Queen Mary
University of London

ECS763 Natural Language Processing

Unit 5: Sequence Classification

Lecturer: Julian Hough
School of Electronic Engineering and Computer Science

OUTLINE

1) Sequence Tagging Tasks: POS tagging and NER
2) Generative: Hidden Markov Models
3) Discriminative: Conditional Random Fields

OUTLINE

1) Sequence Tagging Tasks: POS tagging and NER
2) Generative: Hidden Markov Models
3) Discriminative: Conditional Random Fields

Sequence Tagging Tasks

- Part-of-Speech (POS) tagging:

mary	hires	a	detective
PN	VBZ	DET	CN

- Named Entity tagging/Named Entity Recognition (NER):

Today	President	Donald	J.	Trump	announced
O	B-PER	I-PER	I-PER	E-PER	O

- Dialogue Act tagging:

A:	So do you go to college right now?	YN-QUESTION
B:	Yeah	YES-ANSWER
A:	Are yo-	ABANDONED
B:	it’s my last year	STATEMENT
A:	What did you say?	CLARIFY
B:	my last year	NP-ANSWER
A:	Oh good for you	APPRECIATION
B:	uh-huh	BACKCHANNEL

- Why are these not just individual token (word/sentence) classification tasks? Order matters…

Part-of-speech (POS) tagging

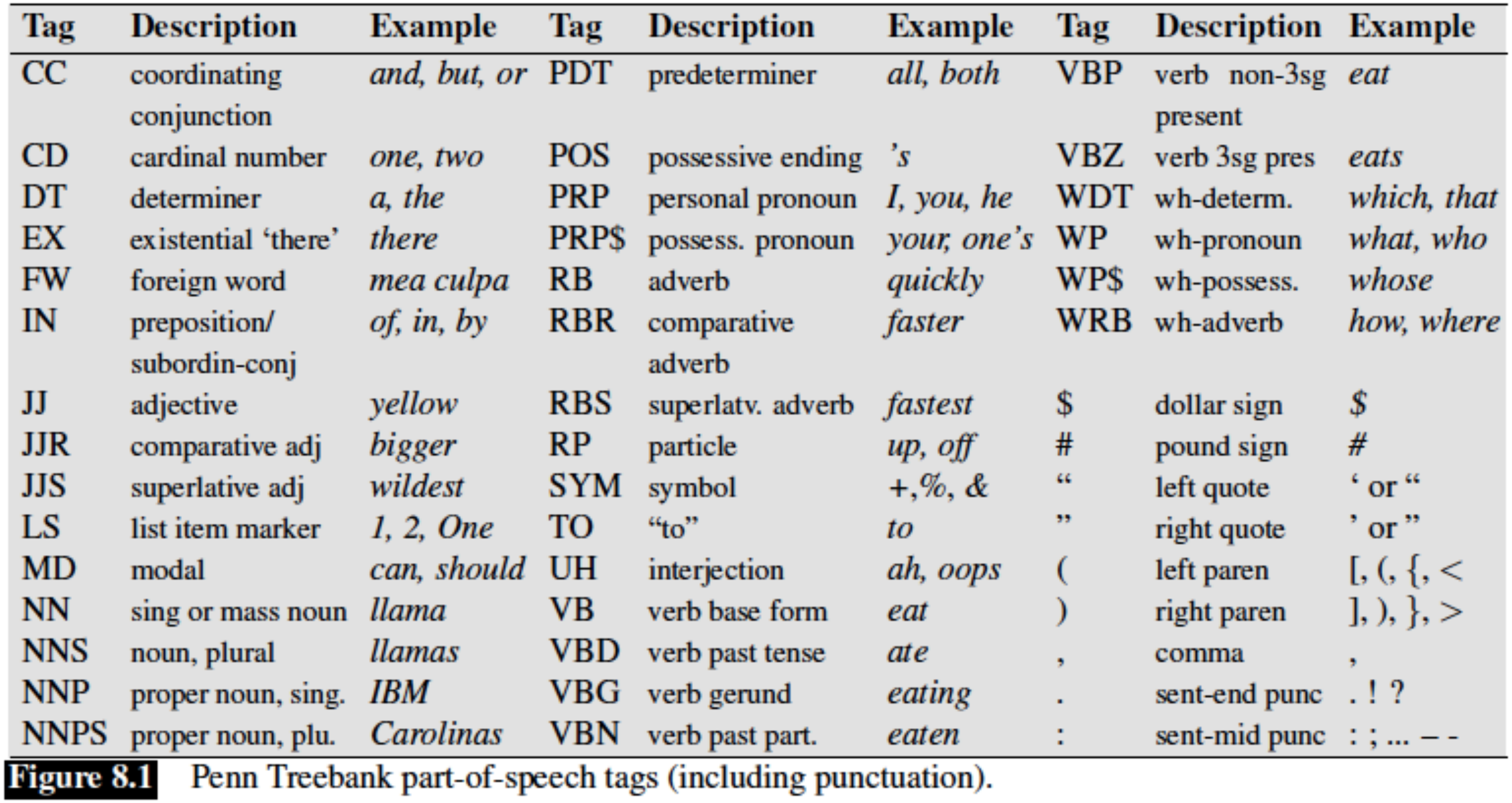
- One way of dividing words into different **classes** is by the **part-of-speech (POS)** assigned to them.
- Most POS tags implicitly encode fine-grained specializations of eight basic parts of a language:
  - noun, verb, pronoun, preposition, adjective, adverb, conjunction, article
- These categories are based on **morphological/syntactic** similarities rather than semantic similarities.
- POS tags used downstream in other tasks like **parsing** and **named entity recognition**.

Part-of-speech (POS) tagging

- **Nouns**
  - NN = singular noun e.g., man, dog, park
  - NNS = plural noun e.g., telescopes, houses, buildings
  - NNP = proper noun e.g., Smith, Gates, IBM
- **Verbs**
  - VB = verb base form e.g. eat
  - VBZ = 3rd person singular present form e.g. eats
- **Determiners**
  - DT = determiner e.g., the, a, some, every
- **Adjectives**
  - JJ = adjective e.g., red, green, large, idealistic
- **Connectives**
  - CC = coordinating conjunction e.g. and, or

Part-of-speech (POS) tagging

- From Jurafsky and Martin, Chapter 8. Penn Treebank POS tags



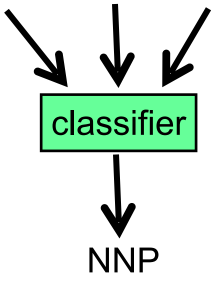
Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	*$*
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	*#*
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

Figure 8.1 Penn Treebank part-of-speech tags (including punctuation).

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



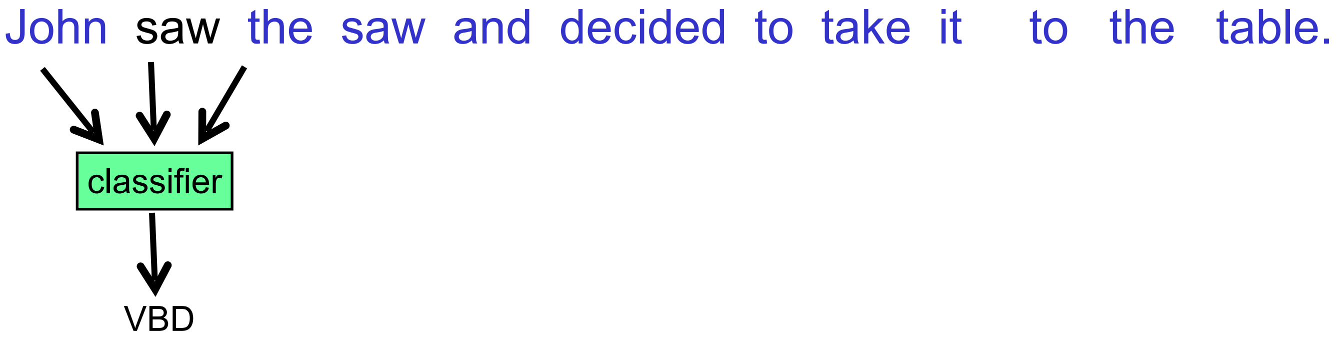
classifier

NNP

Part-of-Speech (POS) tagging

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



John saw the saw and decided to take it to the table.

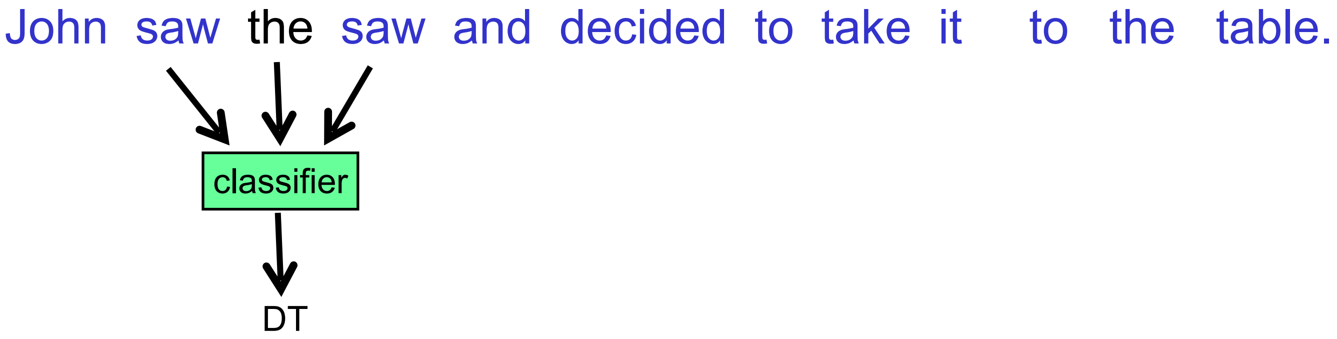
classifier

VBD

Part-of-Speech (POS) tagging

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

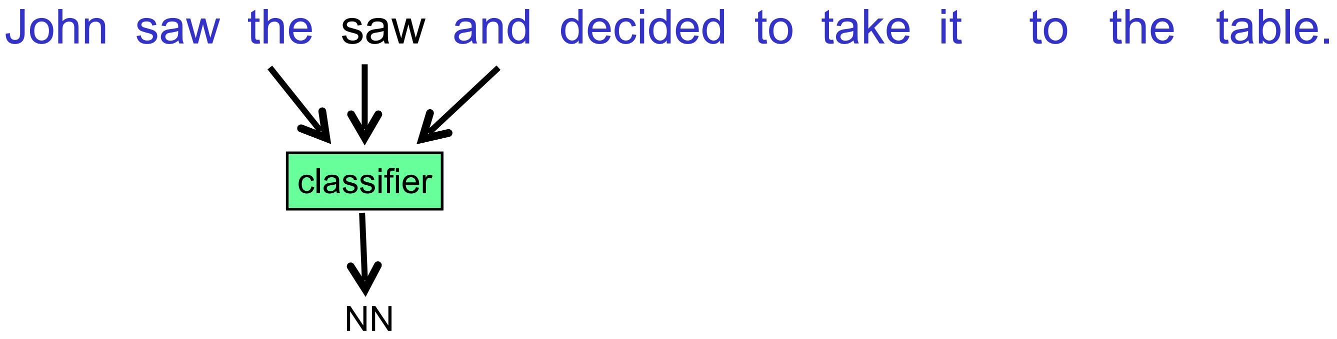


John saw the saw and decided to take it to the table.

Part-of-Speech (POS) tagging

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

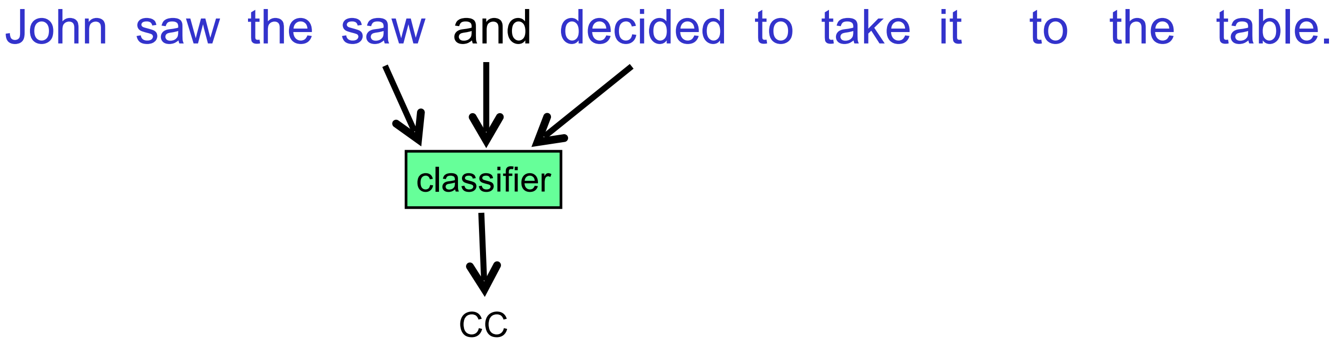


John saw the saw and decided to take it to the table.

Part-of-Speech (POS) tagging

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



John saw the saw and decided to take it to the table.

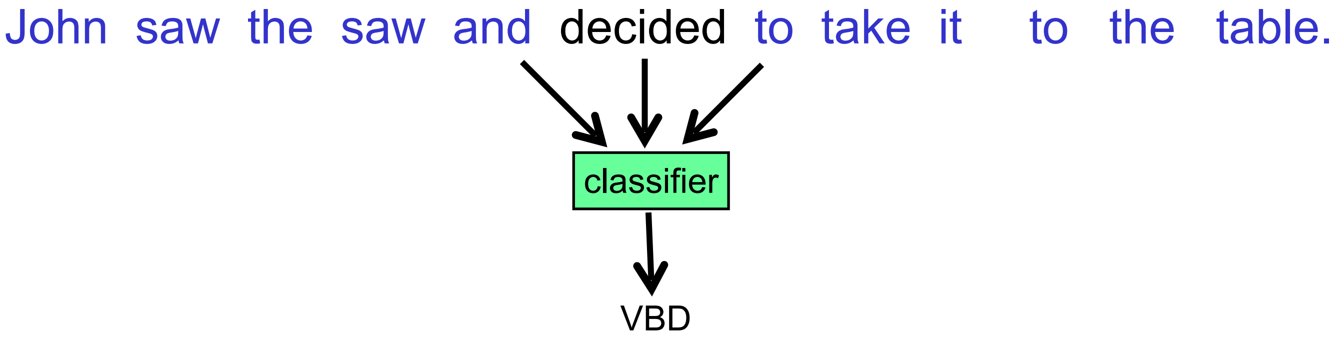
classifier

CC

Part-of-Speech (POS) tagging

Sequence Labeling as Classification

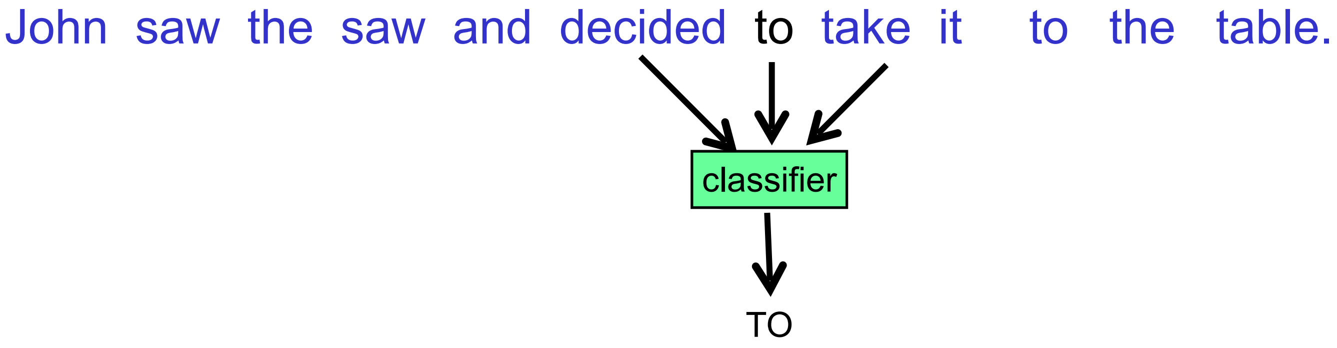
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



Part-of-Speech (POS) tagging

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



John saw the saw and decided to take it to the table.

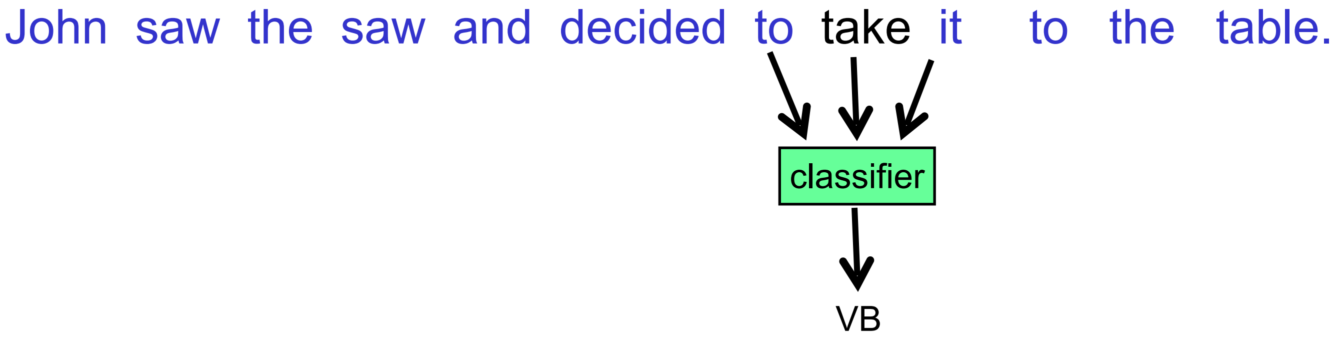
classifier

TO

Part-of-Speech (POS) tagging

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



John saw the saw and decided to take it to the table.

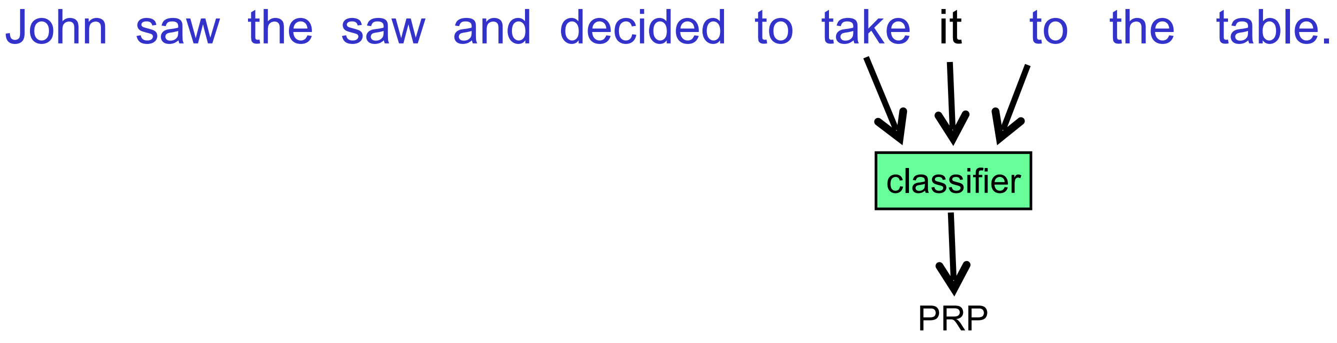
classifier

VB

Part-of-Speech (POS) tagging

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



John saw the saw and decided to take it to the table.

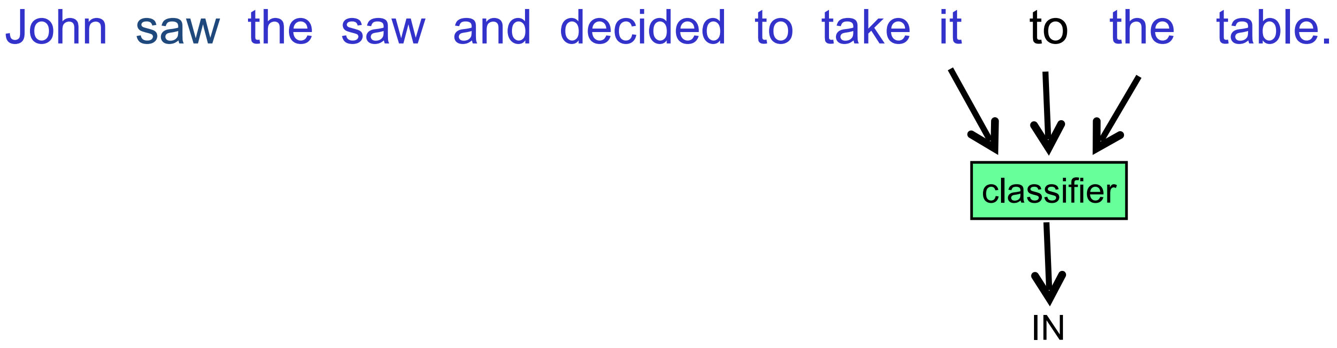
classifier

PRP

Part-of-Speech (POS) tagging

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



John saw the saw and decided to take it to the table.

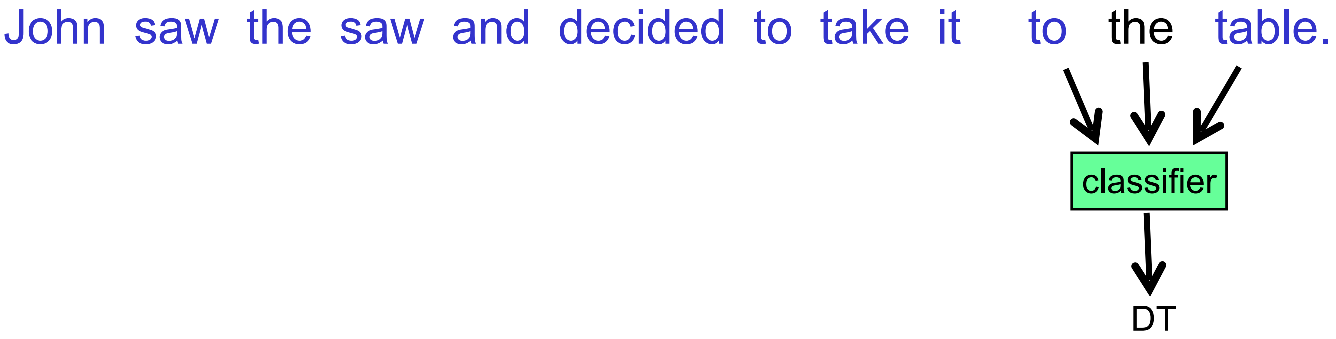
classifier

IN

Part-of-Speech (POS) tagging

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



John saw the saw and decided to take it to the table.

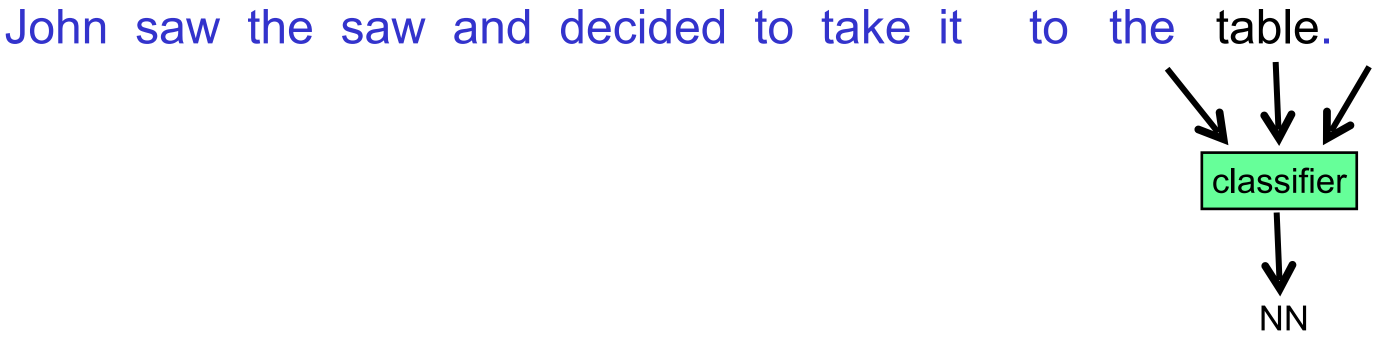
classifier

DT

Part-of-Speech (POS) tagging

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



John saw the saw and decided to take it to the table.

classifier

NN

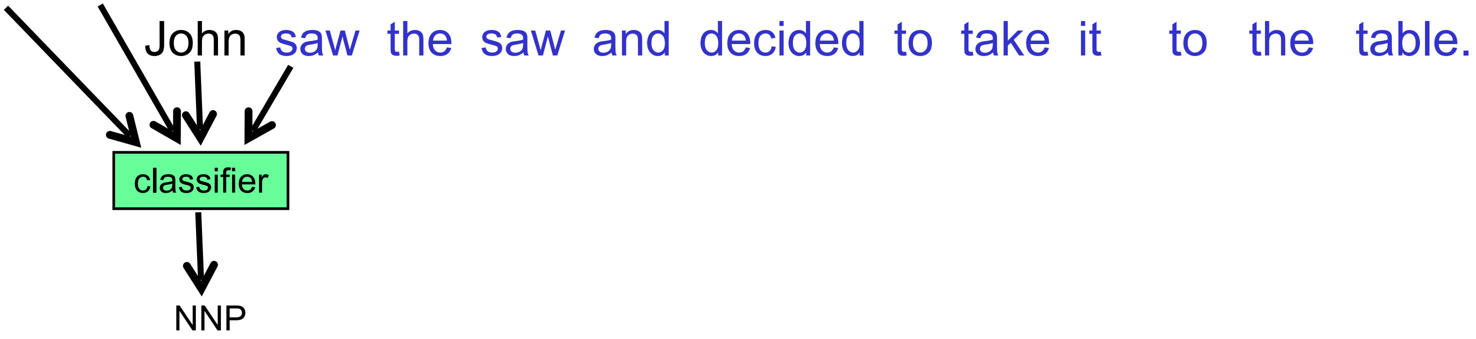
Part-of-Speech (POS) tagging

Using outputs as inputs

- Better input features are usually the **categories** of the surrounding tokens, but these are not available yet as they haven’t been classified.

- You can use category of either the preceding or succeeding tokens by going forward or back and using previous output from the classifier at test time.

Forward Classification



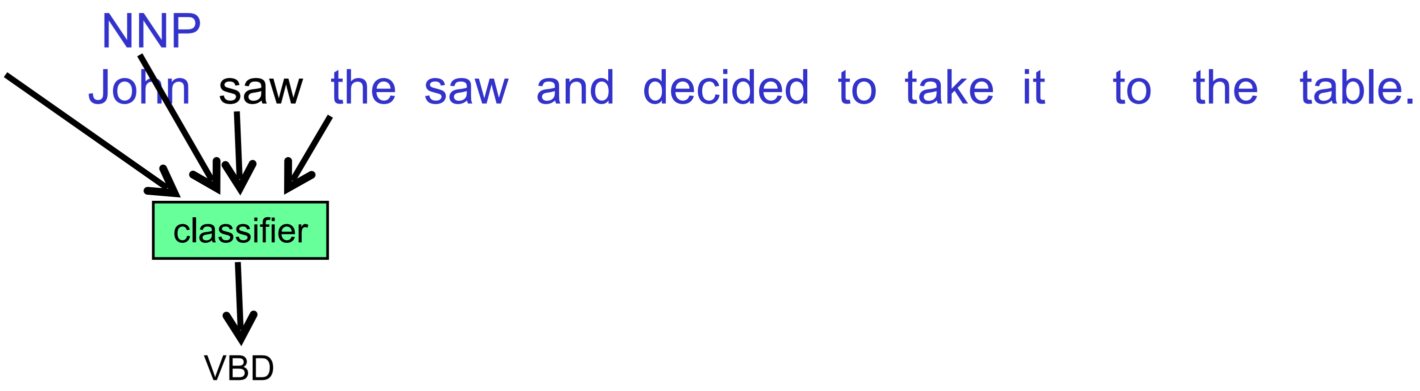
John saw the saw and decided to take it to the table.

classifier

NNP

Part-of-Speech (POS) tagging

Forward Classification



NNP

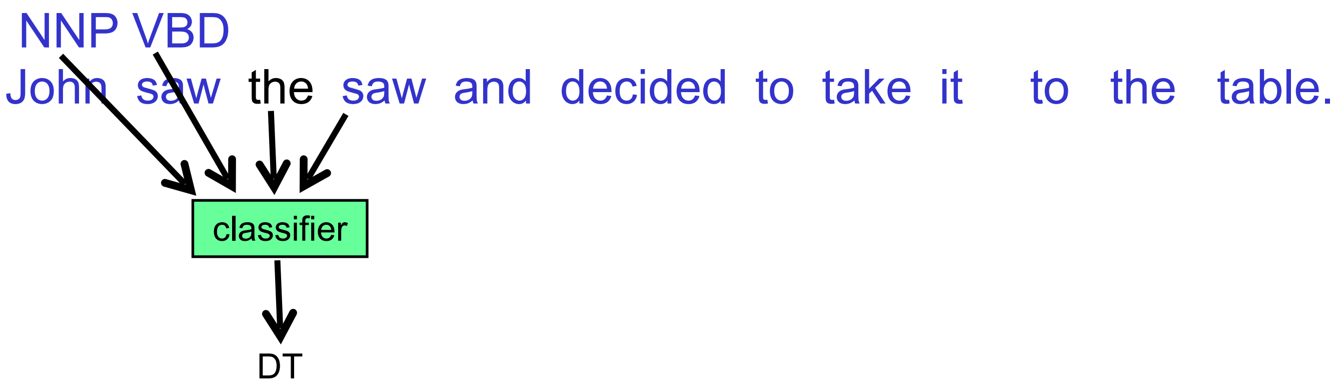
John saw the saw and decided to take it to the table.

classifier

VBD

Part-of-Speech (POS) tagging

Forward Classification



NNP VBD

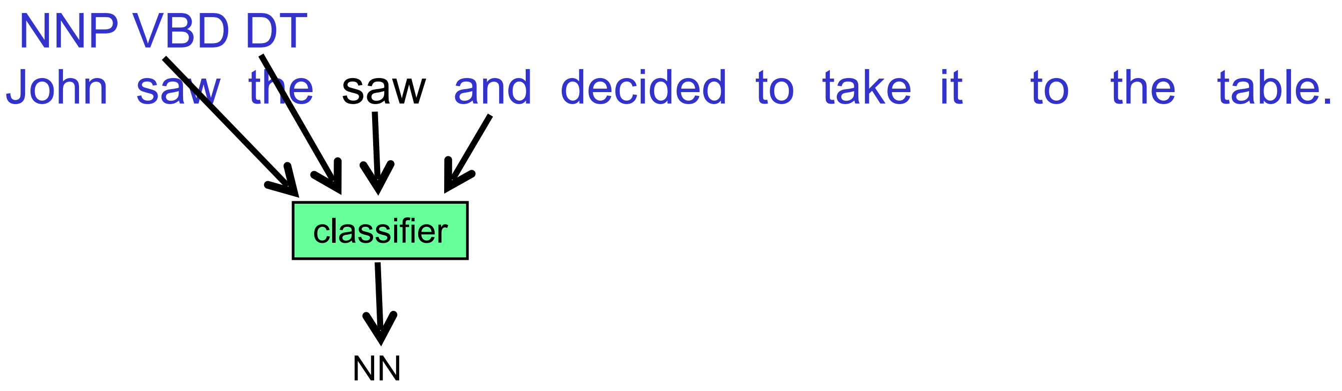
John saw the saw and decided to take it to the table.

classifier

DT

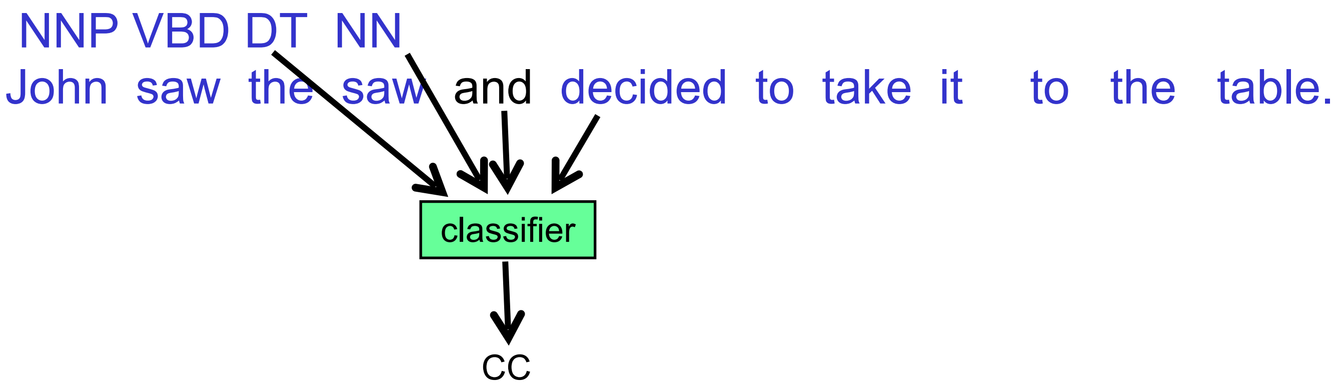
Part-of-Speech (POS) tagging

Forward Classification



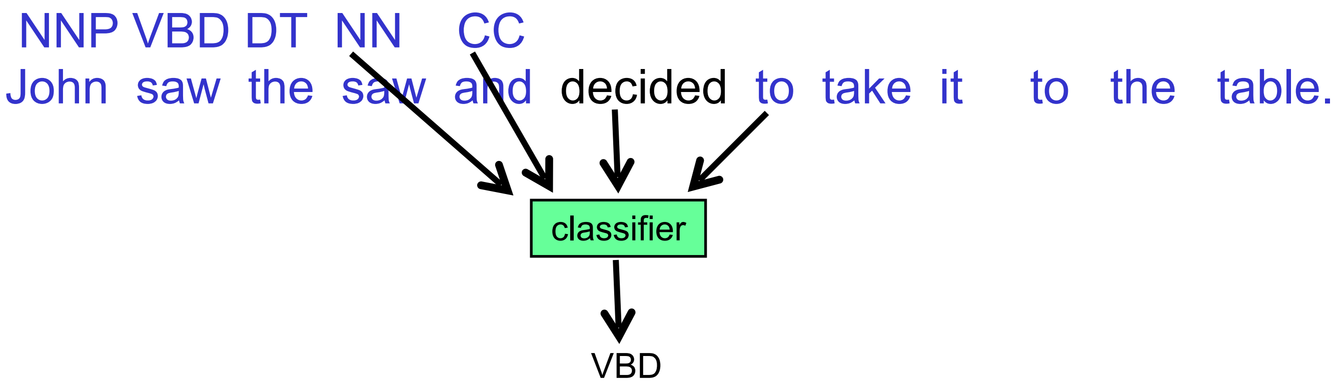
Part-of-Speech (POS) tagging

Forward Classification



Part-of-Speech (POS) tagging

Forward Classification



NNP VBD DT NN CC

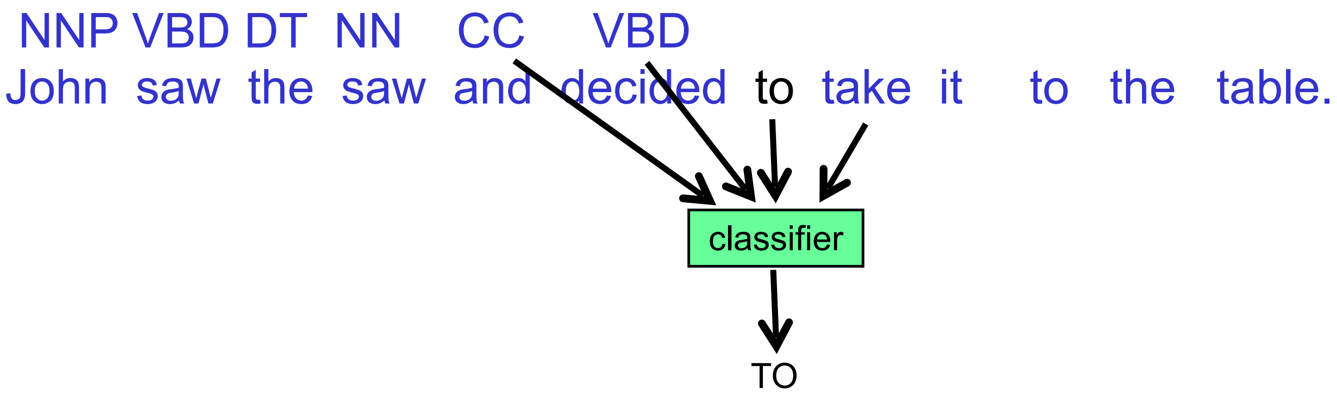
John saw the saw and decided to take it to the table.

classifier

VBD

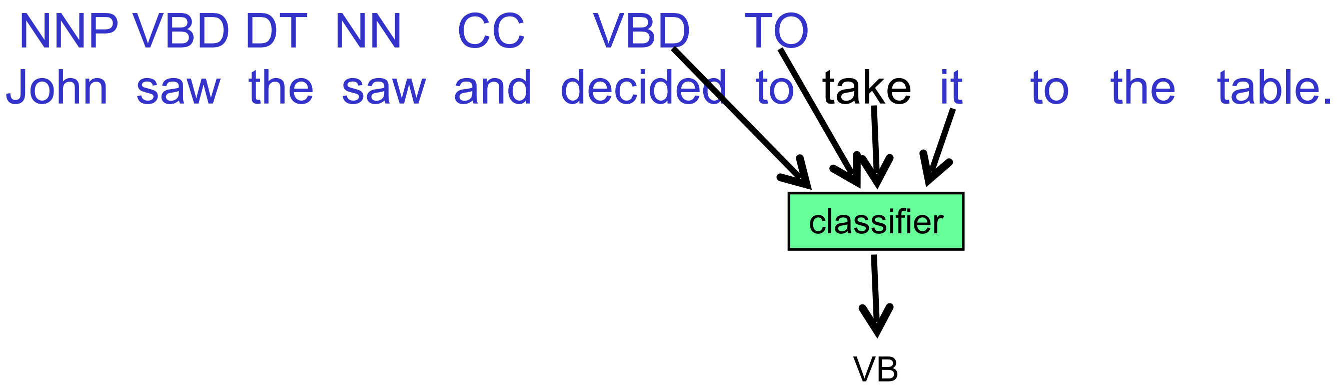
Part-of-Speech (POS) tagging

Forward Classification



Part-of-Speech (POS) tagging

Forward Classification



NNP VBD DT NN CC VBD TO

John saw the saw and decided to take it to the table.

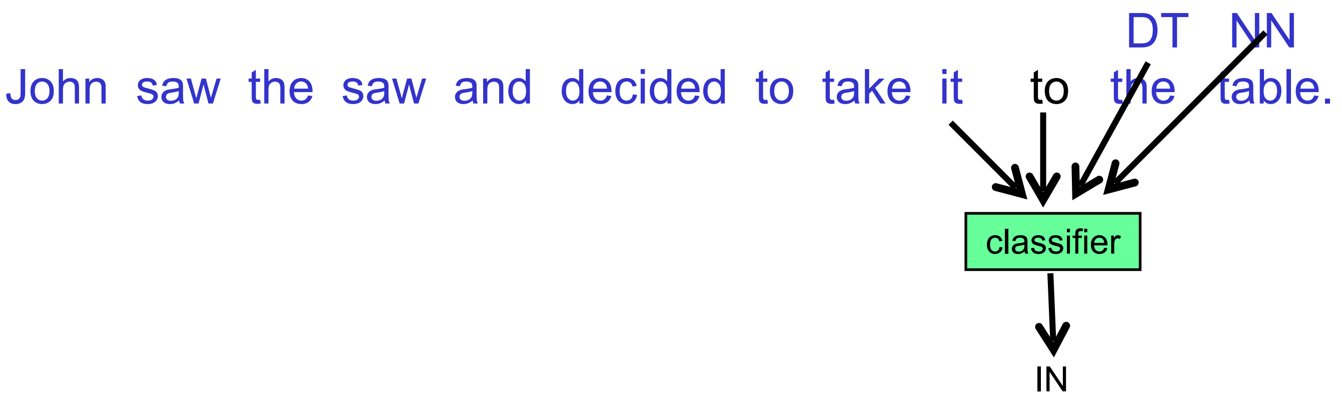
classifier

VB

Part-of-Speech (POS) tagging

Backward Classification

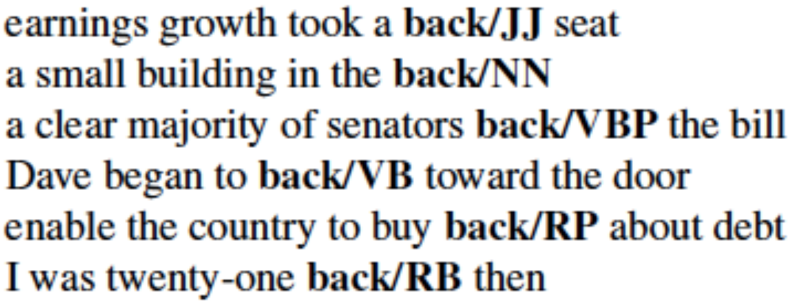
- Disambiguating “to” in this case would be even easier backward.



Part-of-Speech (POS) tagging

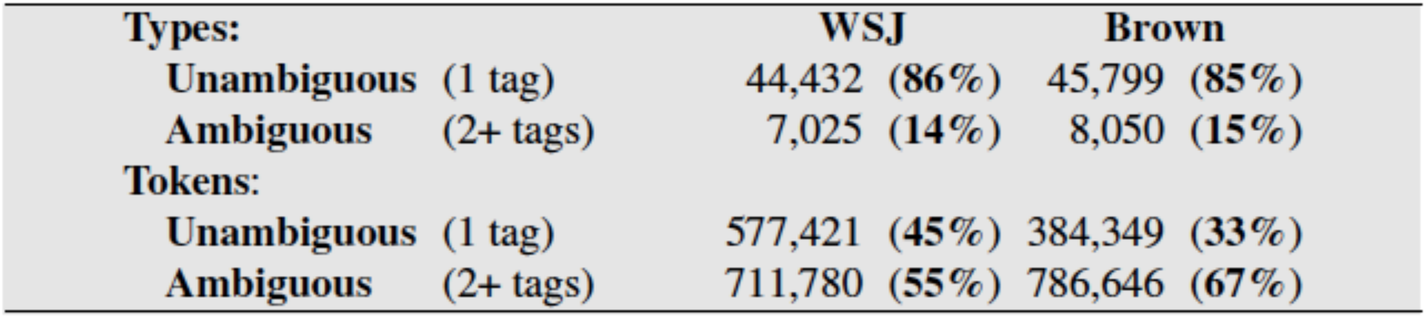
POS-tagging: Evaluation

- POS-tagging is a disambiguation task (as there can be more than one possible tag per word)- see ‘back’:



earnings growth took a **back/JJ** seat
a small building in the **back/NN**
a clear majority of senators **back/VBP** the bill
Dave began to **back/VB** toward the door
enable the country to buy **back/RP** about debt
I was twenty-one **back/RB** then

- However not many word **types** have ambiguous tags, and in fact it’s a relatively ‘easy’ task in NLP, though lots of **tokens** do!:



Types:		**WSJ**	**Brown**
Unambiguous	(1 tag)	44,432 (**86%**)	45,799 (**85%**)
Ambiguous	(2+ tags)	7,025 (**14%**)	8,050 (**15%**)
Tokens:			
Unambiguous	(1 tag)	577,421 (**45%**)	384,349 (**33%**)
Ambiguous	(2+ tags)	711,780 (**55%**)	786,646 (**67%**)

Figure 8.2 Tag ambiguity for word types in Brown and WSJ, using Treebank-3 (45-tag) tagging. Punctuation were treated as words, and words were kept in their original case.

POS-tagging: Evaluation

- A **majority class baseline (per word)** is useful to compare a model against:
  - Given an ambiguous word, assign it the tag that it had most frequently in the ground-truth training data.

Model	Accuracy on Sec. 22-24 of the WSJ
Majority class baseline from WSJ training	92.74%
State-of-the-art POS taggers	97-98%

Named Entity Recognition (NER)

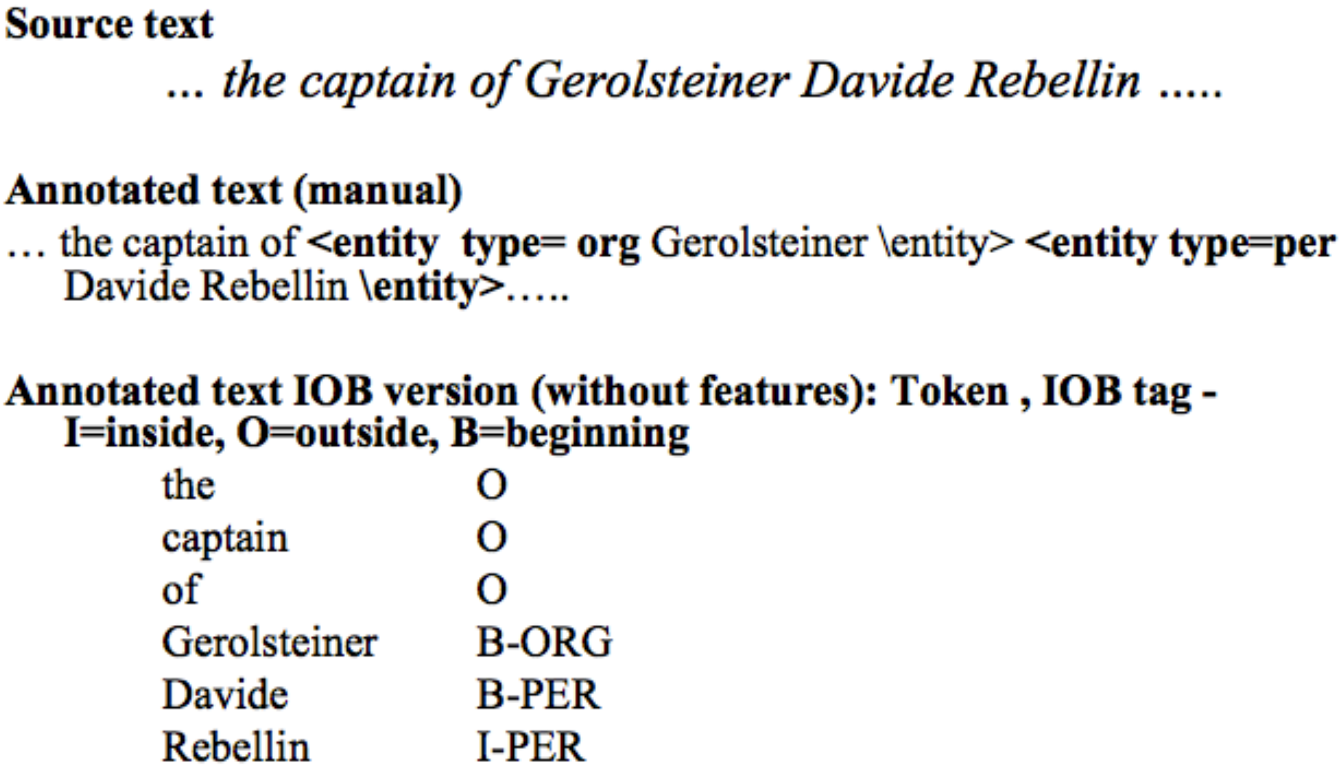
Input:

Apple Inc., formerly Apple Computer, Inc., is an American multinational corporation headquartered in Cupertino, California that designs, develops, and sells consumer electronics, computer software and personal computers. It was established on April 1, 1976, by Steve Jobs, Steve Wozniak and Ronald Wayne.

Output:

Apple Inc., formerly Apple Computer, Inc., is an American multinational corporation headquartered in Cupertino, California that designs, develops, and sells consumer electronics, computer software and personal computers. It was established on April 1, 1976, by Steve Jobs, Steve Wozniak and Ronald Wayne.

Typical ML tagging approach to NER: IOB representation



Source text

... the captain of Gerolsteiner Davide Rebellin .....

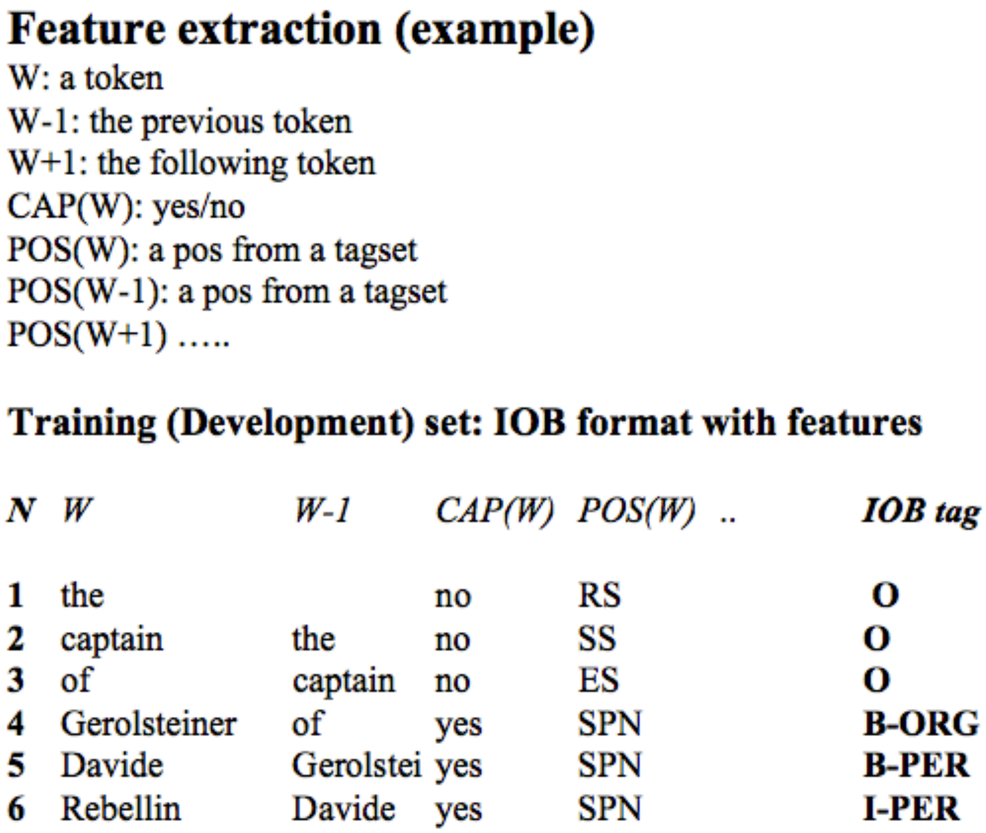
Annotated text (manual)

… the captain of **<entity type= org** Gerolsteiner \entity> **<entity type=per** Davide Rebellin **\entity>**…..

Annotated text IOB version (without features): Token , IOB tag - I=inside, O=outside, B=beginning

the	O
captain	O
of	O
Gerolsteiner	B-ORG
Davide	B-PER
Rebellin	I-PER

Typical ML tagging approach to NER: Features



Feature extraction (example)

W: a token
W-1: the previous token
W+1: the following token
CAP(W): yes/no
POS(W): a pos from a tagset
POS(W-1): a pos from a tagset
POS(W+1) …..

Training (Development) set: IOB format with features

N	*W*	*W-1*	*CAP(W)*	*POS(W)*	..	***IOB tag***
1	the		no	RS		**O**
2	captain	the	no	SS		**O**
3	of	captain	no	ES		**O**
4	Gerolsteiner	of	yes	SPN		**B-ORG**
5	Davide	Gerolstei	yes	SPN		**B-PER**
6	Rebellin	Davide	yes	SPN		**I-PER**

Features

For each running word:

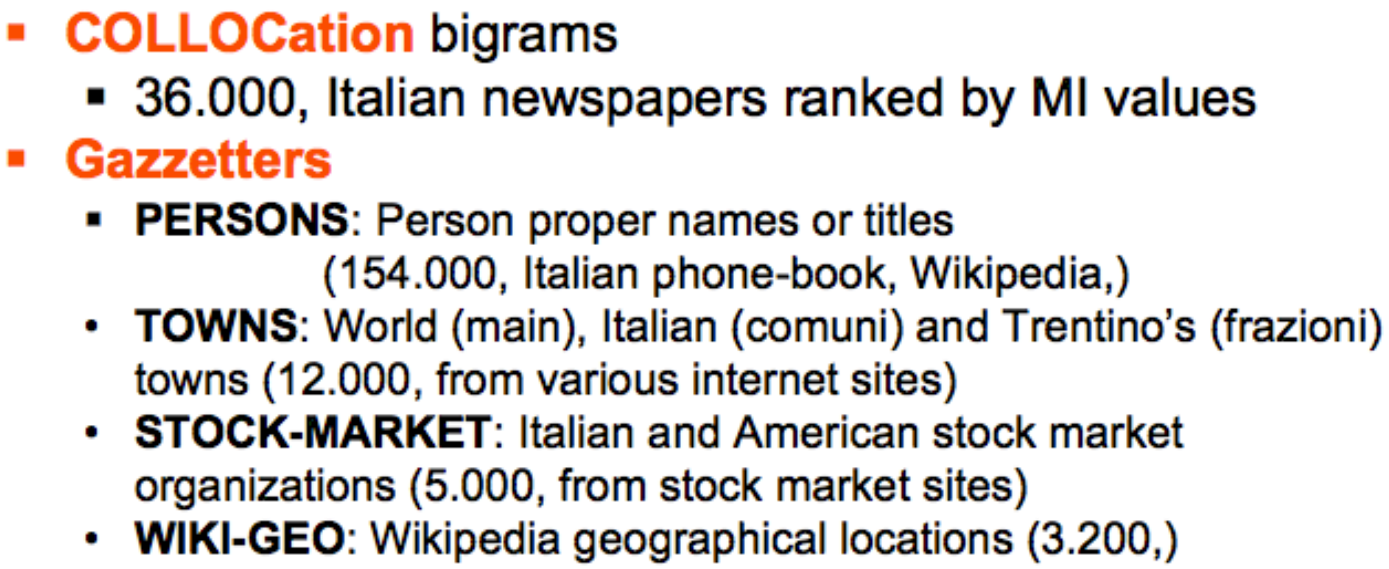
- **WORD**: the word itself (both unchanged and lower-cased)
  e.g. Casa casa

- **POS**: the part of speech of the word (as produced by TagPro)
  e.g. Oggi SS (singular noun)

- **AFFIX**: prefixes/suffixes (1, 2, 3 or 4 chars. at the start/end of the word)
  e.g. Oggi {o,og,ogg,oggi, – i,gi,ggi,oggi}

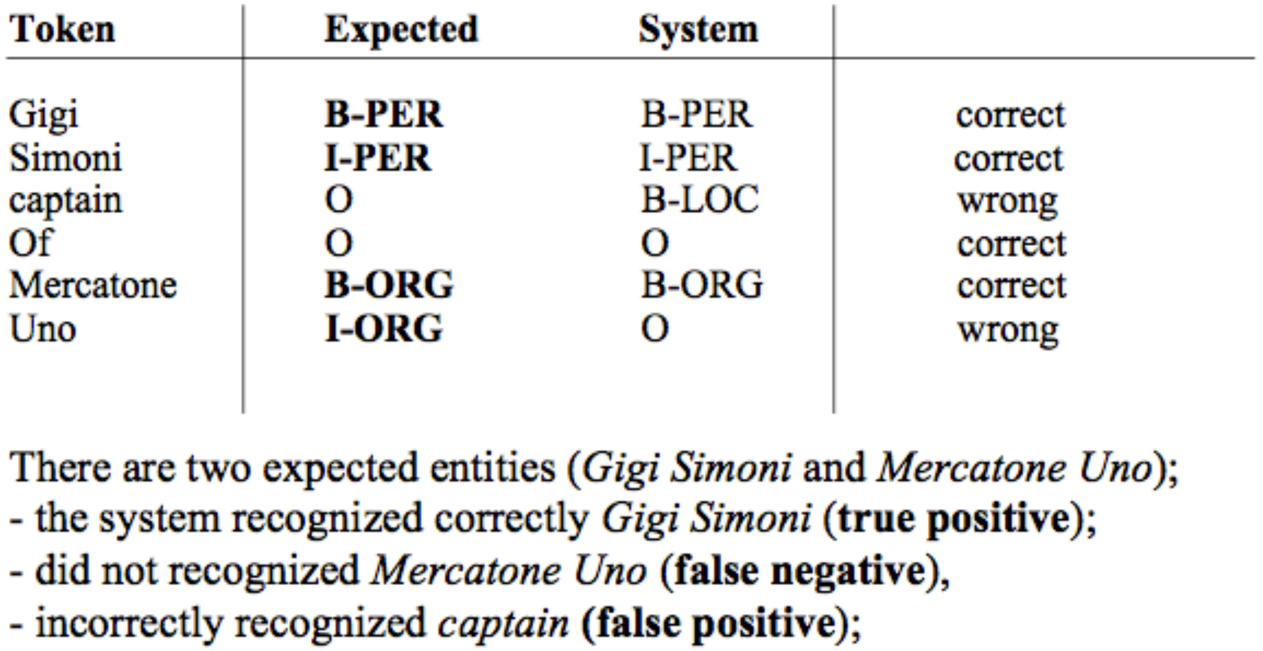
- **ORTHOgraphic** information (e.g. capitalization, hyphenation)
  e.g. Oggi C (capitalized)
  oggi L (lowercased)

Features



- **COLLOCation** bigrams
  - 36.000, Italian newspapers ranked by MI values
- **Gazzetters**
  - **PERSONS**: Person proper names or titles (154.000, Italian phone-book, Wikipedia,)
  - **TOWNS**: World (main), Italian (comuni) and Trentino's (frazioni) towns (12.000, from various internet sites)
  - **STOCK-MARKET**: Italian and American stock market organizations (5.000, from stock market sites)
  - **WIKI-GEO**: Wikipedia geographical locations (3.200,)

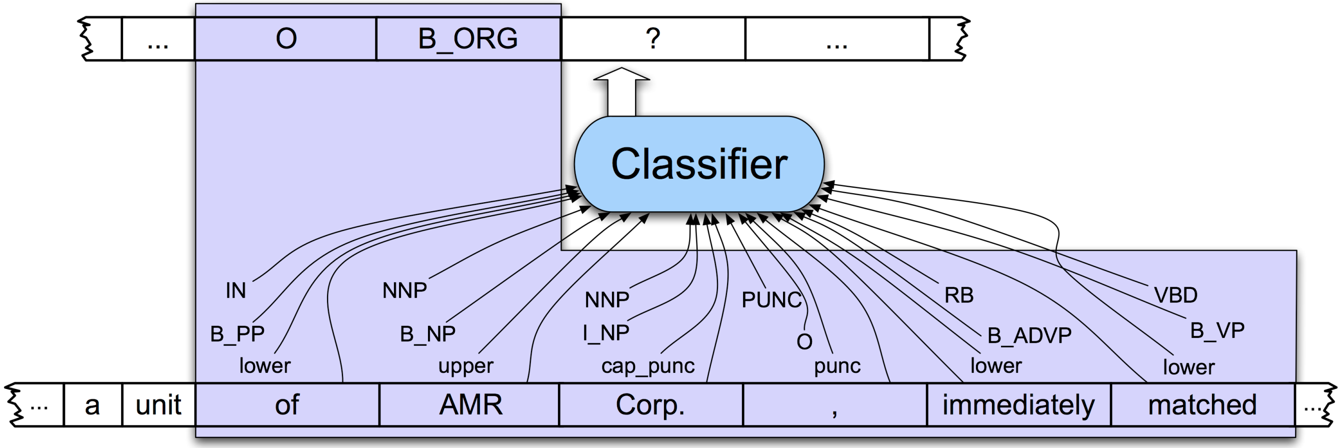
NER: Evaluation



Token	Expected	System	
Gigi	**B-PER**	B-PER	correct
Simoni	**I-PER**	I-PER	correct
captain	O	B-LOC	wrong
Of	O	O	correct
Mercatone	**B-ORG**	B-ORG	correct
Uno	**I-ORG**	O	wrong

There are two expected entities (*Gigi Simoni* and *Mercatone Uno*);
- the system recognized correctly *Gigi Simoni* (**true positive**);
- did not recognized *Mercatone Uno* (**false negative**),
- incorrectly recognized *captain* **(false positive**);

NER as Sequence Labeling

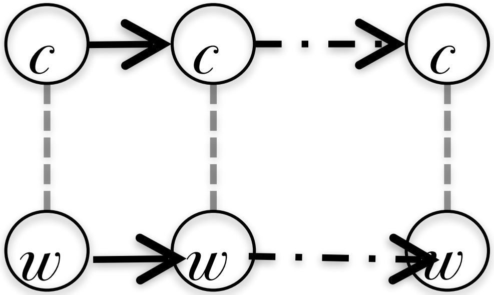


OUTLINE

1) Sequence Tagging Tasks: POS tagging and NER
2) Generative: Hidden Markov Models
3) Discriminative: Conditional Random Fields

Sequence Labelling

- Sequence labelling/tagging
  - A classification problem, but over sequences.
    - Often from words to a sequence of class labels. e.g.:
      - POS-tagging
      - Named Entity Recognition (NER)



- We could try:
  - Rule-based classifier:
    - E.g. transformation-based learning (old school)
  - **Generative sequence model:**
    - (remember Naïve Bayes?) – **Hidden Markov Models**
  - **Discriminative sequence model:**
    - (remember Logistic Regression?) – **Conditional Random Fields**

Generative models- look familiar?

- Unigram language model

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i)$$

- Bigram language model

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_{i-1})$$

- N-gram language model

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- Naïve Bayes

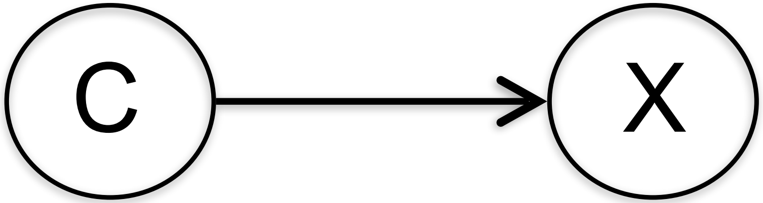
$$P(c_j \mid d) = P(c_j) \prod_i P(w_i \mid c_j)$$

Bayes Rule (Reminder)

- Generative models (non sequence):

$$P(C,X) = P(C \mid X)P(X) = P(X \mid C)P(C)$$

$$\boxed{P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)}}$$



C = latent (hidden) variable/state/class
X = instance data (features)

Bayes Rule

- For lots of NLP sequence classification, observations are **words** and latent variables are **classes**:

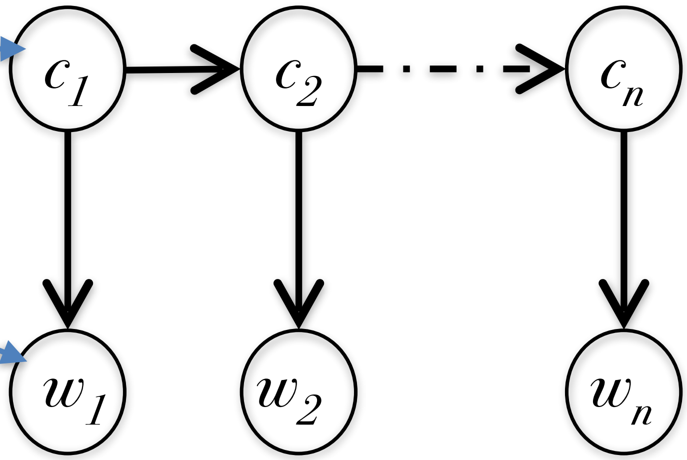
$$P(C|W) = \frac{P(W|C)P(C)}{P(W)}$$

$$P(c_1 \ldots\ c_n | w_1 \ldots w_n) = \frac{P(w_1 \ldots w_n | c_1 \ldots\ c_n) P(c_1 \ldots\ c_n)}{P(w_1 \ldots w_n)}$$

Hidden class/tag sequence (e.g. POS tag)

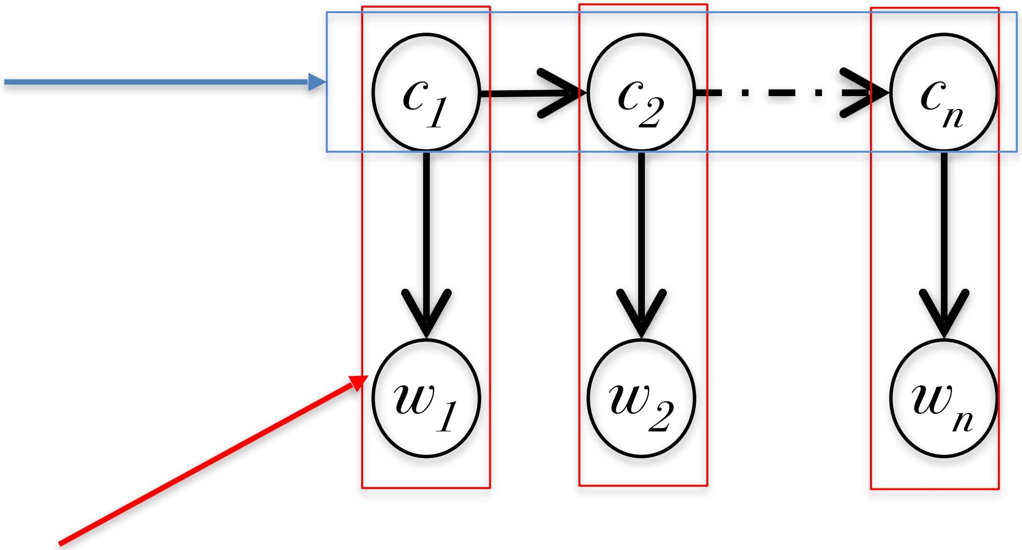
Observed word sequence

Model is like a sequence of Bayesian classifiers.



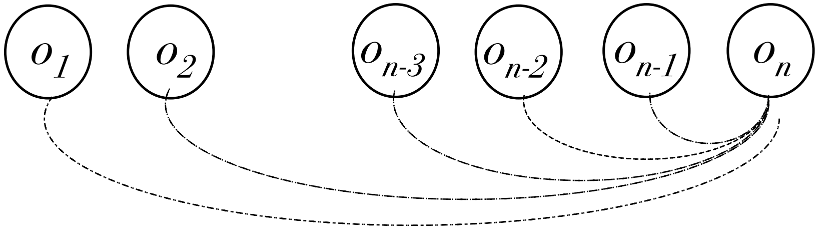
Hidden Markov Models

- HMMs use probability distributions from two models:
  - A class sequence model $p(c_i|c_1 \ldots c_{i-1})$ which is a Markov Model defined by **Transition probabilities** (like a language model)
  - A word/class association model $p(w_i|c_i)$ which are distributions of **Emission probabilities**



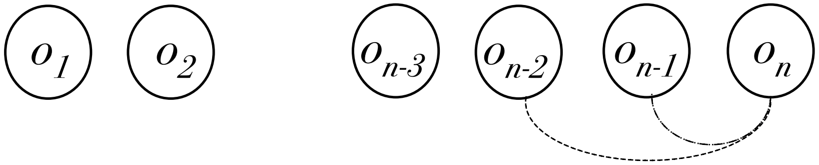
Markov Assumption

- To avoid sparsity (lack of observations), instead of:



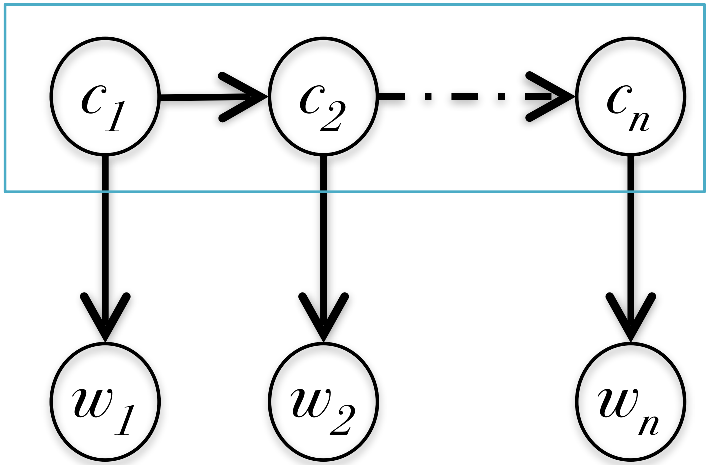
- We approximate by:
  - “n-gram model of length k” (where k = n-1)

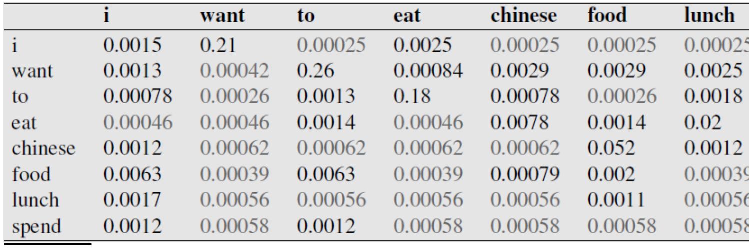
trigram model
(k=2):



Hidden Markov Models

- **Remember Language Models?**
- For the transition probabilities we can define a **Markov Model** (sequence likelihood model using the Markov assumption) which will give us the probability of a possible hidden sequence $C_1 \dots C_n$
- Remember the probability matrix for bigrams? i.e. **Transition matrix** for **transition probabilities.** For 1st order Markov Models, we can do this for class/state sequences too.





	i	want	to	eat	chinese	food	lunch
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058

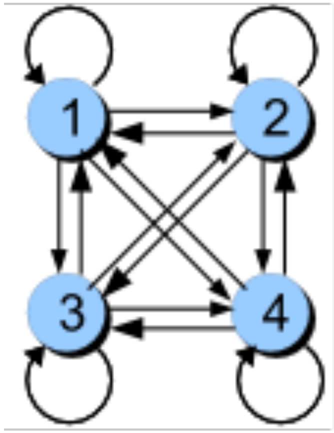
Hidden Markov Models

- **Transition matrix** constrains possible state paths:

C_i (state/class value at position i in sequence)

C_{i-1} (state/class value at position i-1 in sequence)

	c_1	**c_2**	**c_3**	**c_4**
c_1				
c_2				
c_3				
c_4				



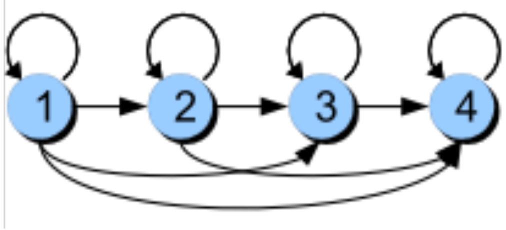
Hidden Markov Models

- **Transition matrix** constrains possible state paths:

C_i (state/class value at position i in sequence)

C_{i-1} (state/class value at position i-1 in sequence)

	c_1	**c_2**	**c_3**	**c_4**
c_1	▨	▨	▨	▨
c_2		▨	▨	▨
c_3			▨	▨
c_4				▨



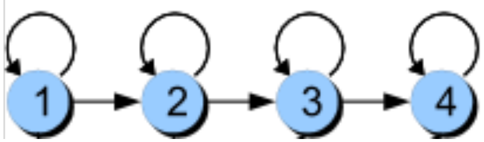
Hidden Markov Models

- **Transition matrix** constrains possible state paths:

$\mathbf{C_i}$ (state/class value at position i in sequence)

$\mathbf{C_{i-1}}$ (state/class value at position i-1 in sequence)

	$\mathbf{c_1}$	$\mathbf{c_2}$	$\mathbf{c_3}$	$\mathbf{c_4}$
$\mathbf{c_1}$	■	■		
$\mathbf{c_2}$		■	■	
$\mathbf{c_3}$			■	■
$\mathbf{c_4}$				■



Hidden Markov Models

- **Transition probabilities** $P(c_i|c_{i-1})$ define a 1st order Markov model of the current tag given the previous one.
- 1st order Markov models (bigram model) can be easily represented in a 2D transition matrix:

C_{i-1} (state/class value at position i-1 in sequence)

Transition probs $P(c_i|c_{i-1})$:

	NN	**NNS**	**VBZ**	**VB**	**end**
NN	0.3	0.3	0.3	0.0	0.1
NNS	0.0	0.2	0.6	0.2	0.0
VBZ	0.5	0.0	0.0	0.1	0.4
VB	0.3	0.5	0.0	0.0	0.2
start	0.3	0.3	0.0	0.4	0.0

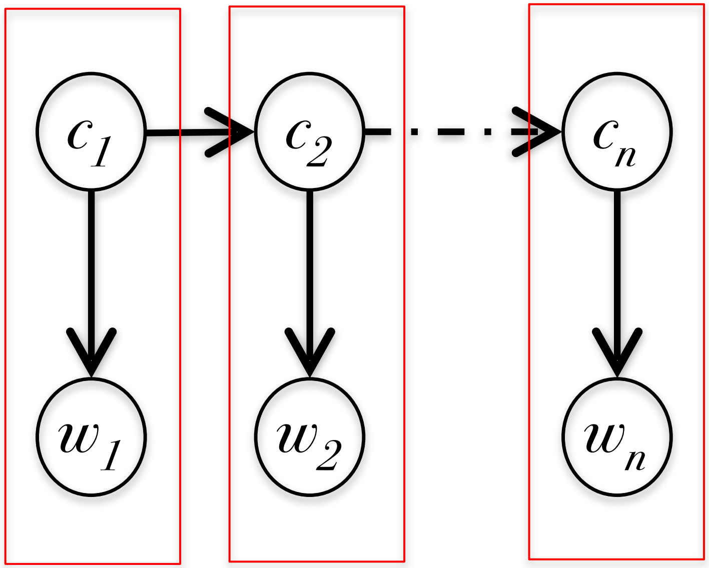
C_i (state/class value at position i in sequence)

Rows are distributions. Probabilities sum to 1.

- The class sequence is not directly observed, hence it is a **hidden** Markov model

Hidden Markov Models

- We can only estimate that a given sequence occurred based on what we **observe (observation sequence)**.
- **Emission probabilities** are needed for us to use Bayesian inference to answer: what is the likelihood that some underlying class *c* generated word *w* ?



Hidden Markov Models

- **Emission probabilities** can be defined in a matrix $P(w_i|c_i)$:

Emission probs $P(w_i|c_i)$:

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

C_i (state/class value at position i in sequence)

W_i (observation/word value at position i in sequence)

Rows are distributions over the vocab. Probabilities sum to 1.

- As with Naive Bayes, we 'flip' the probability around- 'time' was observed, so what's the likelihood that 'NN' **generated** it, or that 'NNS' generated it? etc. i.e. what is the likelihood of different hidden classes.

Hidden Markov Models

- Generative sequence model:
  - Assume observations (e.g. words) generated from **states**
  - States depend on previous state sequence (Markov assumption: just the most recent one, or fixed number in the past)
- Likelihood of observations given hidden class sequence generated by **bigram (first order)** underlying model:

$$P(W) = P(w_1, w_2, \ldots, w_n) = \prod_i p(w_i | c_i) p(c_i | c_{i-1})$$

- Bayes' Rule lets us use it to estimate likelihood of a class sequence given we know the word sequence:

$$P\left(C | W\right) = \frac{P(W \middle| C) P(C)}{P(W)}$$

And from this we have a classifier for tagging word sequences:

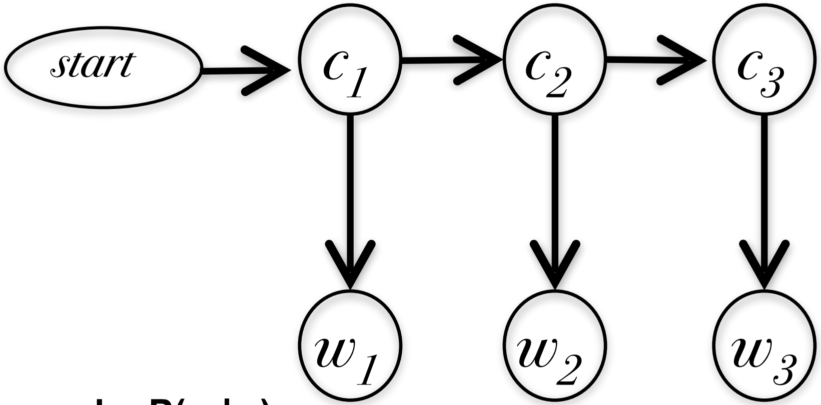
$$C_{MAP} = argmax_C \, p(C|W) = argmax_C \, p(W|C)p(C)$$

Probability calculations

- Given HMM H, what kind of probabilities are available?

```
W = time flies like an arrow
C =  NN    VBZ   PRP DT  NN

W = fruit flies like a banana
C =   NN     NNS   VB DT  NN
```



Transition probs $P(c_i|c_{i-1})$:

c_i (columns), **c_{i-1}** (rows)

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i (columns), **c_i** (rows)

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

What are:

p(c2=VBZ|c1=NN)
p(c2=NNS|c1=NN)
p(w1=fruit|c1=NN)
p(w1=flies|c1=VBZ)

More difficult, what are:

p(w1=fruit)
p(w1=time)
p(c1=NN|w1=time)

Probability calculations

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

What are:

p(c2=VBZ|c1=NN)

p(c2=NNS|c1=NN)

p(w1=fruit|c1=NN)

p(w1=flies|c1=VBZ)

Probability calculations

- (Solution)

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

What are:

p(c2=VBZ|c1=NN) 0.4

p(c2=NNS|c1=NN)

p(w1=fruit|c1=NN)

p(w1=flies|c1=VBZ)

Probability calculations

- (Solution)

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

What are:

p(c2=VBZ|c1=NN) 0.4

p(c2=NNS|c1=NN) 0.2

p(w1=fruit|c1=NN)

p(w1=flies|c1=VBZ)

Probability calculations

- (Solution)

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

What are:

p(c2=VBZ|c1=NN) 0.4

p(c2=NNS|c1=NN) 0.2

p(w1=fruit|c1=NN) 0.3

p(w1=flies|c1=VBZ)

Probability calculations

- (Solution)

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

What are:

p(c2=VBZ|c1=NN)	**0.4**
p(c2=NNS|c1=NN)	**0.2**
p(w1=fruit|c1=NN)	**0.3**
p(w1=flies|c1=VBZ)	**1.0**

Probability calculations

- (Solution)

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

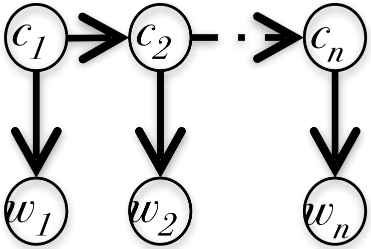
What are:

p(c2=VBZ|c1=NN)	0.4
p(c2=NNS|c1=NN)	0.2
p(w1=fruit|c1=NN)	0.3
p(w1=flies|c1=VBZ)	1.0

Only simple look-up required!

Likelihood of Observed Sequence (words)

- **Likelihood:** given observation W and HMM H, what is the likelihood p(W|H)?



- If we knew the class sequence, we could use:

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid c_i) P(c_i \mid c_{i-1})$$

- But we don't …
  - HMM classes are hidden/unseen: **"latent variables"**

$$P(w_1 w_2 \ldots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

...

Likelihood of Observed Sequence (words)

More difficult, what are:

p(w1=fruit)

p(w1=time)

$$P(w_1 w_2 \ldots w_n) = \sum_{j\in C}\prod_{i} P(w_i \mid c_i^j)\,P(c_i^j \mid c_{i-1}^j)$$

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \ldots w_n) = \sum_{j\in C} \prod_{i} P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

More difficult, what are:

p(w1=fruit)

= p(w1=fruit|c1=NN) * p(c1=NN|c0=start) +
p(w1=fruit|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=fruit|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=fruit|c1=VB) * p(c1=VB|c0=start) +
p(w1=fruit|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=fruit|c1=DT) * p(c1=DT|c0=start)

Transition probs $P(c_i|c_{i-1})$:

c_i (columns), c_{i-1} (rows)

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i (columns), c_i (rows)

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \dots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

→

More difficult, what are:

p(w1=fruit)

= p(w1=fruit|c1=NN) * p(c1=NN|c0=start) +
p(w1=fruit|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=fruit|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=fruit|c1=VB) * p(c1=VB|c0=start) +
p(w1=fruit|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=fruit|c1=DT) * p(c1=DT|c0=start)

Transition probs $P(c_i|c_{i-1})$:

c_i (columns), c_{i-1} (rows)

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i (columns), c_i (rows)

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \ldots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

More difficult, what are:

p(w1=fruit)

= p(w1=fruit|c1=NN) * p(c1=NN|c0=start) +
p(w1=fruit|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=fruit|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=fruit|c1=VB) * p(c1=VB|c0=start) +
p(w1=fruit|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=fruit|c1=DT) * p(c1=DT|c0=start)

Transition probs $P(c_i|c_{i-1})$:

c_i (columns), c_{i-1} (rows)

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i (columns), c_i (rows)

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \ldots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

More difficult, what are:

p(w1=fruit)

= p(w1=fruit|c1=NN) * p(c1=NN|c0=start) +
p(w1=fruit|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=fruit|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=fruit|c1=VB) * p(c1=VB|c0=start) +
p(w1=fruit|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=fruit|c1=DT) * p(c1=DT|c0=start)

= (0.3 * 0.2) +
(0.0 * 0.2) +
(0.0 * 0.0) +
(0.0 * 0.1) +
(0.0 * 0.0) +
(0.0 * 0.5)

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \ldots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

More difficult, what are:

p(w1=fruit)

= p(w1=fruit|c1=NN) * p(c1=NN|c0=start) +
p(w1=fruit|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=fruit|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=fruit|c1=VB) * p(c1=VB|c0=start) +
p(w1=fruit|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=fruit|c1=DT) * p(c1=DT|c0=start)

= (0.3 * 0.2) +
(0.0 * 0.2) +
(0.0 * 0.0) +
(0.0 * 0.1) +
(0.0 * 0.0) +
(0.0 * 0.5)

= 0.06 +
0 +
0 +
0 +
0 +
0

Transition probs $P(c_i|c_{i-1})$:

c_i (columns), c_{i-1} (rows)

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i (columns), c_i (rows)

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \dots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

More difficult, what are:

p(w1=fruit)

= p(w1=fruit|c1=NN) * p(c1=NN|c0=start) +
p(w1=fruit|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=fruit|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=fruit|c1=VB) * p(c1=VB|c0=start) +
p(w1=fruit|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=fruit|c1=DT) * p(c1=DT|c0=start)

= (0.3 * 0.2) +
(0.0 * 0.2) +
(0.0 * 0.0) +
(0.0 * 0.1) +
(0.0 * 0.0) +
(0.0 * 0.5)

= 0.06 +
0 +
0 +
0 +
0 +
0

= 0.06

Likelihood of Observed Sequence (words)

More difficult, what are:

p(w1=time)

- (Solution)

$$P(w_1 w_2 \ldots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

Transition probs $P(c_i|c_{i-1})$:

c_i (columns), **c_{i-1}** (rows)

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i (columns), **c_i** (rows)

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \dots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

More difficult, what are:

p(w1=time)
= p(w1=time|c1=NN) * p(c1=NN|c0=start) +
p(w1=time|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=time|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=time|c1=VB) * p(c1=VB|c0=start) +
p(w1=time|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=time|c1=DT) * p(c1=DT|c0=start)

Transition probs $P(c_i|c_{i-1})$:

c_i (columns), **c_{i-1}** (rows)

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i (columns), **c_i** (rows)

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \ldots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

More difficult, what are:

p(w1=time)

= p(w1=time|c1=NN) * p(c1=NN|c0=start) +
p(w1=time|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=time|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=time|c1=VB) * p(c1=VB|c0=start) +
p(w1=time|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=time|c1=DT) * p(c1=DT|c0=start)

Transition probs $P(c_i|c_{i-1})$:

c_i (columns), c_{i-1} (rows)

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i (columns), c_i (rows)

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \ldots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

More difficult, what are:

p(w1=time)

= p(w1=time|c1=NN) * p(c1=NN|c0=start) +
p(w1=time|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=time|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=time|c1=VB) * p(c1=VB|c0=start) +
p(w1=time|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=time|c1=DT) * p(c1=DT|c0=start)

Transition probs $P(c_i|c_{i-1})$:

c_i (columns), c_{i-1} (rows)

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i (columns), c_i (rows)

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \dots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

More difficult, what are:

p(w1=time)

= p(w1=time|c1=NN) * p(c1=NN|c0=start) +
p(w1=time|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=time|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=time|c1=VB) * p(c1=VB|c0=start) +
p(w1=time|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=time|c1=DT) * p(c1=DT|c0=start)

= (0.3 * 0.2) +
(0.0 * 0.2) +
(0.0 * 0.0) +
(0.2 * 0.1) +
(0.0 * 0.0) +
(0.0 * 0.5)

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \ldots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

More difficult, what are:

p(w1=time)

= p(w1=time|c1=NN) * p(c1=NN|c0=start) +
p(w1=time|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=time|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=time|c1=VB) * p(c1=VB|c0=start) +
p(w1=time|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=time|c1=DT) * p(c1=DT|c0=start)

= (0.3 * 0.2) +
(0.0 * 0.2) +
(0.0 * 0.0) +
(0.2 * 0.1) +
(0.0 * 0.0) +
(0.0 * 0.5)

= 0.06 +
0 +
0 +
0.02 +
0 +
0

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Likelihood of Observed Sequence (words)

- (Solution)

$$P(w_1 w_2 \ldots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

More difficult, what are:

p(w1=time)

= p(w1=time|c1=NN) * p(c1=NN|c0=start) +
p(w1=time|c1=NNS) * p(c1=NNS|c0=start) +
p(w1=time|c1=VBZ) * p(c1=VBZ|c0=start) +
p(w1=time|c1=VB) * p(c1=VB|c0=start) +
p(w1=time|c1=PRP) * p(c1=PRP|c0=start) +
p(w1=time|c1=DT) * p(c1=DT|c0=start)

= (0.3 * 0.2) +
(0.0 * 0.2) +
(0.0 * 0.0) +
(0.2 * 0.1) +
(0.0 * 0.0) +
(0.0 * 0.5)

= 0.06 +
0 +
0 +
0.02 +
0 +
0

= 0.08

Posterior Probability of Latent Variable (Class) sequence

- (Solution)

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

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid c_i) P(c_i \mid c_{i-1})$$

More difficult, what are:

p(c1=NN|w1=time)
(where p(w1=time) = 0.08 from earlier!)
= (p(w1=time|c1=NN) * p(c1=NN|c0=start)) / 0.08

= (0.3 * 0.2) / 0.08

= 0.75

Transition probs $P(c_i|c_{i-1})$:

c_i (columns), c_{i-1} (rows)

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i (columns), c_i (rows)

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Scaling up to Sequences

- We can do these calculations in this way for short sequences for small numbers of states.

- However, summing all possible class sequences is exponential, so use **dynamic programming**
  - we use the **Forward algorithm**
  - $\alpha_n(j)$ = probability of getting to word n and being in state j

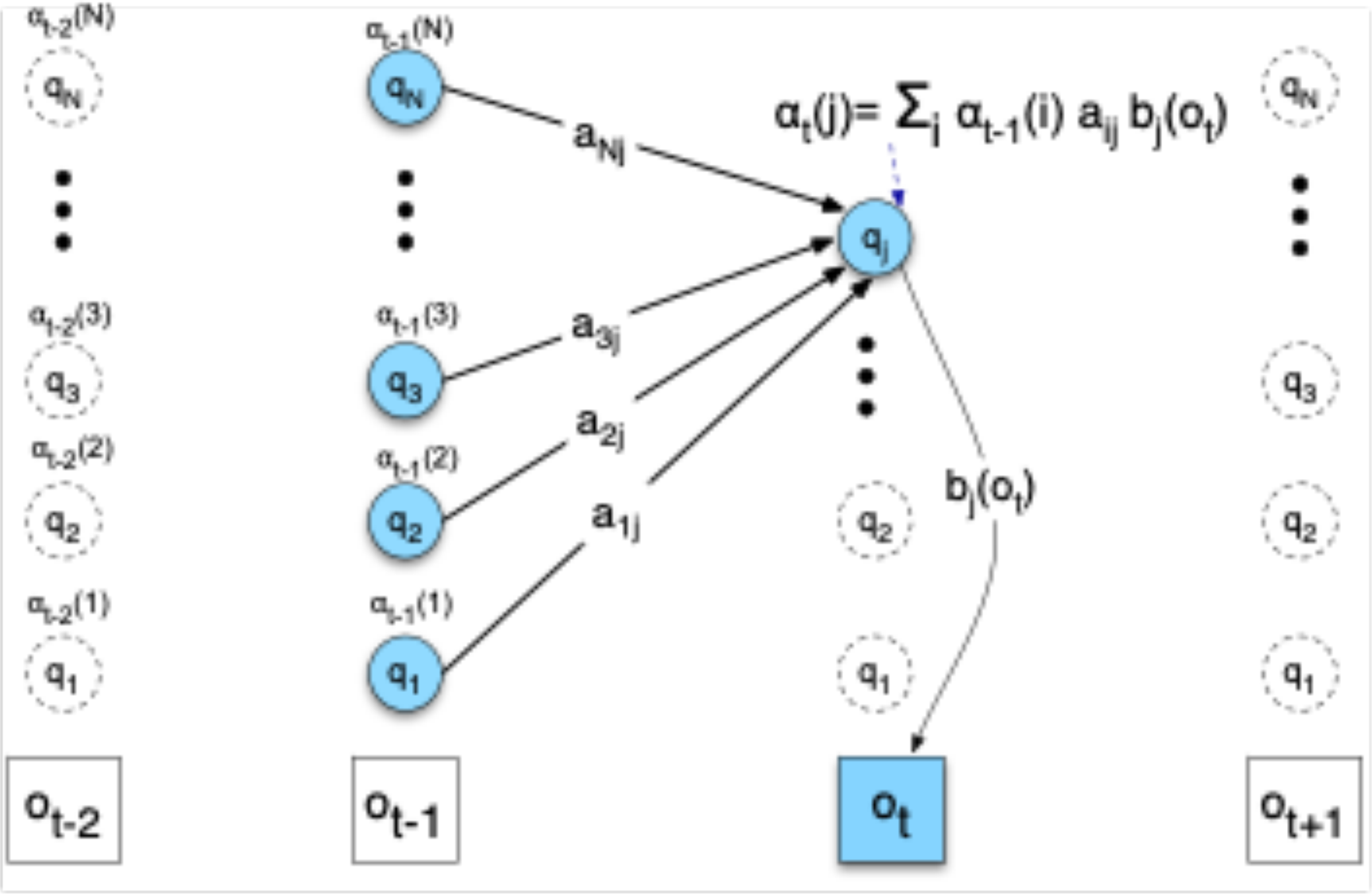
$$\alpha_1(j) = P(w_1 c_j) = P(w_1 \mid c_j) P(c_j)$$

$$\alpha_2(j) = P(w_1 w_2 c_j) = P(w_2 \mid c_j) \sum_i P(c_j \mid c_i) \alpha_1(i)$$

$$\alpha_n(j) = P(w_1 w_2 \ldots w_n c_j) = P(w_n \mid c_j) \sum_i P(c_j \mid c_i) \alpha_{n-1}(i)$$

$\ldots$

Forward algorithm



Decoding- getting the most likely sequence

- **Decoding:** given observations W, what is the most likely state sequence C?
  - $C_{MAP} = \text{argmax}_C\ p(C|W) = \text{argmax}_C\ p(W|C)p(C)$
  - No need to calculate p(W) for classification.
- As a start, let's compare two possible sequences C1, C2 (not all of them):

W	=	time	flies	like	an	arrow
C1	=	NN	VBZ	PRP	DT	NN
C2	=	NN	NNS	PRP	DT	NN

p(W=<time, flies, like, an, arrow>| C=<NN, **VBZ**, PRP, DT, NN>) *
p(C=<NN, **VBZ**, PRP, DT, NN>) =

p(w1=time|c1=NN) * p(c1=NN|c0=start) *
p(w2=flies|c2=VBZ) * p(c2=VBZ|c1=NN) *
p(w3=like|c3=PRP) * **p(c3=PRP|c2=VBZ) ***
p(w4=an|c4=DT) * p(c4=DT|c3=PRP) *
p(w5=arrow|c5=NN) * p(c5=NN|c4=DT)

p(W=<time, flies, like, an, arrow>| C=<NN, **NNS**, PRP, DT, NN>) *
p(C=<NN, **NNS**, PRP, DT, NN>) =

p(w1=time|c1=NN) * p(c1=NN|c0=start) *
p(w2=flies|c2=NNS) * p(c2=NNS|c1=NN) *
p(w3=like|c3=PRP) * **p(c3=PRP|c2=NNS) ***
p(w4=an|c4=DT) * p(c4=DT|c3=PRP) *
p(w5=arrow|c5=NN) * p(c5=NN|c4=DT)

W	=	time	flies	like	an	arrow	
C1	=	NN	VBZ	PRP	DT	NN	= 0.00144
C2	=	NN	NNS	PRP	DT	NN	

p(W=<time, flies, like, an, arrow>| C=<NN, **VBZ**, PRP, DT, NN>) *
p(C=<NN, **VBZ**, PRP, DT, NN>)

= p(w1=time|c1=NN) * p(c1=NN|c0=start) *
p(w2=flies|c2=VBZ) * p(c2=VBZ|c1=NN) *
p(w3=like|c3=PRP) * **p(c3=PRP|c2=VBZ) ***
p(w4=an|c4=DT) * p(c4=DT|c3=PRP) *
p(w5=arrow|c5=NN) * p(c5=NN|c4=DT)

= 0.3 * 0.2 *
1.0 * 0.4 *
1.0 * **0.5 ***
1.0 * 0.6 *
0.4 * 0.5
= 0.00144

Transition probs $P(c_i|c_{i-1})$:

c_i

c_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs $P(w_i|c_i)$:

w_i

c_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

W = time flies like an arrow

C1 = NN VBZ PRP DT NN = 0.00144

C2 = NN NNS PRP DT NN = 0

p(W=<time, flies, like, an, arrow>| C=<NN, **NNS**, PRP, DT, NN>) *
p(C=<NN, **NNS**, PRP, DT, NN>)

= p(w1=time|c1=NN) * p(c1=NN|c0=start) *
p(w2=flies|c2=NNS) * p(c2=NNS|c1=NN) *
p(w3=like|c3=PRP) * **p(c3=PRP|c2=NNS) ***
p(w4=an|c4=DT) * p(c4=DT|c3=PRP) *
p(w5=arrow|c5=NN) * p(c5=NN|c4=DT)

= 0.3 * 0.2 *
1.0 * **0.2** *
1.0 * **0** *
1.0 * 0.6 *
0.4 * 0.5
= 0

Transition probs P(c_i|c_{i-1}):

C_i

C_{i-1}

	NN	NNS	VBZ	VB	PRP	DT
NN	0.2	0.2	0.4	0.2	0.0	0.0
NNS	0.0	0.1	0.5	0.4	0.0	0.0
VBZ	0.1	0.1	0.0	0.0	0.5	0.3
VB	0.2	0.2	0.0	0.0	0.1	0.5
PRP	0.2	0.2	0.0	0.0	0.0	0.6
DT	0.5	0.5	0.0	0.0	0.0	0.0
start	0.2	0.2	0.0	0.1	0.0	0.5

Emission probs P(w_i|c_i):

w_i

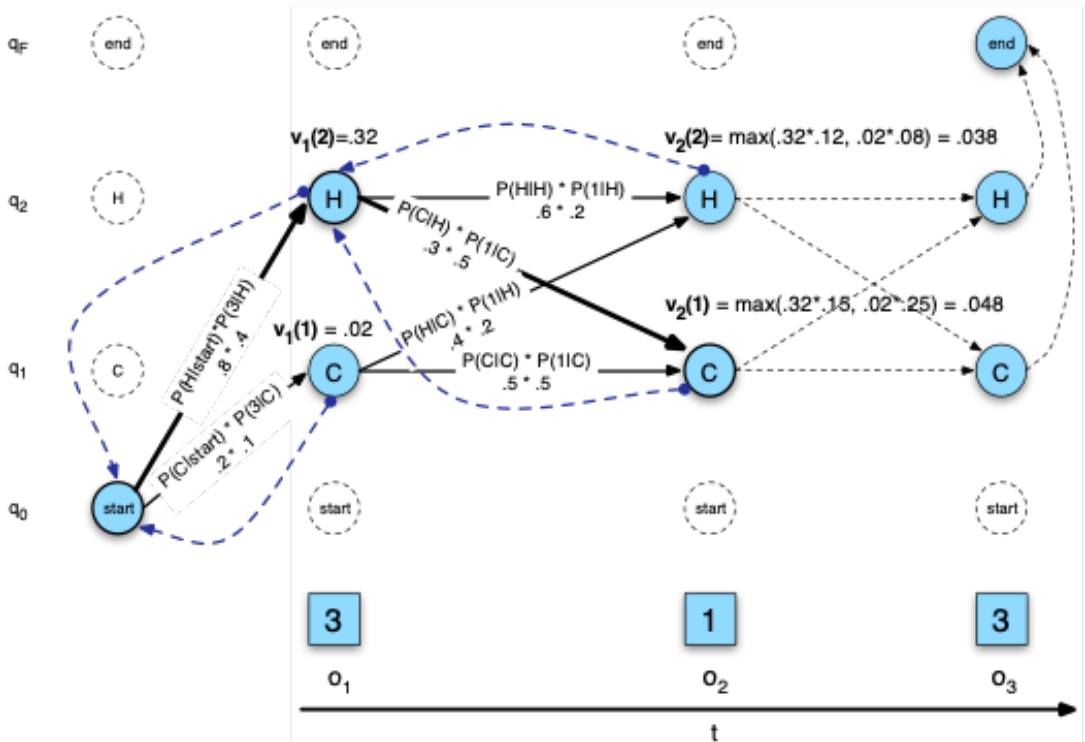
C_i

	time	fruit	flies	arrow	like	an
NN	0.3	0.3	0.0	0.4	0.0	0.0
NNS	0.0	0.0	1.0	0.0	0.0	0.0
VBZ	0.0	0.0	1.0	0.0	0.0	0.0
VB	0.2	0.0	0.0	0.0	0.8	0.0
PRP	0.0	0.0	0.0	0.0	1.0	0.0
DT	0.0	0.0	0.0	0.0	0.0	1.0

Decoding- getting the most likely sequence automatically

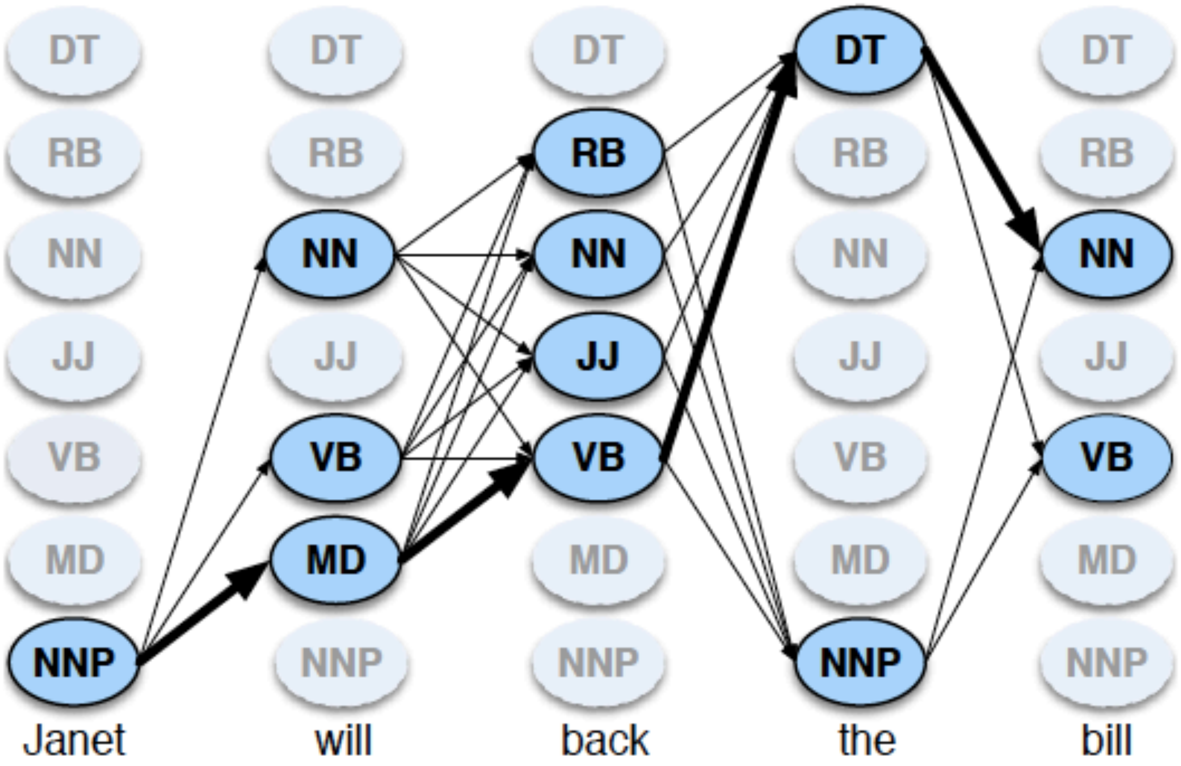
- Searching over all possible tag sequences to get the argmax$_C$ p(W|C)p(C) is exponential in the length of the sequence T.
- Use the **Viterbi algorithm** - dynamic programming reduces state sequences to search hugely from exponential $|S|^T$ to polynomial quadratic $|S|^2 * T$
  - **Beam search** also possible to reduce this search further- keep only the k most likely sequences after each word (keep these in the beam).

- Viterbi is similar to Forward algorithm, but maintain **back-pointer** from each state to most likely previous state
- Then **retrace** from most likely final state

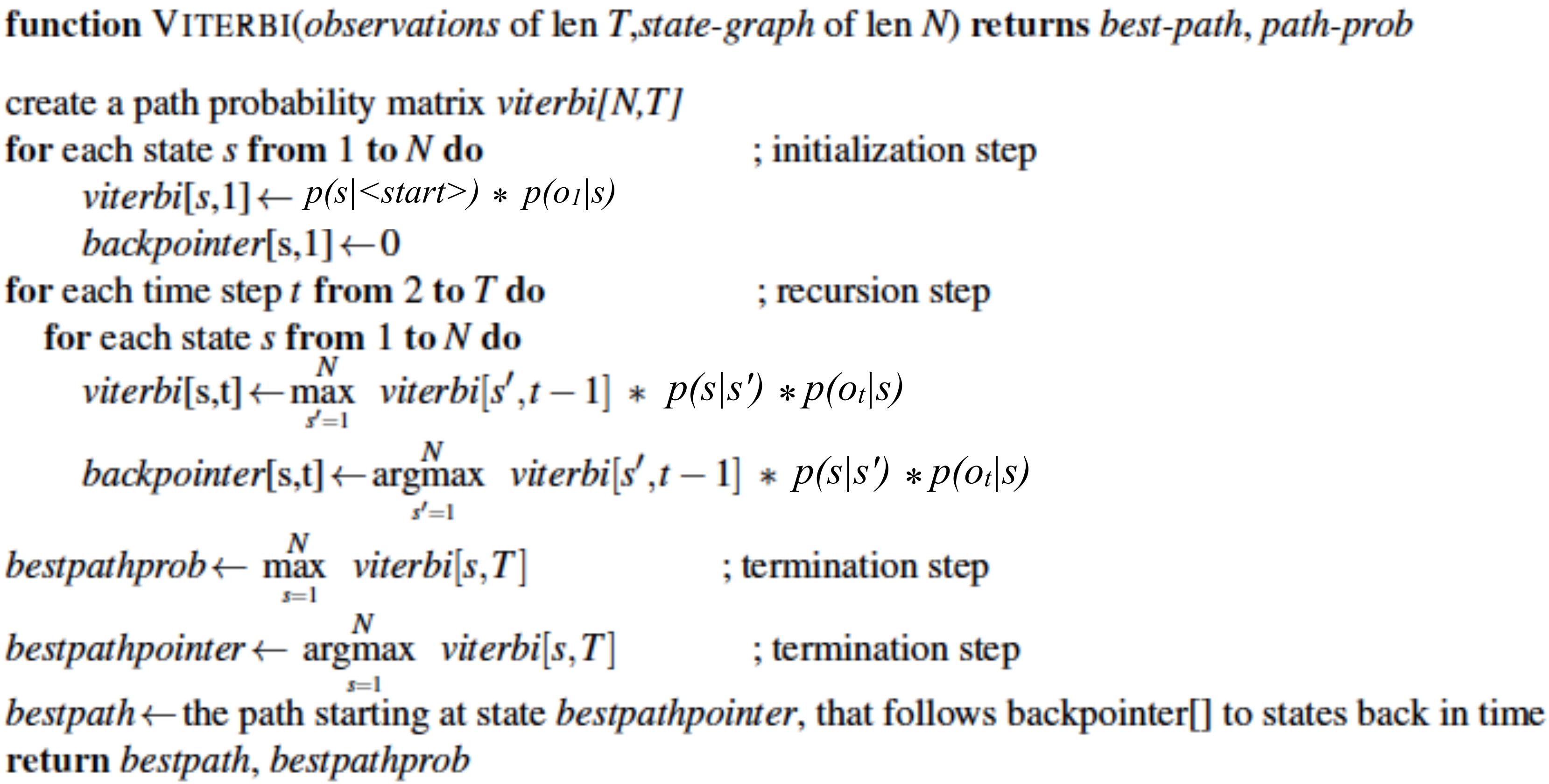


Decoding- getting the most likely sequence automatically

- The Viterbi algorithm sets up a matrix of size *[N, T]* where N = number of possible states (tags) and T is the length of the sequence of observations (words).
- The idea is to find the state path with the highest likelihood given the words - see thickest path below, 0-prob paths greyed out:



Decoding- getting the most likely sequence automatically



```
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] ← $p(s|<start>) * p(o_1|s)$
    backpointer[s,1] ← 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] ← $\max_{s'=1}^{N}$ $viterbi[s',t-1] * p(s|s') * p(o_t|s)$
    backpointer[s,t] ← $\operatorname{argmax}_{s'=1}^{N}$ $viterbi[s',t-1] * p(s|s') * p(o_t|s)$
bestpathprob ← $\max_{s=1}^{N}$ $viterbi[s,T]$                     ; termination step
bestpathpointer ← $\operatorname{argmax}_{s=1}^{N}$ $viterbi[s,T]$           ; termination step
bestpath ← the path starting at state bestpathpointer, that follows backpointer[] to states back in time
return bestpath, bestpathprob
```

Learning

- **Learning/training:** given observation sequence of words W, what is the optimum HMM model H? i.e. what are the optimal emission and transition probability models?
- If we have training data with fully labelled sequences, use standard **Maximum likelihood estimation (MLE)** with counts *C* from training data to get the conditional probabilities:
  - Emission probabilities: word at position *i* given tag at position *i*

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

  - Transition probabilities: tag at position *i* given tag at position *i-1*

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

- e.g. emission prob of word 'will' given an MD $P(will|MD) = \frac{C(MD, will)}{C(MD)} = \frac{4046}{13124} = .31$
- e.g. transition prob of tag VB following tag MD:

$$P(VB|MD) = \frac{C(MD, VB)}{C(MD)} = \frac{10471}{13124} = .80$$

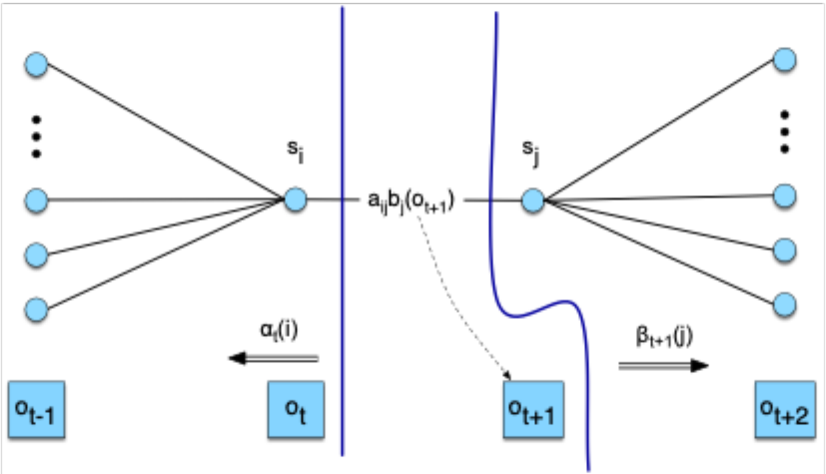
Learning

- Potential for lots of 0s in decoding. We can of course **smooth** these estimates to avoid 0s and not overfit the data.

(See Python notebook book for HMM POS tagging)

Learning

- What if we don’t have fully labelled data?
- We use the **Forward-Backward (Baum-Welch) algorithm**
  - Similar to Forward algorithm, but combine:
    - Forward probability of getting to this state i at time t from start: $\alpha_t(i)$
    - Backward probability of getting from next state j and next time step t+1 to the end: $\beta_{t+1}(j)$
  - Iterate and update these until probability of observations is maximised and cannot improve (**convergence**).
  - (wait for parsing lecture)



Generalising HMMs

- We've only looked at 1st order (bigram) Markov models, largely because their transition probabilities are easy to show in a 2D matrix. What if it made sense for the underlying model to use other previous states (not just the last one)?

- It is possible to generalise the Hidden Markov Model to an **arbitrary order** (see n-grams in language modelling lecture), e.g. tri-gram:

$$P(t_i|t_{i-1}, t_{i-2}) = \frac{C(t_{i-2}, t_{i-1}, t_i)}{C(t_{i-2}, t_{i-1})}$$

- We can use back-off and interpolation of lower order models just like we did with n-gram language models.

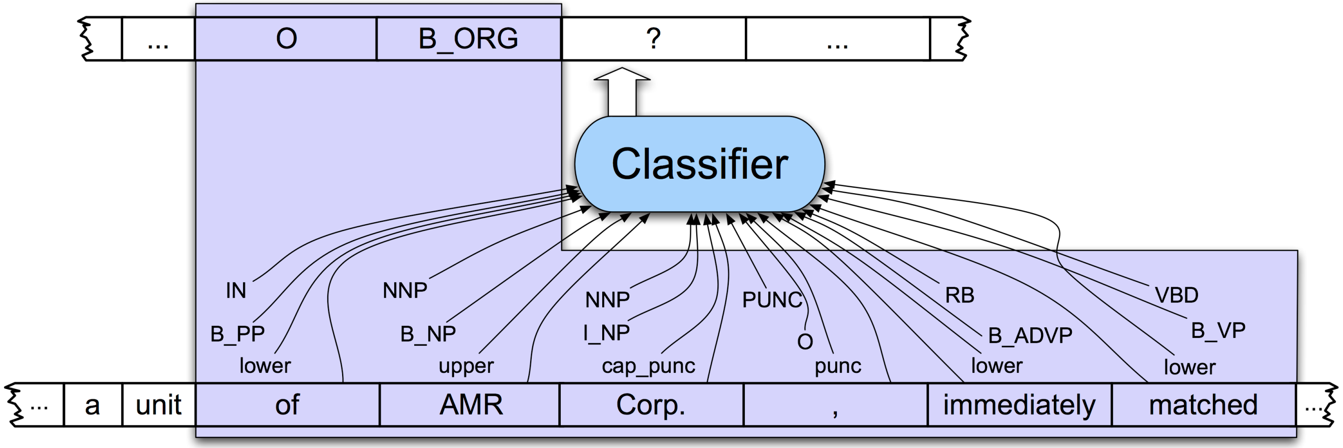
- However, this complicates Viterbi, as it requires going over all possible combinations of the last 3 states (not just 2), making the complexity $|S|^3 * T$.

OUTLINE

1) Sequence Tagging Tasks: POS tagging and NER
2) Generative: Hidden Markov Models
3) Discriminative: Conditional Random Fields

Discriminative Sequence Classification

- Can we use a **discriminative** approach instead? Remember alternative text classification methods:
  - Naïve Bayes: generative – argmax$_C$ *p(X|C)p(C)*
  - Logistic Regression/SVM: discriminative – argmax$_C$ *p(C|X)* directly, allows **many more features** to be used without having to estimate *p(X),* which isn't needed for classification anyway.
- How do we make this change for a sequence model?



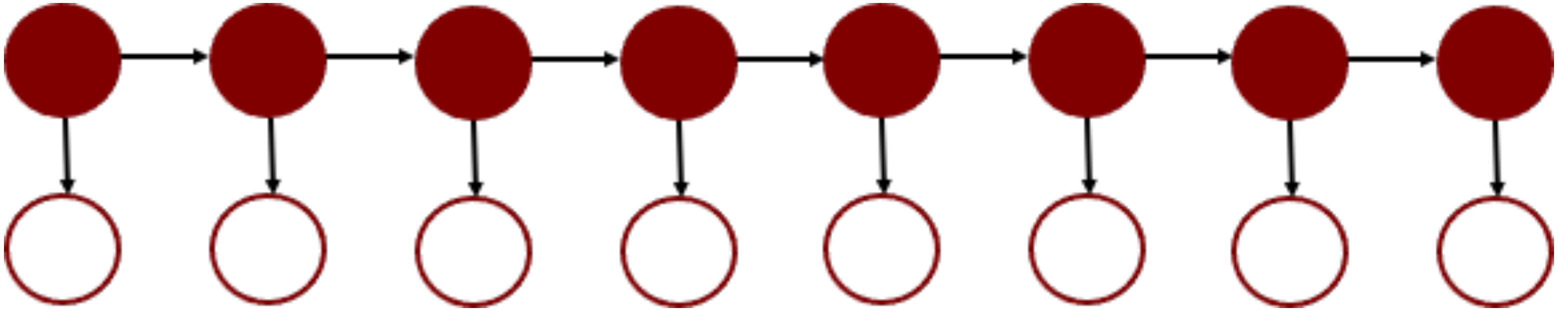
Discriminative Sequence Classification

- The difficulty in modelling p(X|C) is that it often contains many **highly dependent features** that are difficult to model:
    - e.g. in NER, a naive application of an HMM relies on only one feature, the word's identity, but many words, especially proper names, will not have occurred in the training set, so the word-identity feature is uninformative.

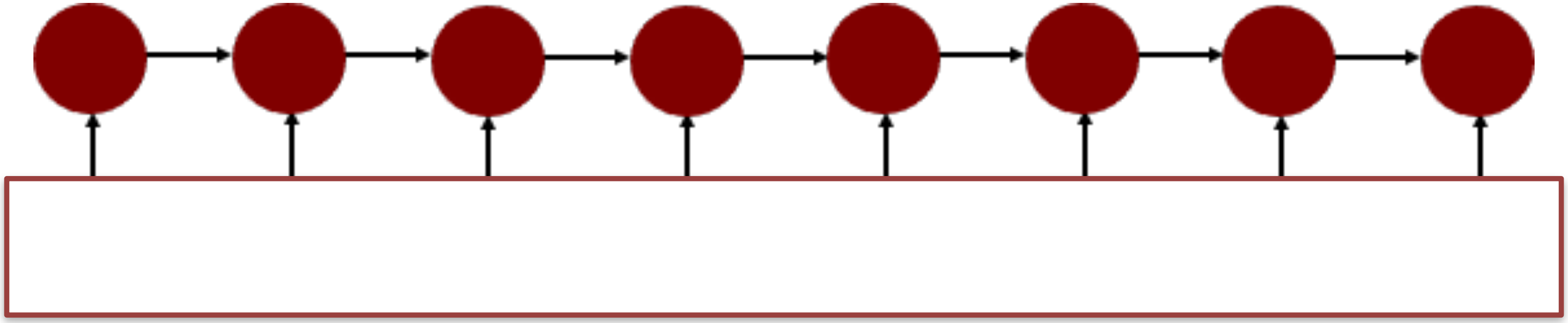
- The principal advantage of discriminative modelling is that it is better suited to including **rich, overlapping features** which can given information even if a word is unknown:
    - e.g. in NER, to label unseen words, we would like to exploit other features such as capitalization, neighboring words, affixes, membership in predetermined lists of people and locations etc.

Discriminative Sequence Classification

- HMM:

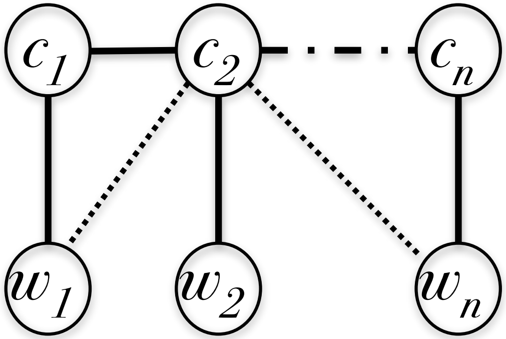


- (Linear-chain) Conditional Random Field (CRF):



Conditional Random Fields

- Conditional Random Fields (CRF), discriminative Markov models.
  - HMM (generative):
    C_{MAP} = argmax$_C$ p(C|W) = argmax$_C$ p(W|C)p(C)
  - CRF (discriminative):
    C_{MAP} = argmax$_C$ p(C|W)



$$p(C \mid W) = \frac{1}{Z}\prod_i \exp(\sum_j \lambda_j f_j(y_{i-1}, y_i, W, i))$$

  - Define **feature function** *f* which returns a set of features for a sequence position *i*:
    - e.g. f_i = {"w_{i-1} = fruit, w_i = flies, c_{i-1} = NN, c_i = NNS"}
  - Learn optimal weights λ which apply to each feature f_j through a **gradient descent** method like L-BFGS.

Conditional Random Fields

- A CRF model consists of
  - $\mathbf{F} = \langle f_1, \ldots, f_k \rangle$, a vector of “feature functions”
  - $\boldsymbol{\theta} = \langle \theta_1, \ldots, \theta_k \rangle$, a vector of weights for each feature function.

- Let **O** = $\langle o_1, \ldots, o_T \rangle$ be an observed sentence
- Let **A** = $\langle a_1, \ldots, a_T \rangle$ be the latent variables (i.e. sequence tags).

$$P(\mathbf{A} = \mathbf{y} \mid \mathbf{O}) = \frac{\exp\left(\boldsymbol{\theta} \cdot \mathbf{F}\left(\mathbf{y}, \mathbf{O}\right)\right)}{\sum_{\mathbf{y}'} \exp\left(\boldsymbol{\theta} \cdot \mathbf{F}\left(\mathbf{y}', \mathbf{O}\right)\right)}$$

- This is the same as the Maximum Entropy equation.

Finding the Best Sequence

- Best sequence *y* is:

$$\begin{aligned}\arg\max_{\mathbf{y}} P(\mathbf{A}=\mathbf{y}\mid\mathbf{O}) &= \arg\max_{\mathbf{y}}\left[\frac{1}{Z(\mathbf{O})}\exp\left(\boldsymbol{\theta}\cdot\mathbf{F}\left(\mathbf{y},\mathbf{O}\right)\right)\right] \\ &= \arg\max_{\mathbf{y}}\left[\boldsymbol{\theta}\cdot\mathbf{F}\left(\mathbf{y},\mathbf{O}\right)\right]\end{aligned}$$

- Recall from HMM discussion, if there are:
  - *K* possible states for each y_i variable,
  - *N total* y_i *variables,*

  Then there are K^N possible settings for *y*

- **So brute force can’t find the best sequence.**

- **Instead, we resort to a Viterbi-like dynamic program.**

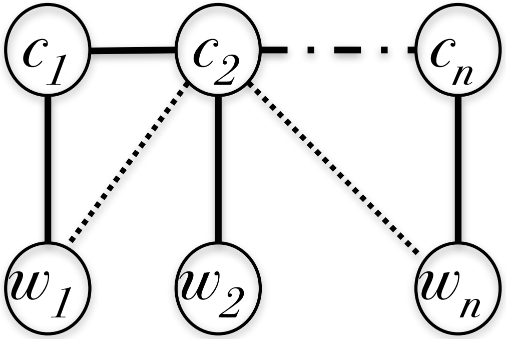
Training/optimizing CRFs

- In defining a CRF model, you have to consider:
  - **The feature function:** what kind of features do you want to extract for each step in the sequence? These can include previous/future words as input into the current time-step, and include features like **‘word-shape’** (e.g. XX-XXX), boolean values for **capitalisation** etc.
  - **Min. document frequency** for features (can be quite high like 5+ as many features can be extracted).
  - The shape of the **Markov model** for the labels- most commonly used in NLP is the **linear chain CRF**- much like a bigram language model/first order HMM, just connecting one state to the next.
  - **Regularisation** parameters (L1 and L2), sometimes called ‘C1’ and ‘C2’ in CRF.
  - **Learning algorithm** (usually a gradient descent method).

Conditional Random Fields

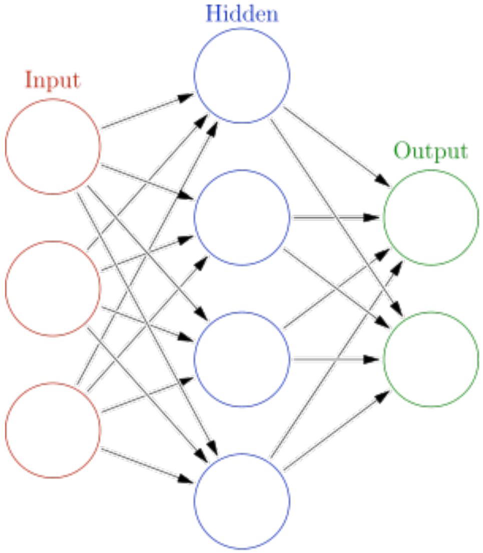
- Advantages:
  - – You can define (nearly) arbitrary features
  - – Often outperform HMMs
  - – Available implementations e.g. NLTK CRF tagger

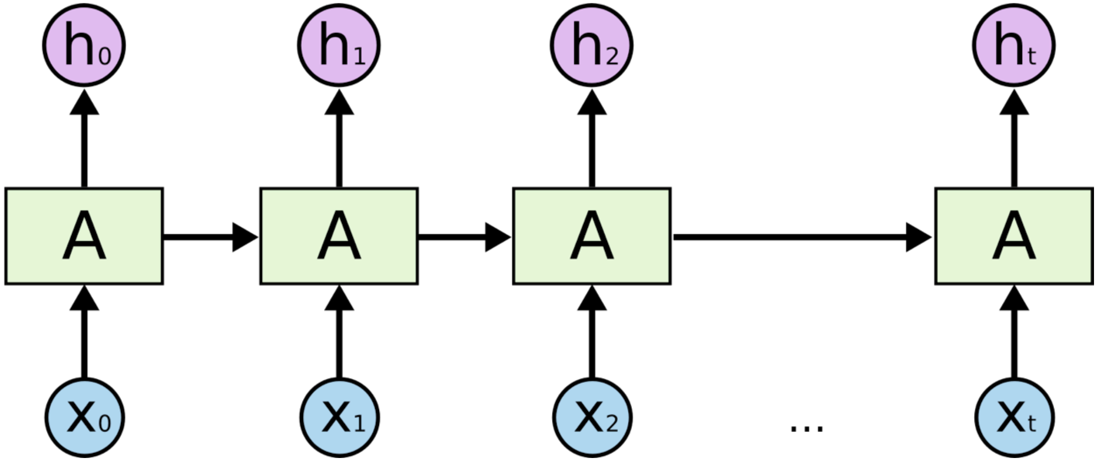
- Disadvantages:
  - – Complex inference (dynamic programming again)
  - – Needs manual definition of features
  - – Output is not a sequence probability
    - it's the confidence of sequence given the data
  - – (i.e. it's not really a language model)



- In general, this is **structured prediction** rather than **classification**
  - – Predicting structured objects not just classes/values

Extra: Recurrent Neural Networks





(Unit on Neural Nets and course next semester!)

Sequence Classification

- Hidden Markov Models
  - Like Language Models, use Markov Models of a given order.
  - Though the Markov Model not directly observed.
  - ‘Flip’ the sequence likelihoods round in a Bayesian style.
  - Robust, good baseline for sequence tagging tasks
  - Learnable without much labelled data
  - Be careful with smoothing!

- Conditional Random Fields / Recurrent Neural Nets
  - Discriminative: higher accuracy for many tasks
  - More complex learning; need more data
  - Can be more complex feature definition process
  - Be careful with regularisation, weighting, activation functions, …

Reading

- Jurafsky and Martin (3rd Ed. online):
  - Chapter 8 (HMMs and CRFs for POS tagging/ NER)
  - Appendix A (HMMs in detail)
- (Optional) Manning and Schuetze (1999):
  - Chapter 9 (Markov Models)
  - Chapter 10 (POS tagging & HMMs)