UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Natural Language Processing

Part-Of-Speech Tagging

LESSONS 17-18

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

Some of the slides are taken from DeepLearning.AI and from the slides accompanying the textbook by D. Jurafsky

What're we going to learn

- What part of speech tagging is
- Markov chains
- Hidden Markov Models
- Viterbi Algorithm

Part of Speech

- From the earliest linguistic traditions (Yaska and Panini 5th C. BCE, Aristotle 4th C. BCE), the idea that words can be classified into grammatical categories
  - part of speech, word classes, POS, POS tags
- 8 parts of speech attributed to Dionysius Thrax of Alexandria (c. 1st C. BCE):
  - noun, verb, pronoun, preposition, adverb, conjunction, participle, article
  - These categories are relevant for NLP today

Part of Speech

- Part-of-speech (POS) tags are lexical categories such as noun, verb adjective, pronoun, preposition, article, etc.
  - Also known as word classes or morphological classes
- We call a tagset the set of all POS tags used by some model
- Different languages, different grammatical theories, and different applications may require different tagsets
- POS is a clue to sentence structure and meaning

Two classes of words: Open vs Closed

- POS tags fall into two broad categories: closed class and open class
- Closed-class words usually include function words: Short, frequent words with grammatical function
  - determiners: *a, an, the*
  - pronouns: *she, he, I*
  - prepositions: *on, under, over, near, by, …*
  - Relatively fixed membership
    - New words in this class are rarely coined
- Open-class words usually consist of content words: Nouns (including proper nouns), verbs, adjectives, and adverbs
  - Plus interjections: oh, ouch, uh-huh, yes, hello
  - New words appear almost always in this class
    - E.g., iPhone (noun), to fax (verb)

Open class ("content") words

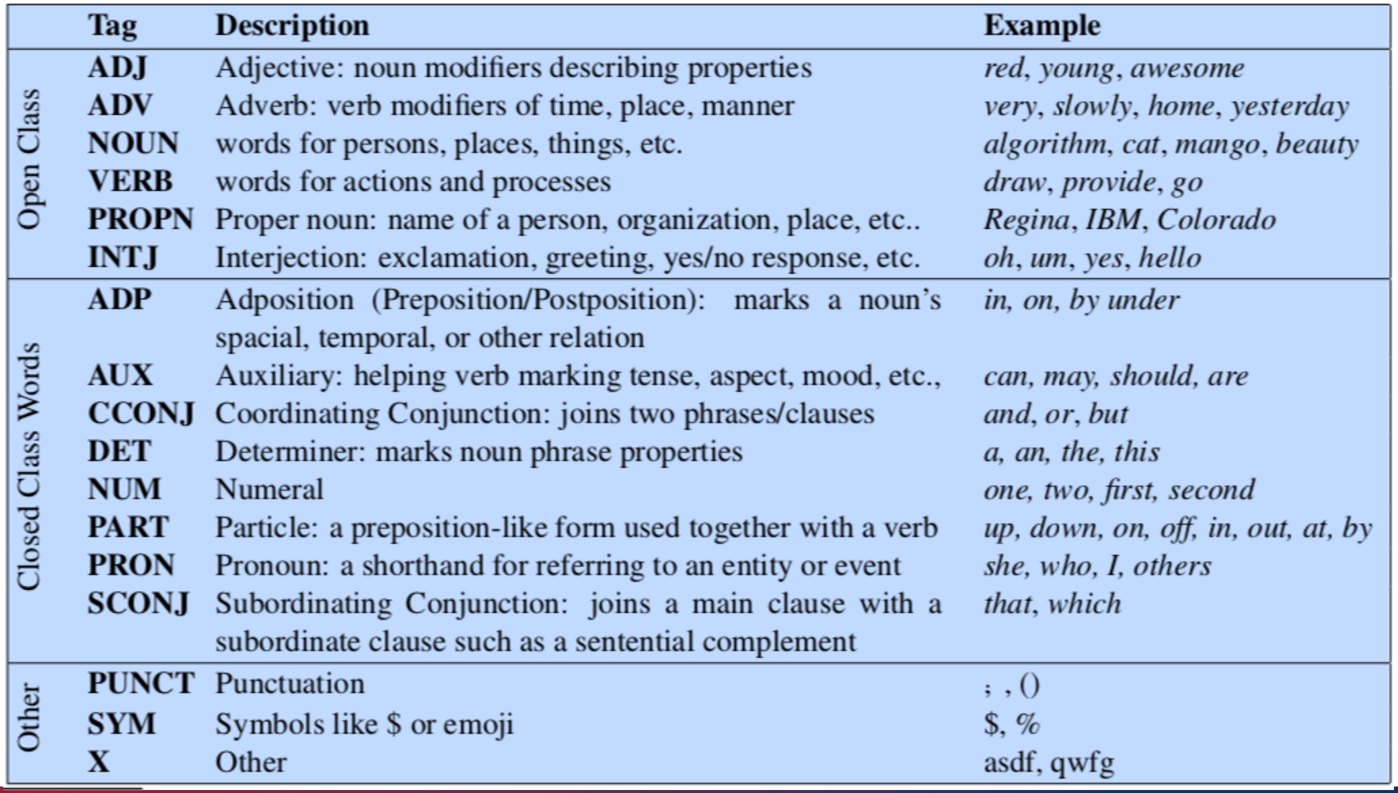
- Nouns
  - Proper: *Janet* *Italy*
  - Common: *cat, cats* *mango*
- Verbs
  - Main: *eat* *went*
  - Auxiliary: *can* *had*
- Adjectives *old green tasty*
- Adverbs *slowly yesterday*
- Numbers *122,312* *one*
- Interjections *Ow hello*
- *… more*

Closed class ("function")

- Determiners *the some*
- Conjunctions *and or*
- Pronouns *they its*
- Prepositions *to with*
- Particles *off up*
- *… more*

The Universal Dependencies (UD) tagset

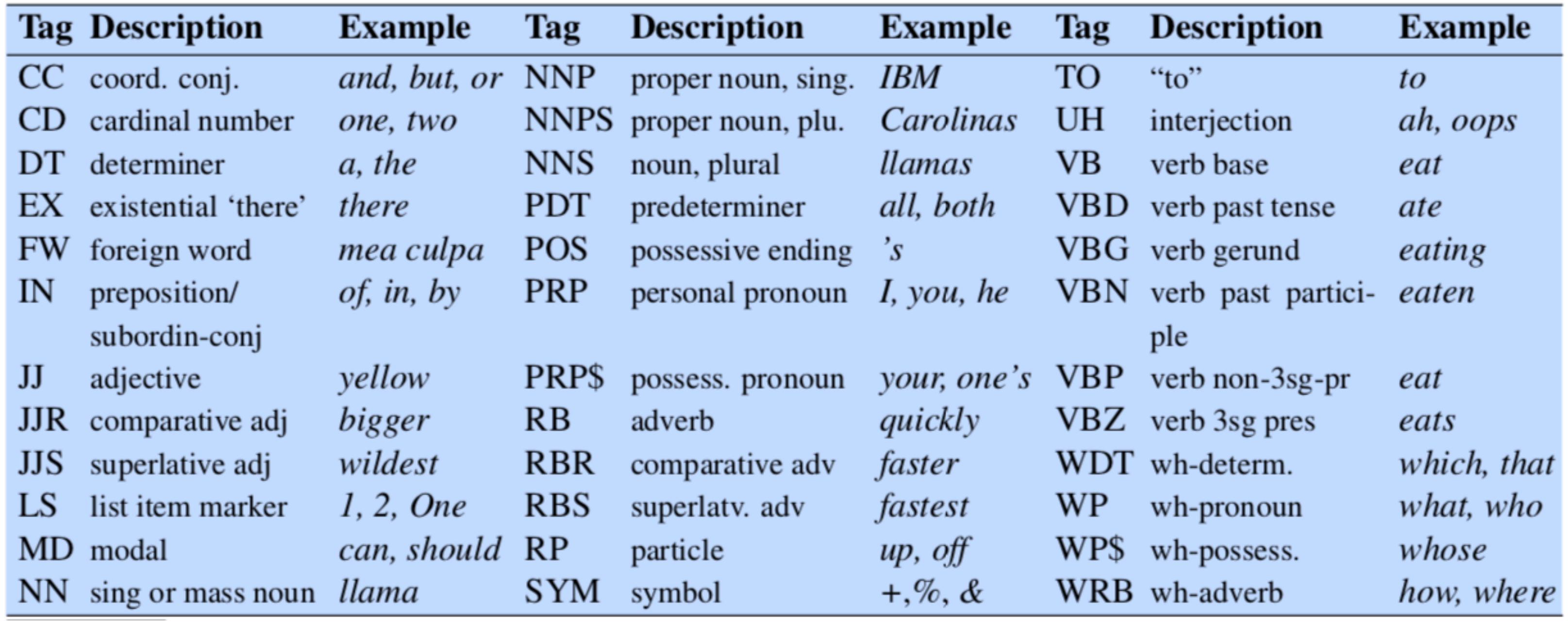
- The UD tagset contains 17 tags
  - POS tagged corpora for 92 languages



	Tag	Description	Example
Open Class	ADJ	Adjective: noun modifiers describing properties	*red, young, awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very, slowly, home, yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm, cat, mango, beauty*
	VERB	words for actions and processes	*draw, provide, go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina, IBM, Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh, um, yes, hello*
Closed Class Words	ADP	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and, or, but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that, which*
Other	PUNCT	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

The English-specific Penn Treebank tagset

- The English-specific Penn Treebank (PTB) tagset is also very popular
  - it contains 45 tags



Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	“to”	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential ‘there’	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*’s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past partici- ple	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one’s*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	*+,%, &*	WRB	wh-adverb	*how, where*

Example: Tagged English sentences

- *There*/PRO *were*/VERB *70*/NUM *children*/NOUN *there*/ADV *.*/PUNC

- *Preliminary*/ADJ *findings*/NOUN *were*/AUX *reported*/VERB *in*/ADP *today*/NOUN *'s*/PART *New*/PROPN *England*/PROPN *Journal*/PROPN *of*/ADP *Medicine*/PROPN

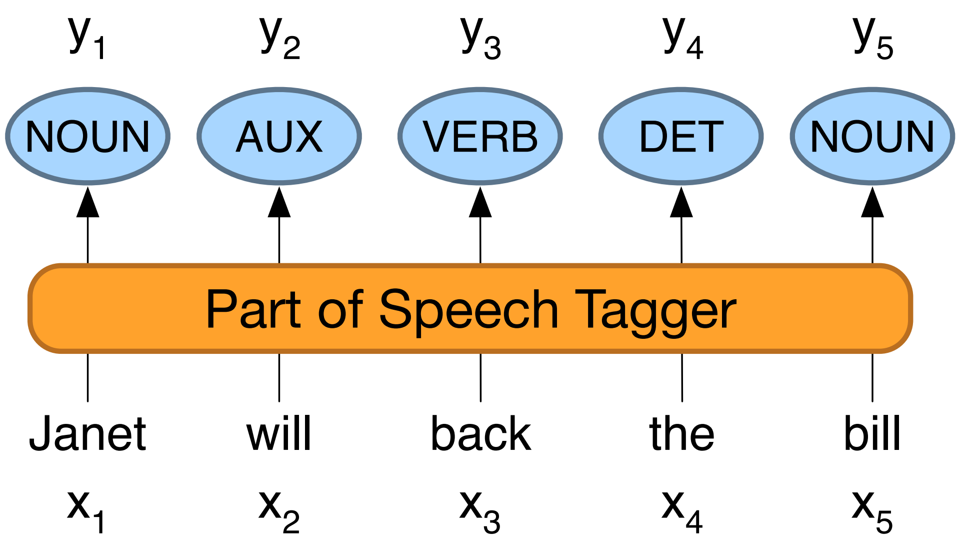
- Notice that the same word can have different tags in different sentences or within the same sentence
  - we need also to make decisions about segmentation
    - punctuation or the English 's is often segmented off as a particular

Part-of-Speech Tagging

- Assigning a proper (unique) part-of-speech tag to each word in a sentence
  - POS tagging must be done in the context of an entire sentence, based on the grammatical relationship of a word with its neighboring words
  - This is an instance of a more general task called sequence labeling
- Tagging is a disambiguation task
  - Words are ambiguous
    - depending on the context in which they appear, they may have different tags
  - Example:
    - The word 'book' can be tagged as VERB or as a NOUN
    - *Book/*VERB *that flight*
    - *Hand me that book/*NOUN
  - The goal is to find the correct tag for the situation

Part of speech tagging

- The input is a sequence $x_1, x_2, \ldots, x_n$ of (tokenized) words, and the output is a sequence $y_1, y_2, \ldots, y_n$ of tags, with y_i the tag assigned to x_i
- We assume the tagset is fixed, not part of the input



Part of speech tagging

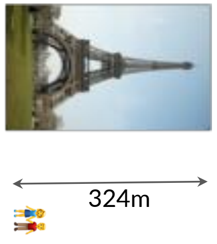
- In POS tagging we need to output a whole sequence of tags y_1, y_2, . . . , y_n for the input string, not just a category
- POS tagging is therefore a structured prediction task, not a classification task
- The number of output structures can be exponentially large in the length of the input, which makes structured prediction more challenging than classification

Why POS Tagging?

- Can be useful for other NLP tasks
  - Parsing: POS tagging can improve syntactic parsing
  - MT: reordering of adjectives and nouns (say from Spanish to English)
  - Sentiment or affective tasks: may want to distinguish adjectives or other POS
  - Text-to-speech (how do we pronounce "lead" or "object"?)
- Or linguistic or language-analytic computational tasks
  - Need to control for POS when studying linguistic change like the creation of new words, or meaning shift
  - Or control for POS in measuring meaning similarity or difference

Named entities

Co-reference resolution



Speech recognition

How difficult is POS tagging?

- Roughly 15% of word types are ambiguous
  - Hence 85% of word types are unambiguous
  - Janet is always PROPN, hesitantly is always ADV
- But those 15% tend to be very common, so ~60% of word tokens are ambiguous
- Example: *back*
  - *earnings growth took a back*/ADJ *seat*
  - *a small building in the back*/NOUN
  - *a clear majority of senators back*/VERB *the bill*
  - *enable the country to buy back*/PART *debt*
  - *I was twenty-one back*/ADV *then*

Evaluation

- The accuracy of a POS tagger is the percentage of test set tags that match human gold labels
- Human ceiling
  - How often do human annotators agree on the same tag? For PTB this is around 97%
- Accuracies over 97% have been reported across several languages, using the UD tagset

POS tagging performance in English

- How many tags are correct? (Tag accuracy)
  - About 97%
    - Hasn't changed in the last 10+ years
    - HMMs, CRFs, BERT perform similarly
    - Human accuracy about the same
- But baseline is 92%!
  - Baseline is performance of stupidest possible method
    - "Most frequent class baseline" is an important baseline for many tasks
      - Tag every word with its most frequent tag
      - (and tag unknown words as nouns)
  - Partly easy because
    - Many words are unambiguous

Sources of information for POS tagging

- Part of speech taggers generally make use of three sources of information

```
Janet will back the bill
```

AUX/NOUN/VERB? NOUN/VERB?

- Prior probabilities of word/tag
  - "will" is usually an AUX
- Identity of neighboring words
  - "the" means the next word is probably not a verb
- Morphology and wordshape:

• Prefixes	unable:	un- → ADJ
• Suffixes	importantly:	-ly → ADV
• Capitalization	Janet:	CAP → PROPN

Standard algorithms for POS tagging

- Supervised Machine Learning Algorithms:
  - Hidden Markov Models
  - Conditional Random Fields (CRF)/ Maximum Entropy Markov Models (MEMM)
  - Neural sequence models (RNNs or Transformers)
  - Large Language Models (like BERT), finetuned
- All required a hand-labeled training set, all about equal performance (97% on English)
  - All make use of information sources we discussed
  - Via human-created features: HMMs and CRFs
  - Via representation learning: Neural LMs

Markov chains

Markov Chains

- The HMM is based on augmenting the Markov chain model
- A Markov chain is a model that expresses the probabilities of sequences of random variables, the states, taking values from some set
  - Words, tags, or symbols representing anything, e.g., the weather
- A Markov chain assumes that if we want to predict the future in the sequence, all that matters is the current state

POS tagging

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

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**	**.**

Markov chains

- Example

Why not learn ▮ ...

verb verb? noun? ... ?

Why not learn ▮ ...

verb → **noun?** verb? ... ?

POS dependencies

Why not learn swimming?

verb **noun**

The most likely word

Why not learn swim?

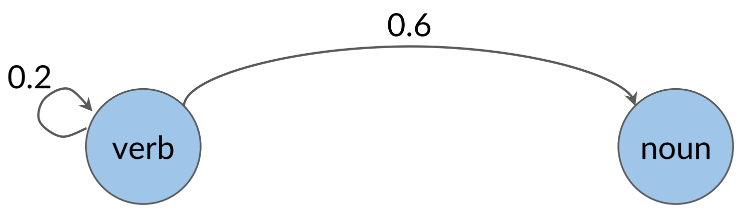
verb verb

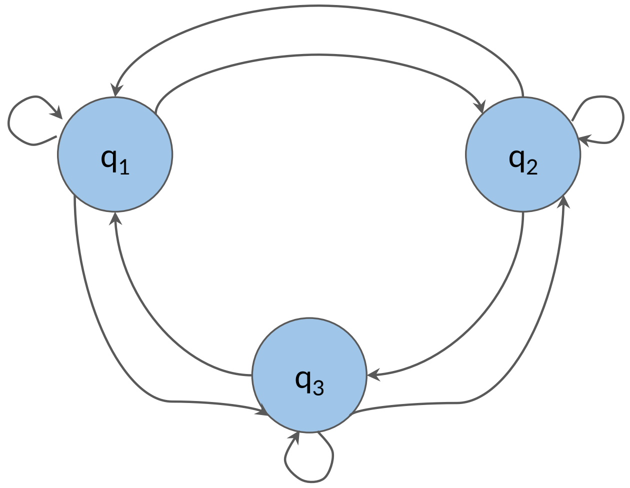
The less likely word

- The idea is that the likelihood of the next word's POS tag in a sentence tends to depend on the POS tag of the previous word

Visual representation

- We can visually represent the likelihoods of the next word pos tags





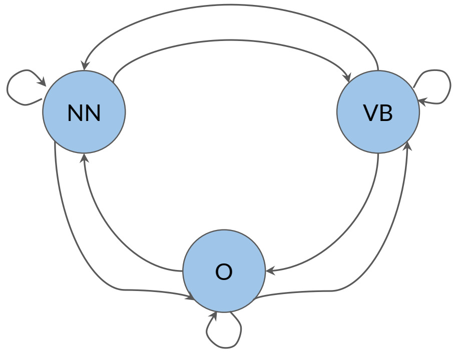
STATES

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

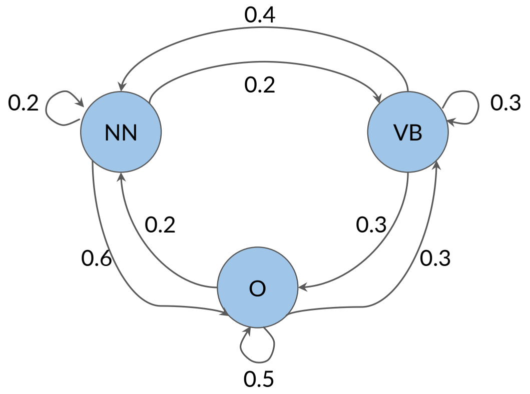
- A Markov chain is a stochastic model that describes a sequence of possible events
  - To get the probability for each event, it needs only the states of the previous event
  - It can be depicted as a directed graph

POS tags as states & transition probabilities

- The parts of speech tags are events that can occur
- To go from one state to another state we define a term that we call transition probabilities
  - Transition probabilities express the chances of going from one POS tag to another



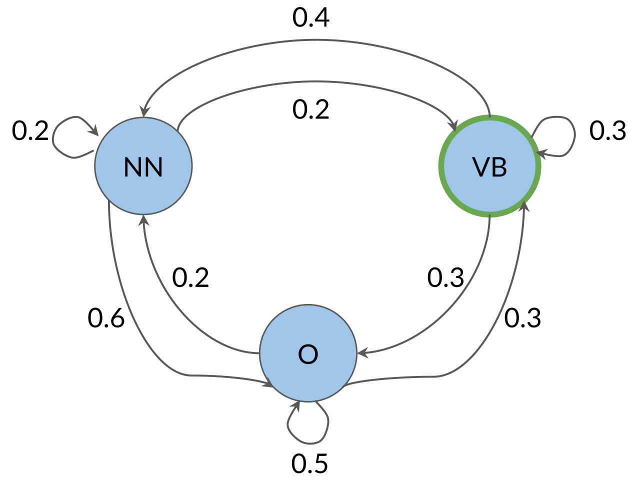
Transition probability



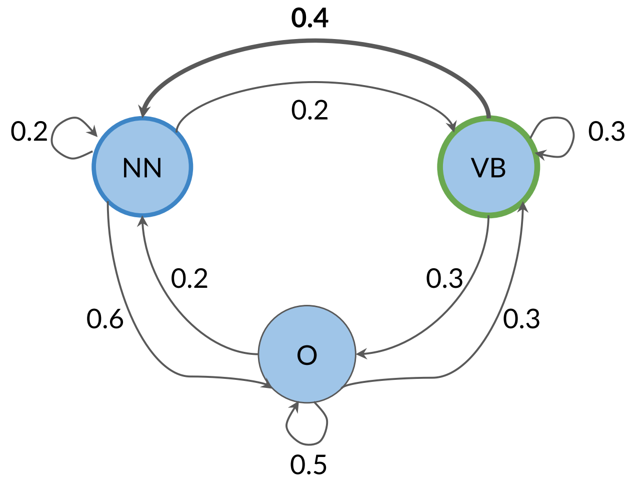
- Markov property
  - The probability of the next event only depends on the current event
    - All one needs to determine the next state is the current state

Transition probabilities

- Example



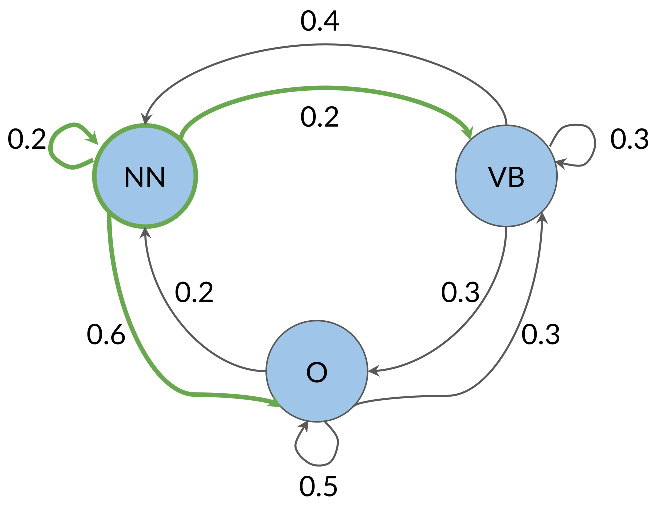
Why not learn something ?



Why not learn something ?

The transition matrix

- States and transition probabilities are stored in a table (transition matrix)
  - equivalent but more compact representation of the Markov chain model



$A =$

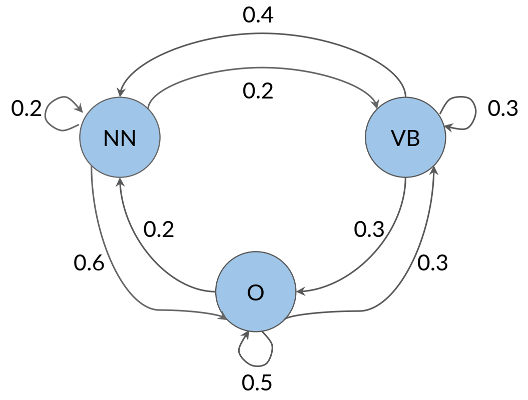
	NN	VB	O
NN (noun)	0.2	0.2	0.6
VB (verb)	0.4	0.3	0.3
O (other)	0.2	0.3	0.5

$$\sum_{j=1}^{N} a_{ij} = 1$$

  - A transition matrix is an NxN matrix, where N is the number of states

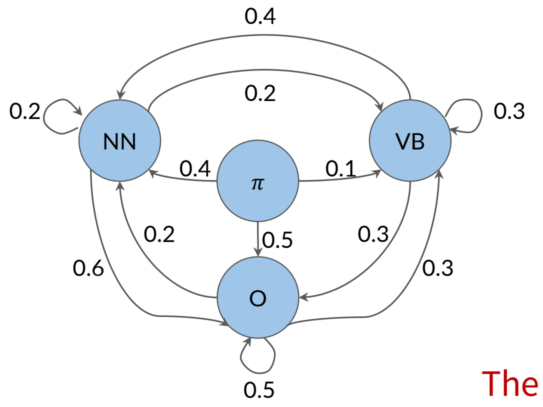
The first word and initial probabilities

- How about the first word of the sentence?
  - We introduce an initial state



Why not learn something ?

NN?
VB?
O?



$A =$

	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

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

The transition matrix now has dimension (N+1)xN

States and Transition matrix

- In general, given a set of states Q, the transition matrix is

States

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

Transition matrix

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

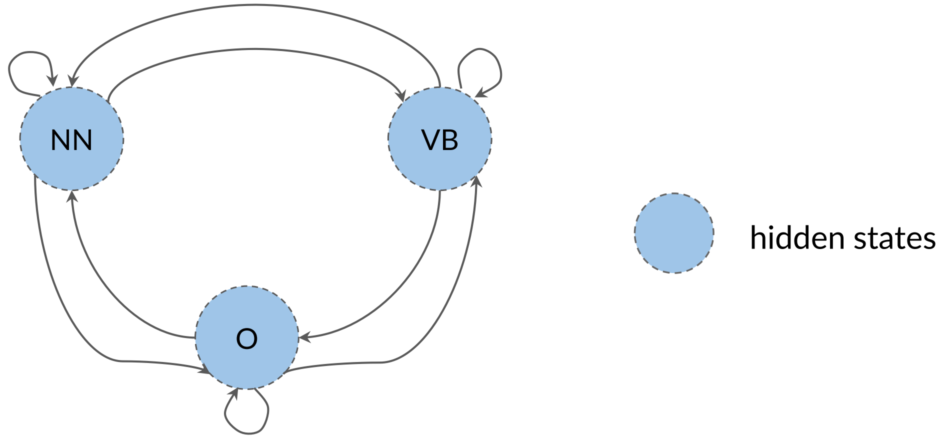
- The initial probabilities are in the first row

Hidden Markov Model

- An HMM is a probabilistic sequence model
  - Given a sequence of units (words, letters, morphemes, sentences, whatever), it computes a probability distribution over possible sequences of labels and chooses the best label sequence
- An HMM allows talking about observed events (like words we see in the input) and hidden events (like POS tags) that we think of as causal factors in our probabilistic model

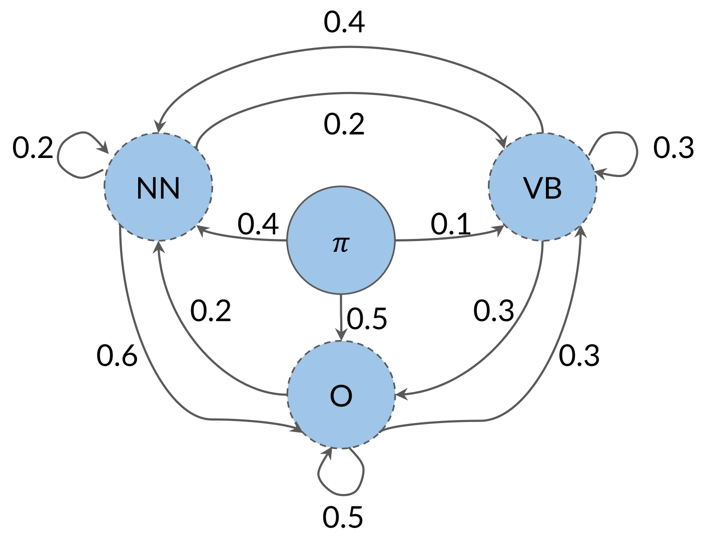
Hidden Markov Model (HMM)

- In the Markov model, the transition probabilities represent the probability from one POS tag to another
  - one can think of these states as hidden states because they are not directly observable from the text data
  - Rather, the words within a phrase are observable (what the machine sees)
- Hidden Markov Models make use of emission probabilities that give us the probability to go from one state (POS tag) to a specific word



Transition probabilities in HMM

- The transition probabilities are represented by an (N+1)xN matrix A
  - N = number of hidden states

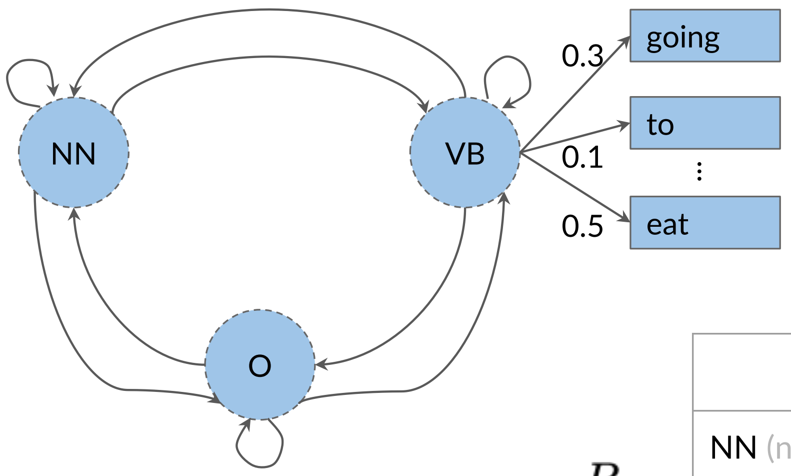


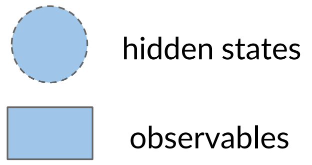
$A =$

	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

Emission probabilities

- Describe the transition from the hidden states of a Hidden Markov model, which are parts of speech, to the observables or the words of a corpus

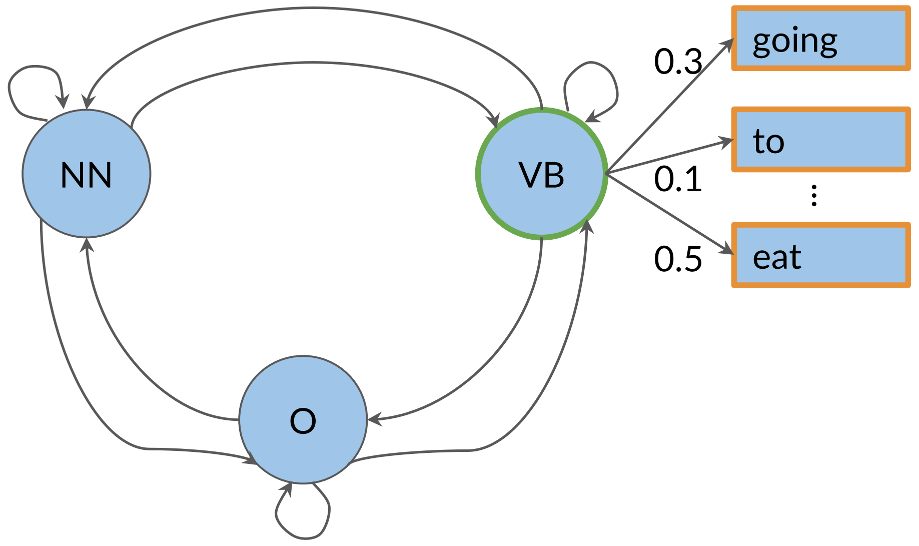




$B =$

	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	0.18	

Emission probabilities



$$B = \begin{array}{c|cccc} & \text{going} & \text{to} & \text{eat} & \ldots \\ \hline \text{NN (noun)} & 0.5 & 0.1 & 0.02 & \\ \text{VB (verb)} & 0.3 & 0.1 & \boxed{0.5} & \\ \text{O (other)} & 0.3 & 0.5 & 0.18 & \end{array}$$

The Emission matrix

- The row sum of probabilities is one

$B =$

	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	0.18	

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

- In this example is that there are emission probabilities greater than zero for all three of our parts of speech tags
  - This is because words can have different parts of speech tag assigned depending on the context in which they appear
    - Example:

He lay on his back. ← NOUN

I'll be back. ← ADV

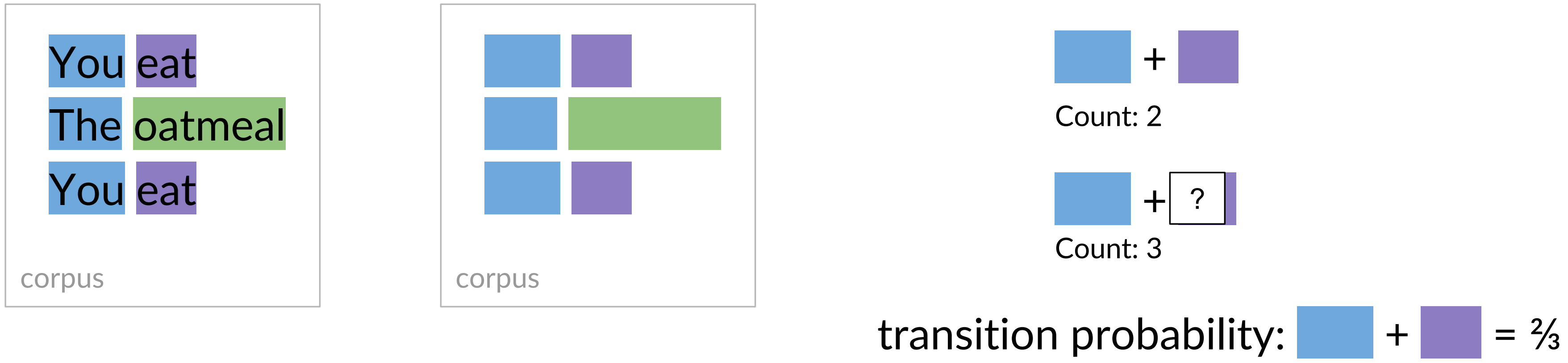
To recap

States	Transition matrix	Emission matrix
$Q = \{q_1, \ldots, q_N\}$	$A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{N+1,1} & \cdots & a_{N+1,N} \end{pmatrix}$	$B = \begin{pmatrix} b_{11} & \cdots & b_{1V} \\ \vdots & \ddots & \vdots \\ b_{N1} & \cdots & b_{NV} \end{pmatrix}$

- N = number of states
- V = number of words

Calculating Probabilities

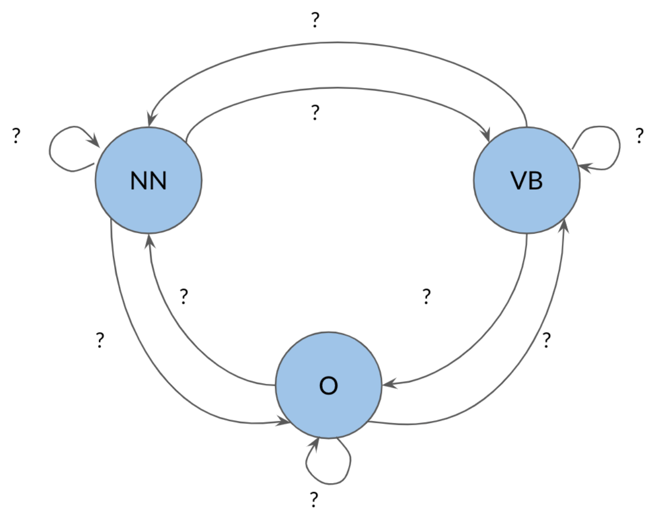
- Given a corpus, let's see how to compute the probabilities for the transition and emission matrices
  - Transition matrix



- To calculate the transition probabilities, we only use the parts of speech tags from the training corpus
  - Example
    - To calculate the probability of the blue parts of the speech tag, transitioning to the purple one, one counts the occurrences of that tag combination in the corpus, which is two

Transition probabilities

- More formally, to calculate all the transition probabilities of the Markov model, we'd first have to count all occurrences of tag pairs in the training corpus



1. Count occurrences of tag pairs

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

2. Calculate probabilities using the counts

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

- C(t(i−1),t(i)) is the count of times tag (i-1) shows up before *tag i*
  - From this, we can compute the probability that a tag shows up after another tag

The corpus and its preparation

- Let’s consider a little example corpus (Haiku, shorts Japanese poetry)
  - Before any tagging activity, preprocess it by adding a starting symbol for each sentence (for the initial probabilities) and lowering all the letters

```
In a Station of the Metro
The apparition of these faces in the crowd:
Petals on a wet, black bough.
```

Ezra Pound, 1913

```
<s> In a Station of the Metro
<s> The apparition of these faces in the crowd:
<s> Petals on a wet, black bough.
```

```
<s> in a station of the metro
<s> the apparition of these faces in the crowd:
<s> petals on a wet, black bough.
```

The Transition matrix

- To populate the transition matrix, you need to calculate the probability of a tagging happening, given that another tag already happened

$A =$

	NN	VB	O
π			
NN (noun)			
VB (verb)			
O (other)			

<s> in a station of the metro

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

<s> petals on a wet, black bough.

$A =$

	NN	VB	O
π	$C(\pi,\text{NN})$		
NN (noun)	$C(\text{NN},\text{NN})$		
VB (verb)	$C(\text{VB},\text{NN})$		
O (other)	$C(\text{O},\text{NN})$		

<s> in a station of the metro

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

<s> petals on a wet, black bough.

The Transition matrix

$A =$

	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.

$$P(t_i|t_{i-1}) = \frac{\boxed{C(t_{i-1}, t_i)}}{\boxed{\sum_{j=1}^{N} C(t_{i-1}, t_j)}} \qquad P(\text{NN}|\pi) = \frac{\boxed{C(\pi, \text{NN})}}{\boxed{\sum_{j=1}^{N} C(\pi, t_j)}} = \frac{1}{3} \qquad P(\text{NN}|\text{O}) = \frac{\boxed{C(\text{O}, \text{NN})}}{\boxed{\sum_{j=1}^{N} C(\text{O}, t_j)}} = \frac{6}{14}$$

Smoothing

- Two problems here:
  1. The row sum of the VB tag is 0, which would lead to a division by 0
  2. Lots of entries in the transition matrix are 0, meaning that these transitions will have a probability of 0
  - This won't work if you want the model to generalize to other Haiku, which might contain verbs

$A =$

	NN	VB	O	
π	1+ε	0+ε	2+ε	3+3*ε
NN	0+ε	0+ε	6+ε	6+3*ε
VB	0+ε	0+ε	0+ε	0+3*ε
O	6+ε	0+ε	8+ε	14+3*ε

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

$A =$

	NN	VB	O
π	0.3333	0.0003	0.6663
NN	0.0001	0.0001	0.9996
VB	0.3333	0.3333	0.3333
O	0.4285	0.0000	0.5713

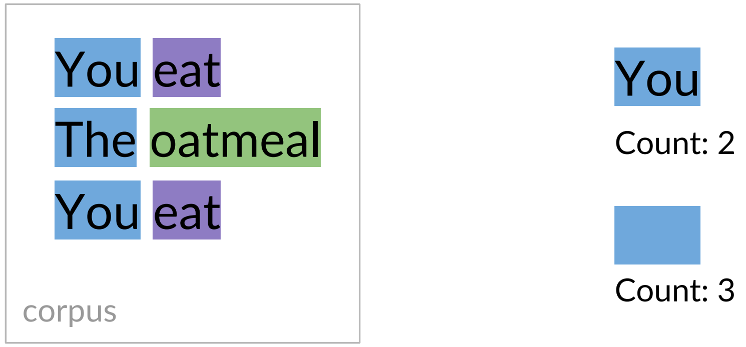
$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i) + \epsilon}{\sum_{j=1}^{N} C(t_{i-1}, t_j) + N * \epsilon}$$

First 4 decimal positions.
The actual value is 0.00007141

$\epsilon = 0.001$

Populating the Emission Matrix

- We count the co-occurrences of a POS tag with a specific word
- Example



emission probability: You = ⅔

The Emission Matrix

- Count how often a word is tagged with a specific tag (e.g., NN, VB or O)

$B =$

	in	a	...
NN (noun)	$C(\text{NN}, \text{in})$		
VB (verb)	$C(\text{VB}, \text{in})$		
O (other)	$C(\text{O}, \text{in})$		

<s> in a station of the metro

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

<s> petals on a wet, black bough.

$B =$

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

<s> in a station of the metro

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

<s> petals on a wet, black bough.

The Emission matrix

- N = number of tags
- V = size of the vocabulary

$B =$

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

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

- Smoothing for better generalization

HMM tagging as decoding

- The task of determining the sequence of the hidden variables corresponding to the sequence of observations is called decoding
- Decoding
  - *Given as input an HMM with A and B matrices, and a sequence of observations $O=o_1,o_2, \ldots, o_T$, find the most probable sequence of states $Q=q_1q_2q_3\ldots q_T$*
  - For POS tagging, the goal of HMM decoding is to choose the tag sequence $t_1,\ldots,t_n$ that is most probable given the observation sequence of n words $w_1,\ldots,w_n$
- The decoding algorithm for HMMs is the Viterbi algorithm