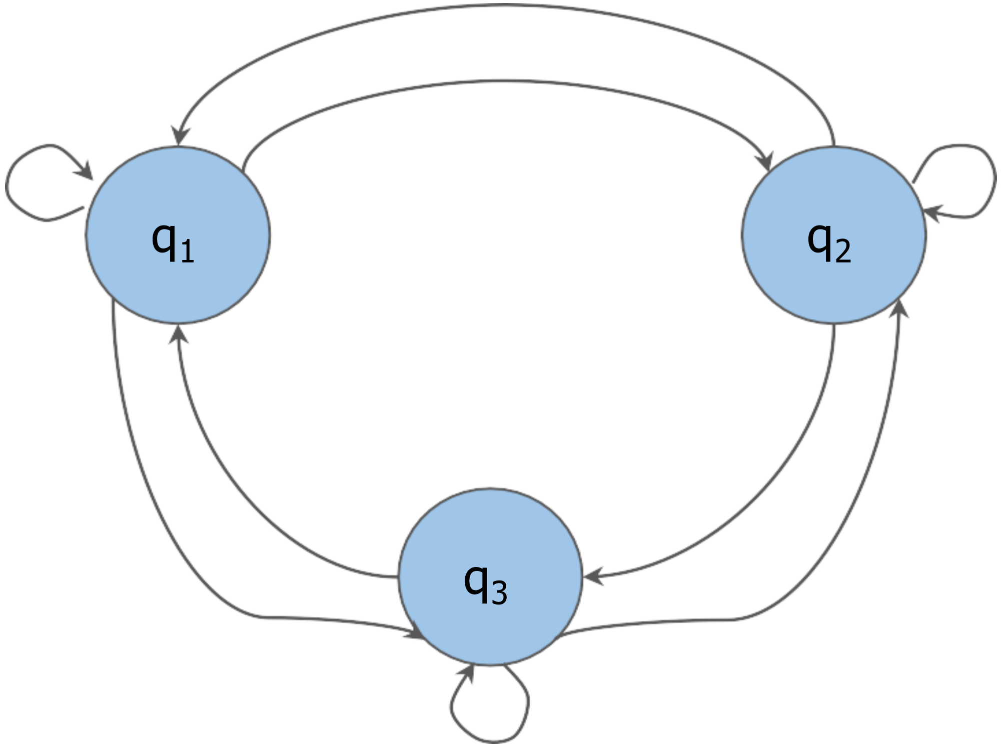
verb verb?
noun?
...?

- Visual Representation
  - The likelihood of the next words part of speech tag in a sentence tends to depend on the part of speech tag of the previous word



Markov Chain

- Markov chain can be depicted as a directed graph
  - a graph is a kind of data structure that is visually represented as a set of circles connected by lines.
- The circles of the graph represents states of our model
- The arrows from state s1 to s2 represents the transition probabilities, the likelihood to move from s1 to s2

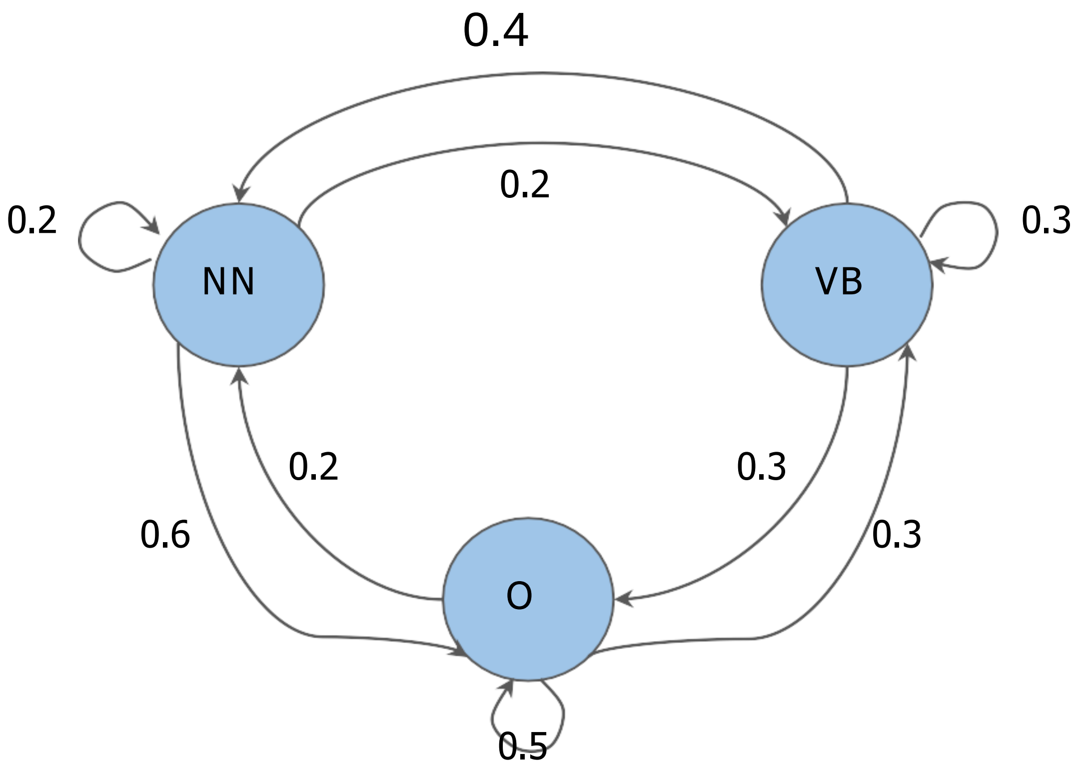


$$Q = \{q_1, q_2, q_3\}$$

Markov Chains and POS Tags

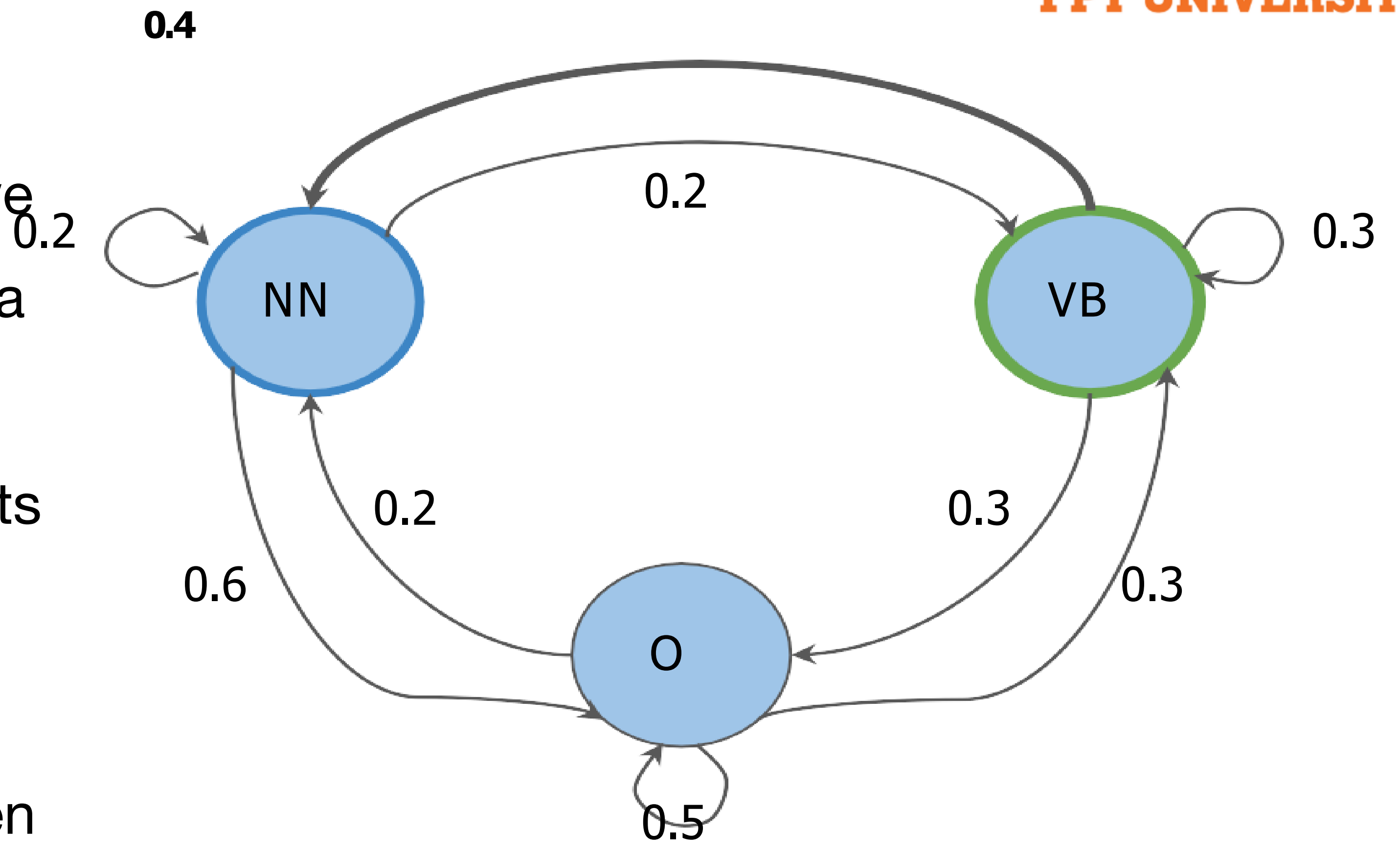
- Transition probabilities

Why not learn something ?



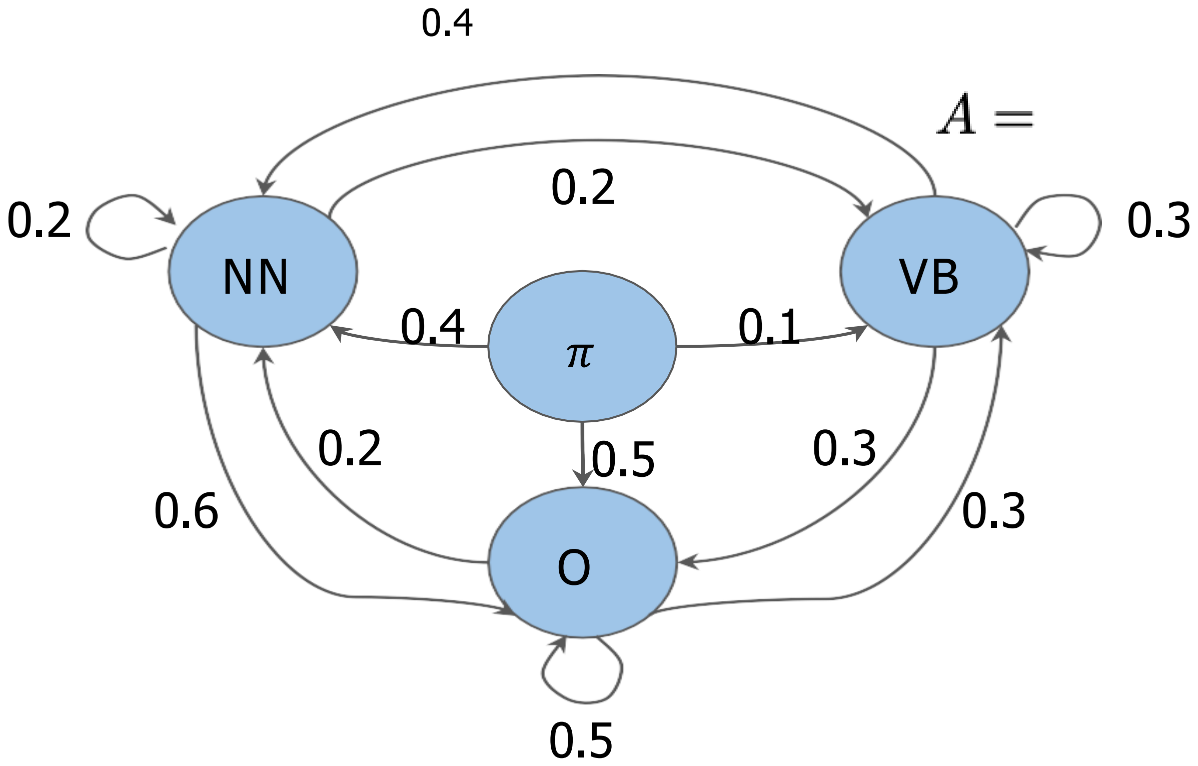
Transition probabilities

- A sequence of words with associated parts of speech tags we can represent that sequence with a graph
- The parts of speech tags are events that can occur depicted by the states of the model graph.
- The weights on the arrows between the states define the probability of going from one state to another



Why not learn something ?

Initial probabilities



Why not learn something ?

NN?
VB?
O?

	NN	VB	O
π (initial)	0.4	0.1	0.5
NN (noun)	0.2	0.2	0.6
VB (verb)	0.4	0.3	0.3
O (other)	0.2	0.3	0.5

- The model graph can be represented as a Transition matrix with dimension n+1 by n when no previous state, we introduce an initial state π.
- The sum of all transition from a state should always be 1

Transition table and matrix

$A =$

	NN	VB	O
π (initial)	0.4	0.1	0.5
NN (noun)	0.2	0.2	0.6
VB (verb)	0.4	0.3	0.3
O (other)	0.2	0.3	0.5

$$A = \begin{pmatrix} 0.4 & 0.1 & 0.5 \\ 0.2 & 0.2 & 0.6 \\ 0.4 & 0.3 & 0.3 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}$$

States

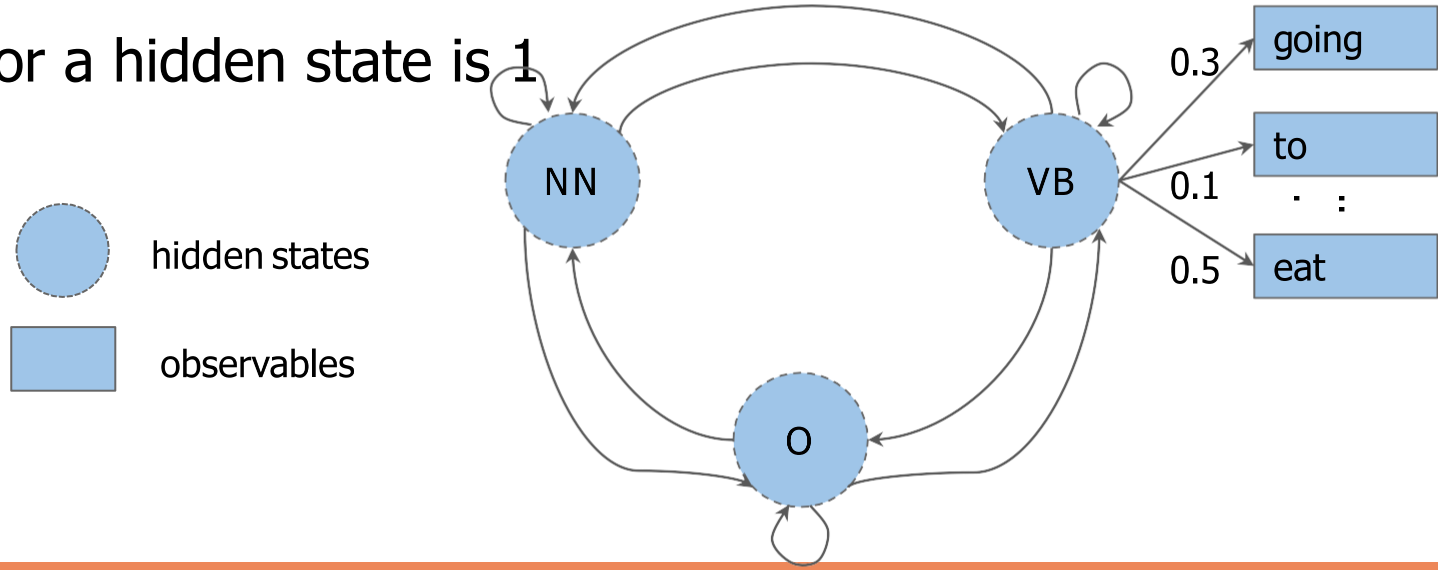
$$Q = \{q_1, \ldots, q_N\}$$

Transition matrix

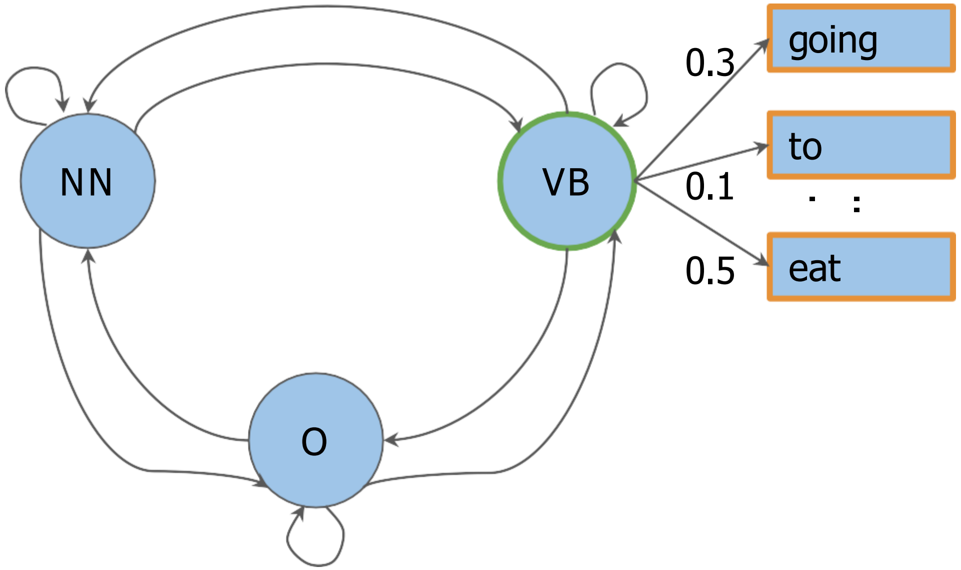
$$A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{N+1,1} & \cdots & a_{N+1,N} \end{pmatrix}$$

Hidden Markov Models

- The hidden Markov model implies that states are hidden or not directly observable
- The hidden Markov model have a transition probability matrix A of dimensions (N+1,N) where N is number of hidden states
- The hidden Markov model have emission probabilities matrix B describe the transition from the hidden states to the observables(the words of your corpus)
- the row sum of emission probabity for a hidden state is 1



The emission matrix



$B =$

	going	to	eat	...
NN (noun)	0.5	0.1	0.02	
VB (verb)	0.3	0.1	0.5	
O (other)	0.3	0.5	0.68	

$$\sum_{j=1}^{V} b_{ij} = 1$$

He lay on his back.

I'll be back.

States

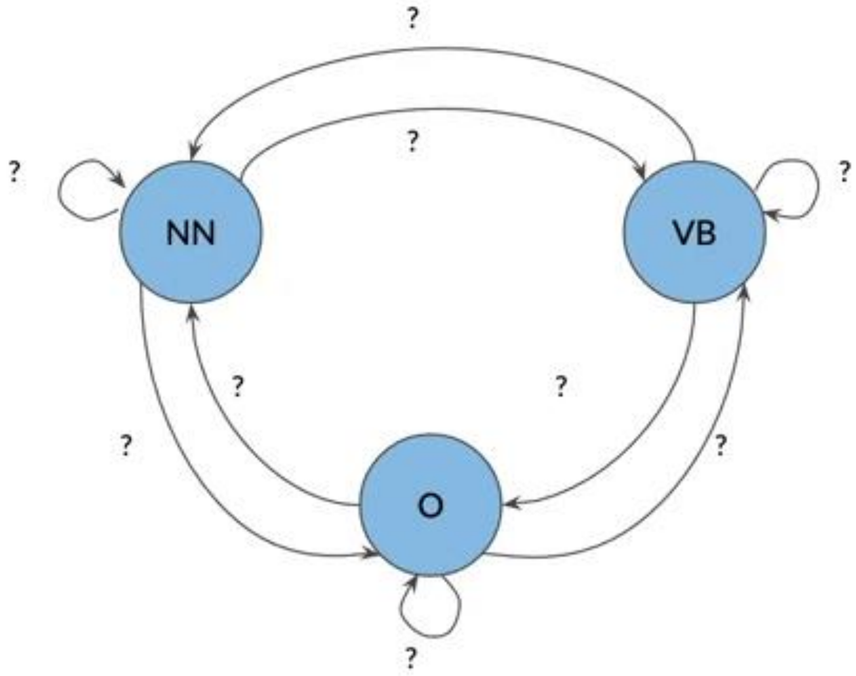
Transition matrix

Emission matrix

$$Q = \{q_1, \ldots, q_N\} \quad A = \begin{pmatrix} a_{1,1} & \dots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{N+1,1} & \dots & a_{N+1,N} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & \dots & b_{1V} \\ \vdots & \ddots & \vdots \\ b_{N1} & \dots & b_{NV} \end{pmatrix}$$

The Transition Matrix

- Transition matrix holds all the transition probabilities between states of the Markov model
  - $C(t_{i-1},t_i)$ count all occurrences of tag pairs in your training corpus
  - $C(t_{i-1},t_j)$ count all occurrences of tag t_{i-1}



1. Count occurrences of tag pairs

$$C(t_{i-1}, t_i)$$

2. Calculate probabilities using the counts

$$\boxed{P(t_i|t_{i-1})} = \frac{C(t_{i-1}, t_i)}{\sum_{j=1}^{N} C(t_{i-1}, t_j)}$$

The Transition Matrix

e apply

$A =$

	NN	VB	O
π	1	0	2
NN (noun)	0	0	6
VB (verb)	0	0	0
O (other)	6	0	$\boxed{8}$

<s> in a station of the metro

<s> the apparition of these faces in the crowd :

<s> petals on a wet , black bough .

Ezra Pound – 1913

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i) + \epsilon}{\sum_{j=1}^{N} C(t_{i-1}, t_j) + N * \epsilon}$$

The Emission Matrix

- Count the co-occurrences of a part of speech tag with a specific word

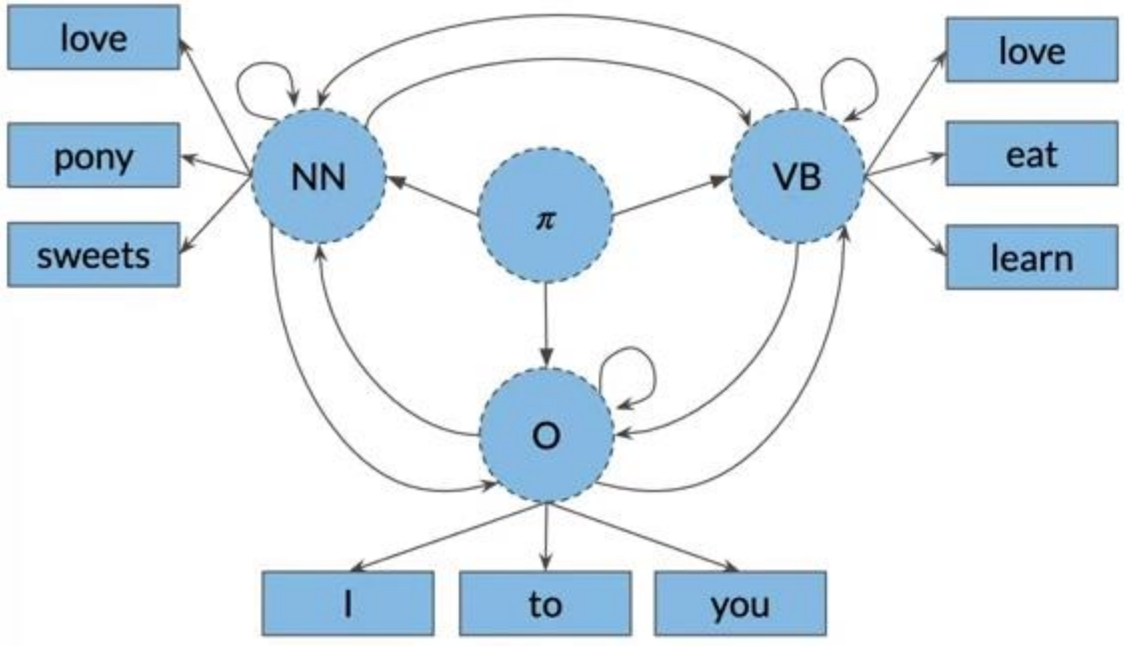
$B =$

	in	a	...
NN (noun)	0	...	...
VB (verb)	0	...	...
O (other)	2	...	...

$$P(w_i|t_i) = \frac{C(t_i, w_i) + \epsilon}{\sum_{j=1}^{V} C(t_i, w_j) + N * \epsilon} = \frac{C(t_i, w_i) + \epsilon}{C(t_i) + N * \epsilon}$$

The Viterbi Algorithm

- The Viterbi algorithm is actually a graph algorithm
- The goal is to to find the sequence of hidden states or parts of speech tags that have the highest probability for a sequence



Why not learn something ?

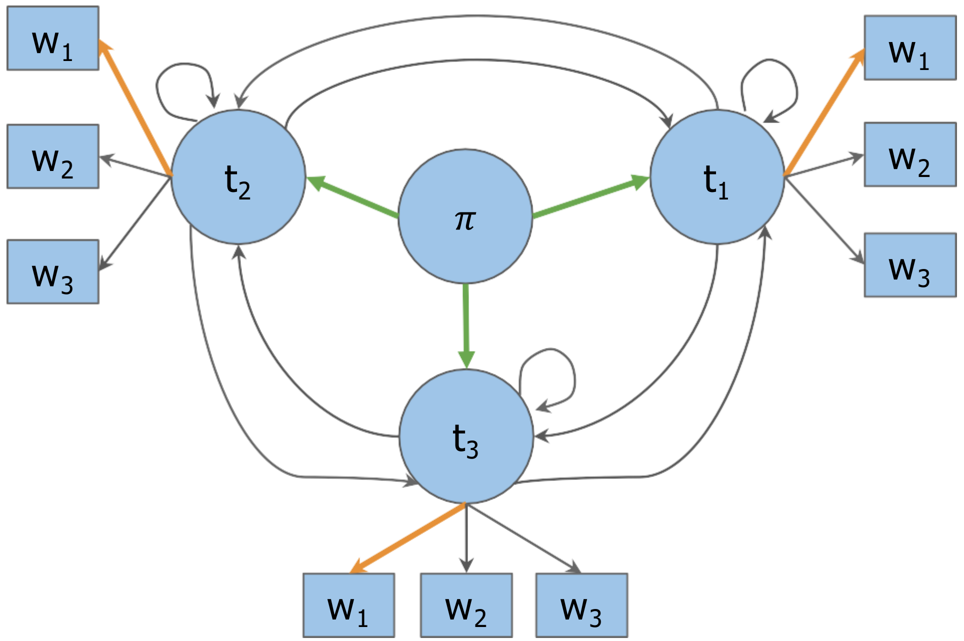
? ? ? ? ?

<s> I love to learn

The Viterbi Algorithm

- The algorithm can be split into three main steps: the initialization step, the forward pass, and the backward pass.
- Given your transition and emission probabilities, we first populates and then use the auxiliary matrices C and D
  - matrix C holds the intermediate optimal probabilities
  - matrix D holds the indices of the visited states as we are traversing the model graph to find the most likely sequence of parts of speech tags for the given sequence of words, W_1 all the way to W_k.
  - C and D matrix have n rows (number of parts of speech tags) and k comlumns (number of words in the given sequence)

Initialization step



$C =$

	w_1	w_2	…	w_K
t_1	$c_{1,1}$			
…				
t_N	$c_{N,1}$			

$$c_{i,1} = \boxed{\pi_i} * \boxed{b_{i,cindex(w_1)}}$$
$$= a_{1,i} * b_{i,cindex(w_1)}$$

$D =$

	w_1	w_2	…	w_K
t_1	$d_{1,1}$			
…				
t_N	$d_{N,1}$			

- The initialization step is one of three steps to populate the auxiliary matrices C and D is populated.

$$d_{i,1} = 0$$

Forward Pass

- For the C matrix, the entries are calculated by this formula:

$C =$

	w_1	w_2	...	w_K
t_1	$c_{1,1}$	$c_{1,2}$		$c_{1,K}$
...				
t_N	$c_{N,1}$	$c_{N,2}$		$c_{N,K}$

$$c_{i,j} = \max_k c_{k,j-1} * a_{k,i} * b_{i,cindex(w_j)}$$

$$d_{i,j} = \operatorname*{argmax}_k c_{k,j-1} * a_{k,i} * b_{i,cindex(w_j)}$$

- For matrix D, save the k, which maximizes the entry in $c_{i,j}$.

Backward Pass

- The backward pass help retrieve the most likely sequence of parts of speech tags for your given sequence of words.
- First calculate the index of the entry, Ci,K, with the highest probability in the last column of C represents the last hidden state we traversed when we observe the word wi
- Use this index to traverse back through the matrix D to reconstruct the sequence of parts of speech tags multiply many very small numbers like probabilities leads to numerical issues
- Use log probabilities instead where numbers are summed instead of multiplied.

$$c_{i,j} = \max_k c_{k,j-1} * a_{k,i} * b_{i,cindex(w_j)}$$

$$\downarrow$$

$$log(c_{i,j}) = \max_k log(c_{k,j-1}) + log(a_{k,i}) + log(b_{i,cindex(w_j)})$$

Summary

1. From word sequence to POS tag sequence
2. Viterbi algorithm
3. Log probabilities