Sequence Labeling I

POS Tagging with Hidden Markov Models

Mausam

(Slides based on Michael Collins, Dan Klein, Chris Manning, Dan Jurafsky, Heng Ji, Luke Zettlemoyer, Alex Simma, Erik Sudderth, David Fernandez-Baca, Drena Dobbs, Serafim Batzoglou, William Cohen, Andrew McCallum, Dan Weld)

Sequence problems

- Many problems in NLP have data which is a sequence of characters, words, phrases, lines, or sentences …
- We can think of our task as one of labeling each item

VBG	NN	IN	DT	NN	IN	NN
Chasing	opportunity	in	an	age	of	upheaval

POS tagging

B	B	I	I	B	I	B	I	B	B
而	相	对	于	这	些	品	牌	的	价

Word segmentation

PERS	O	O	O	ORG	ORG
Murdoch	discusses	future	of	News	Corp.

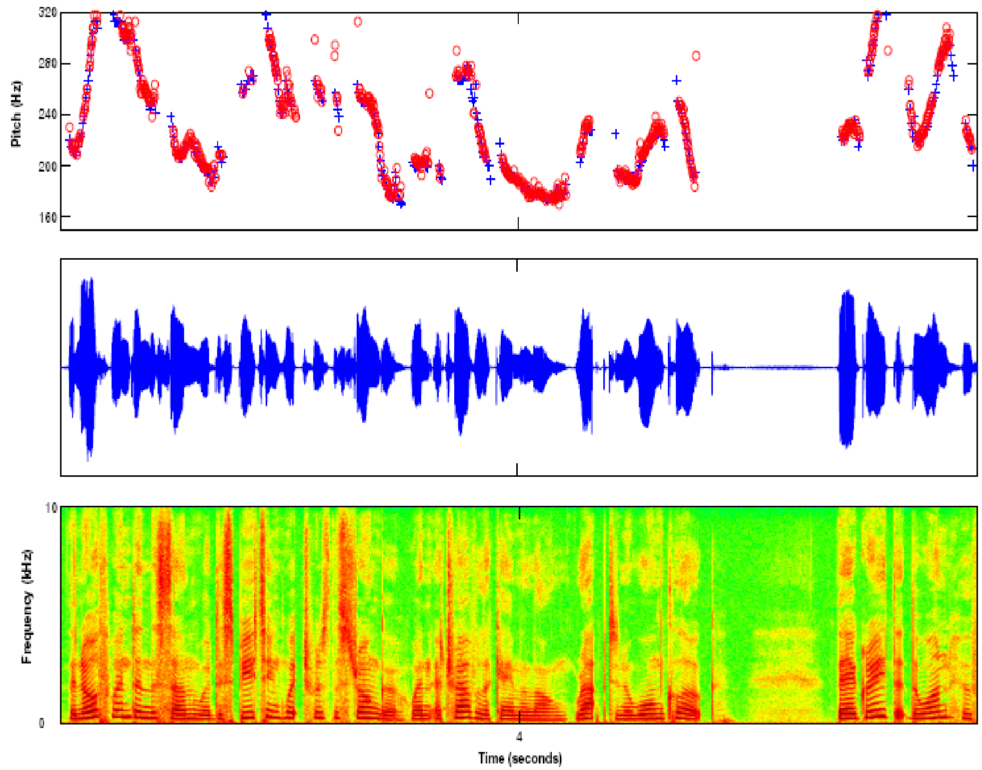
Named entity recognition

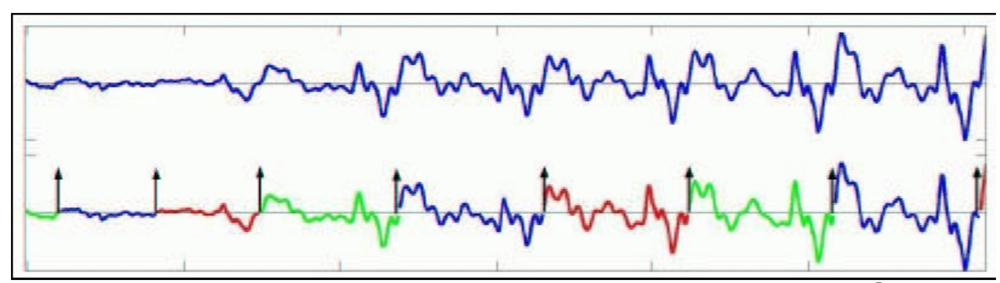
Text segmentation

Example: Speech Recognition

- Given an audio waveform, would like to robustly extract & recognize any spoken words

- Observations
  – accoustics
- Labels
  – words





S. Roweis, 2004

POS Tagging

DT NNP NN VBD VBN RP NN NNS
The Georgia branch had taken on loan commitments …

DT NN IN NN VBD NNS VBD
The average of interbank offered rates plummeted …

- Observations
  - Sentence
- Tagging
  - POS for each word

What is Part-of-Speech (POS)

- Generally speaking, Word Classes (=POS) :
  - Verb, Noun, Adjective, Adverb, Article, …
- We can also include inflection:
  - Verbs: Tense, number, …
  - Nouns: Number, proper/common, …
  - Adjectives: comparative, superlative, …
  - …
- Lots of debate within linguistics about the number, nature, and universality of these
    - We'll completely ignore this debate.

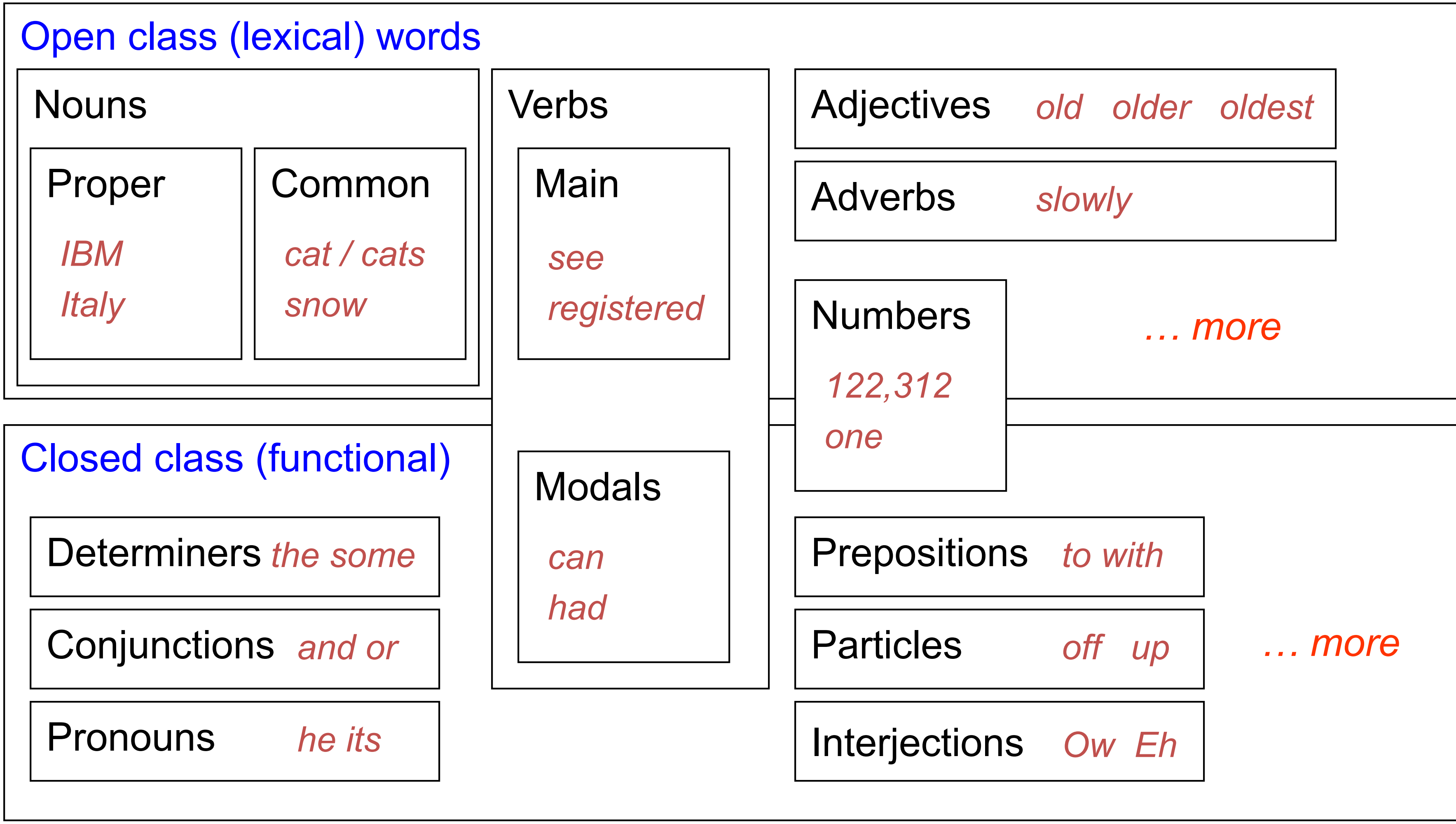
Penn TreeBank POS Tag Set

- Penn Treebank: hand-annotated corpus of *Wall Street Journal*, 1M words
- 45 tags
- Some particularities:
  - *to* /TO not disambiguated
  - Auxiliaries and verbs not distinguished

Penn Treebank Tagset

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	$
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	' or "
POS	Possessive ending	*'s*	"	Right quote	' or "
PRP	Personal pronoun	*I, you, he*	(	Left parenthesis	[, (, {, <
PRP$	Possessive pronoun	*your, one's*	)	Right parenthesis	],), }, >
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	. ! ?
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	: ; ... – -
RP	Particle	*up, off*			

Figure 5.6 Penn Treebank part-of-speech tags (including punctuation).


Open class (lexical) words
Nouns
Proper
IBM
Italy
Common
cat / cats
snow
Verbs
Main
see
registered
Adjectives old older oldest
Adverbs slowly
Numbers
122,312
one
… more
Closed class (functional)
Modals
can
had
Determiners the some
Conjunctions and or
Pronouns he its
Prepositions to with
Particles off up
Interjections Ow Eh
… more

Open vs. Closed classes

- Open vs. Closed classes
  - Closed:
    - determiners: ***a, an, the***
    - pronouns: ***she, he, I***
    - prepositions: ***on, under, over, near, by, …***
    - Usually function words (short common words which play a role in grammar)
    - Why "closed"?
  - Open:
    - Nouns, Verbs, Adjectives, Adverbs.

Open Class Words

- Nouns
  - Proper nouns (Boulder, Granby, Eli Manning)
    - English capitalizes these.
  - Common nouns (the rest).
  - Count nouns and mass nouns
    - Count: have plurals, get counted: goat/goats, one goat, two goats
    - Mass: don't get counted (snow, salt, communism) (*two snows)
- Adverbs: tend to modify verbs
  - Unfortunately, John walked home extremely slowly yesterday
  - Directional/locative adverbs (here,home, downhill)
  - Degree adverbs (extremely, very, somewhat)
  - Manner adverbs (slowly, slinkily, delicately)
- Verbs
  - In English, have morphological affixes (eat/eats/eaten)

Closed Class Words

Examples:

- prepositions: *on, under, over, …*
- particles: *up, down, on, off, …*
- determiners: *a, an, the, …*
- pronouns: *she, who, I, ..*
- conjunctions: *and, but, or, …*
- auxiliary verbs: *has, been, do, …*
- numerals: *one, two, three, third, …*
- modal verbs: *can, may, should, …*

Prepositions from CELEX

of	540,085	through	14,964	worth	1,563	pace	12
in	331,235	after	13,670	toward	1,390	nigh	9
for	142,421	between	13,275	plus	750	re	4
to	125,691	under	9,525	till	686	mid	3
with	124,965	per	6,515	amongst	525	o'er	2
on	109,129	among	5,090	via	351	but	0
at	100,169	within	5,030	amid	222	ere	0
by	77,794	towards	4,700	underneath	164	less	0
from	74,843	above	3,056	versus	113	midst	0
about	38,428	near	2,026	amidst	67	o'	0
than	20,210	off	1,695	sans	20	thru	0
over	18,071	past	1,575	circa	14	vice	0

English Particles

aboard	aside	besides	forward(s)	opposite	through
about	astray	between	home	out	throughout
above	away	beyond	in	outside	together
across	back	by	inside	over	under
ahead	before	close	instead	overhead	underneath
alongside	behind	down	near	past	up
apart	below	east, etc.	off	round	within
around	beneath	eastward(s),etc.	on	since	without

Conjunctions

and	514,946	yet	5,040	considering	174	forasmuch as	0
that	134,773	since	4,843	lest	131	however	0
but	96,889	where	3,952	albeit	104	immediately	0
or	76,563	nor	3,078	providing	96	in as far as	0
as	54,608	once	2,826	whereupon	85	in so far as	0
if	53,917	unless	2,205	seeing	63	inasmuch as	0
when	37,975	why	1,333	directly	26	insomuch as	0
because	23,626	now	1,290	ere	12	insomuch that	0
so	12,933	neither	1,120	notwithstanding	3	like	0
before	10,720	whenever	913	according as	0	neither nor	0
though	10,329	whereas	867	as if	0	now that	0
than	9,511	except	864	as long as	0	only	0
while	8,144	till	686	as though	0	provided that	0
after	7,042	provided	594	both and	0	providing that	0
whether	5,978	whilst	351	but that	0	seeing as	0
for	5,935	suppose	281	but then	0	seeing as how	0
although	5,424	cos	188	but then again	0	seeing that	0
until	5,072	supposing	185	either or	0	without	0

POS Tagging Ambiguity

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

POS Tagging

Penn Treebank POS tags

- Input: Plays well with others
- Ambiguity: NNS/VBZ UH/JJ/NN/RB IN NNS
- Output: Plays/VBZ well/RB with/IN others/NNS
- Uses:
  - Text-to-speech (how do we pronounce "lead"?)
  - Can write regexps like (Det) Adj* N+ over the output for phrases, etc.
  - An early step in NLP pipeline: output used later
  - If you know the tag, you can back off to it in other tasks

Human Upper Bound

- Deciding on the correct part of speech can be difficult even for people
- Mrs/NNP Shaefer/NNP never/RB got/VBD around/?? to/TO joining/VBG
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/?? the/DT corner/NN
- Chateau/NNP Petrus/NNP costs/VBZ around/?? 250/CD

Human Upper Bound

- Deciding on the correct part of speech can be difficult even for people
- Mrs/NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN
- Chateau/NNP Petrus/NNP costs/VBZ around/RB 250/CD

Measuring Ambiguity

		87-tag Original Brown	45-tag Treebank Brown
Unambiguous (1 tag)		**44,019**	**38,857**
Ambiguous (2–7 tags)		**5,490**	**8844**
Details:	2 tags	4,967	6,731
	3 tags	411	1621
	4 tags	91	357
	5 tags	17	90
	6 tags	2 (*well*, *beat*)	32
	7 tags	2 (*still*, *down*)	6 (*well*, *set*, *round*, *open*, *fit*, *down*)
	8 tags		4 (*'s*, *half*, *back*, *a*)
	9 tags		3 (*that*, *more*, *in*)

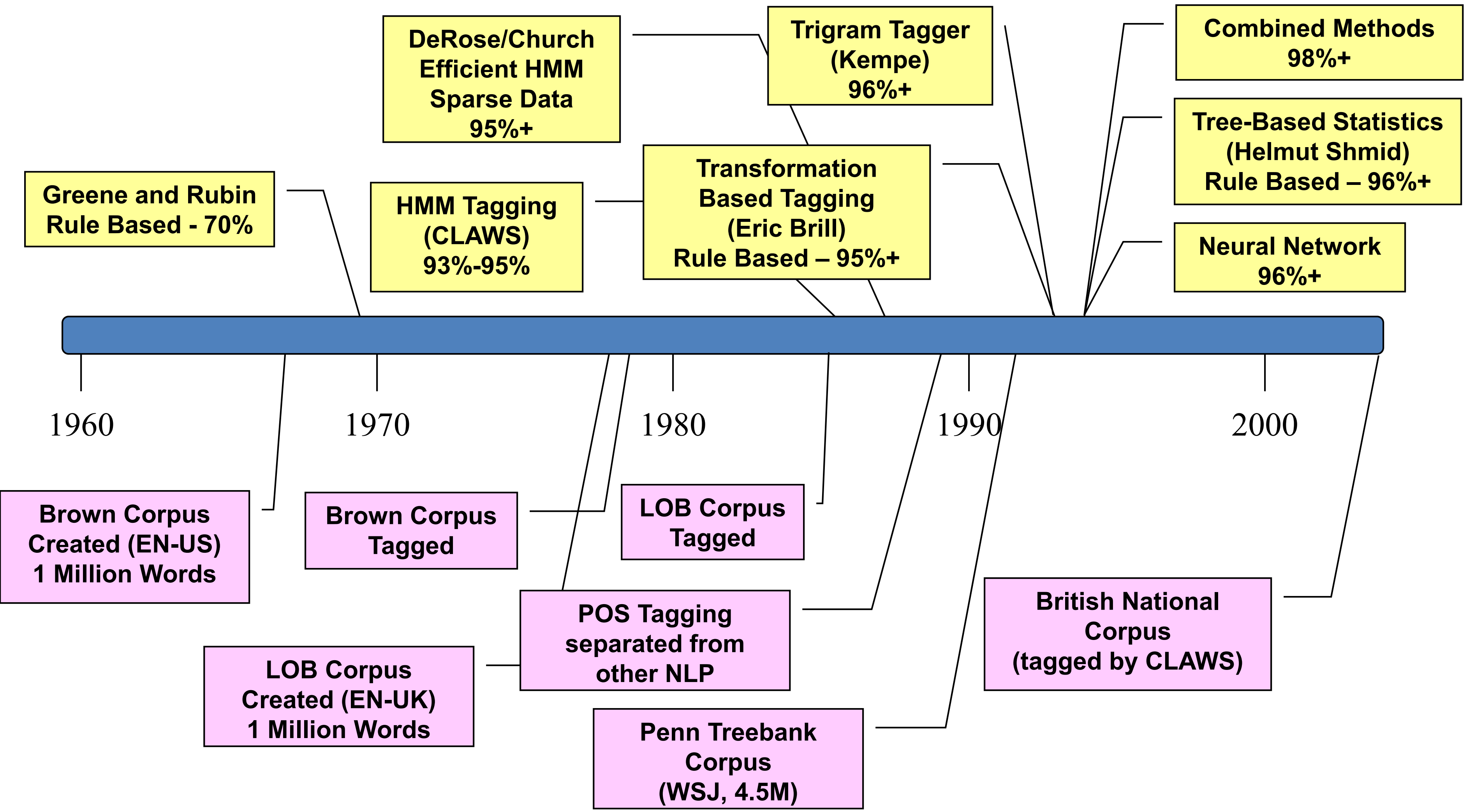
How hard is POS tagging?

- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words. E.g., *that*
  - I know *that* he is honest = IN
  - Yes, *that* play was nice = DT
  - You can’t go *that* far = RB
- 40% of the word tokens are ambiguous

POS tagging performance

- How many tags are correct? (Tag accuracy)
  - About 97% currently
  - But baseline is already 90%
    - Baseline is performance of stupidest possible method
      - Tag every word with its most frequent tag
      - Tag unknown words as nouns
  - Partly easy because
    - Many words are unambiguous
    - You get points for them (*the*, *a*, etc.) and for punctuation marks!

History of POS Tagging

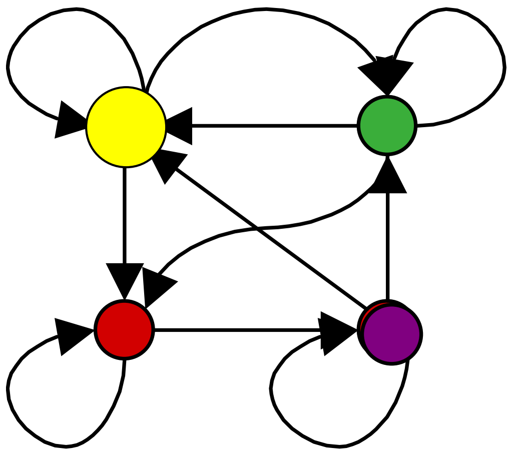


Sources of information

- What are the main sources of information for POS tagging?
  - Knowledge of neighboring words
    - Bill saw that man yesterday
    - NNP NN DT NN NN
    - VB VB(D) IN VB NN
  - Knowledge of word probabilities
    - *man* is rarely used as a verb....
- The latter proves the most useful, but the former also helps

Markov Chain

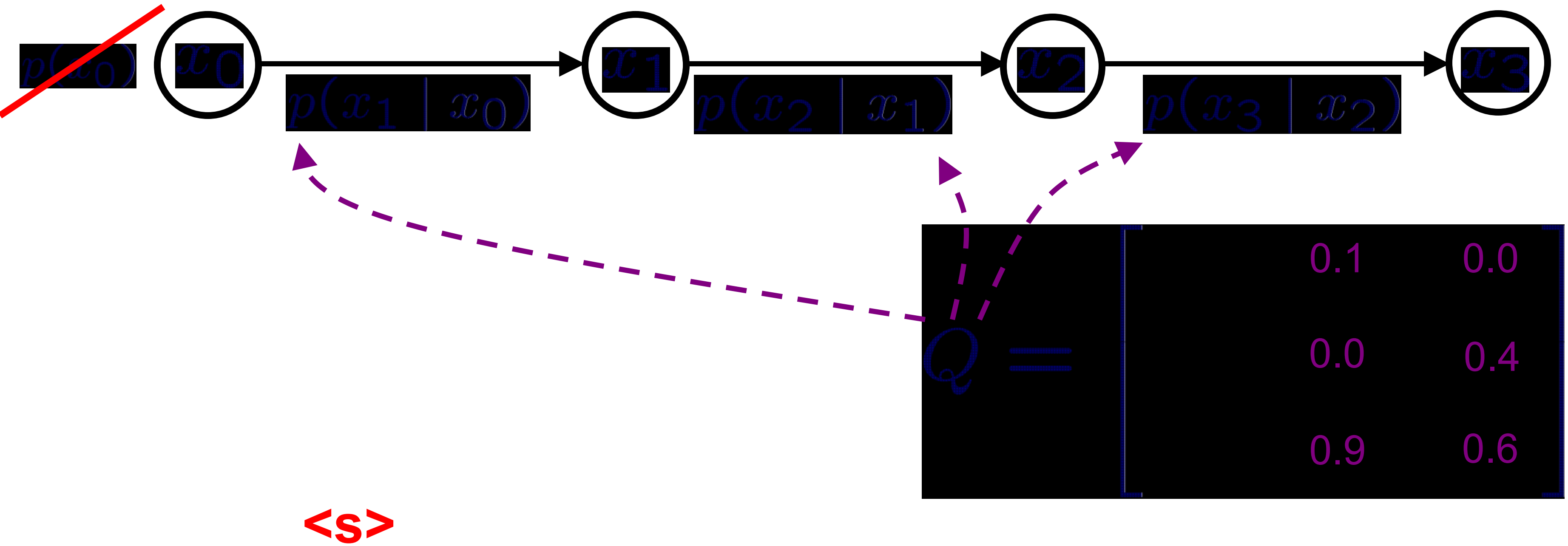
- Set of states
  - Initial probabilities
  - Transition probabilities



Markov Chain models system dynamics

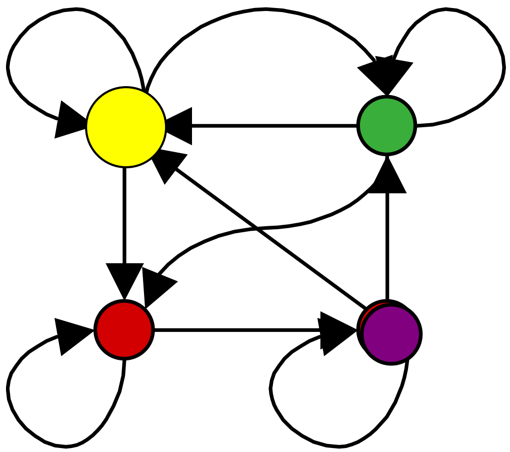
Markov Chains: Language Models

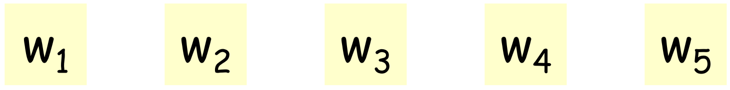
$$p(x_0, x_1, \ldots, x_T) = \cancel{p(x_0)} \prod_{t=1}^{T} p(x_t \mid x_{t-1})$$



Hidden Markov Model

- Set of states
  - ~~Initial probabilities~~
  - Transition probabilities
- Set of potential observations
  - Emission/Observation probabilities

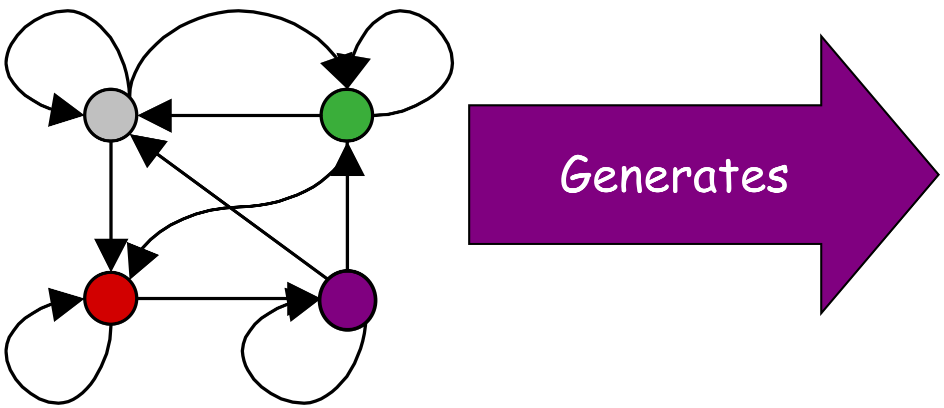




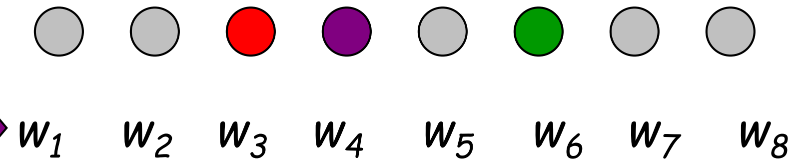
HMM generates observation sequence

Hidden Markov Models (HMMs)

Finite state machine



Hidden state sequence

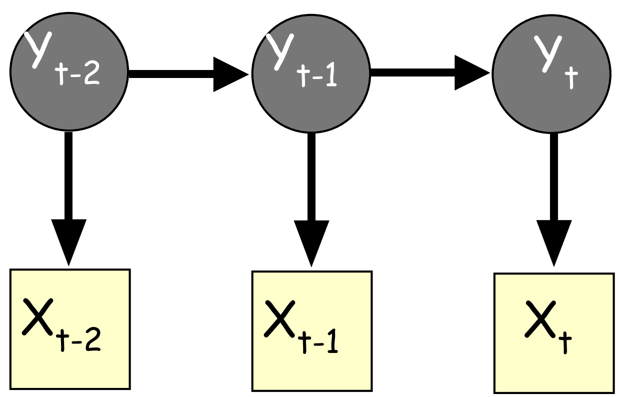


Observation sequence

Graphical Model

Hidden states . . .

Observations . . .

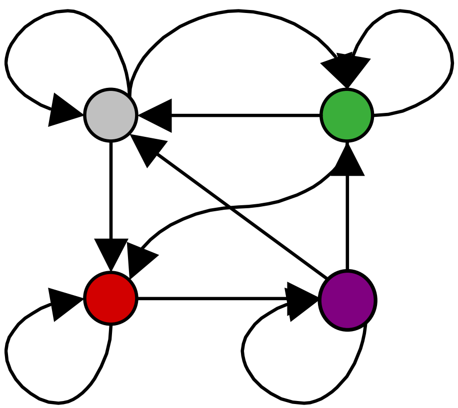


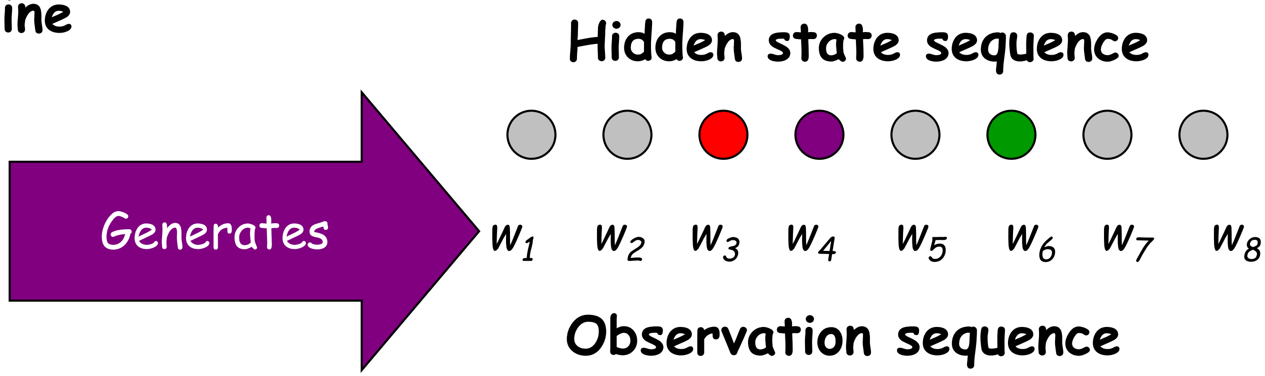
. . . Random variable Y_t takes values from $\{s_1, s_2, s_3, s_4\}$

. . . Random variable X_t takes values from $\{w_1, w_2, w_3, w_4, w_5, \ldots\}$

HMM

Finite state machine

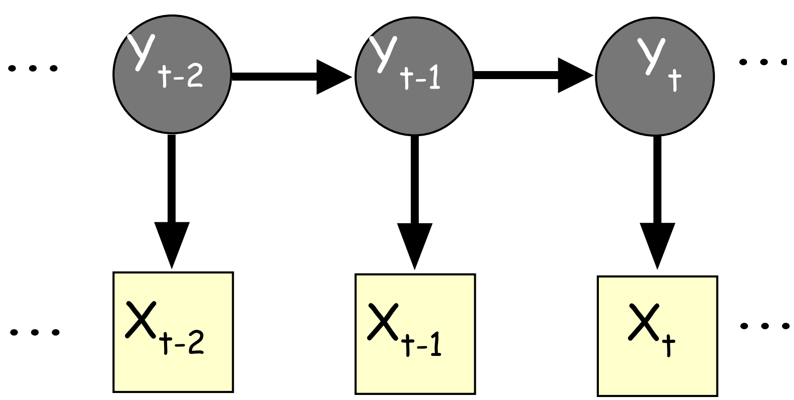




Graphical Model

Hidden states

Observations

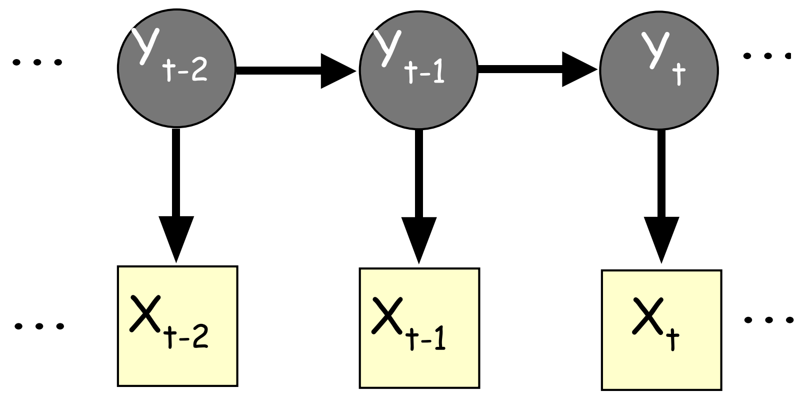


Random variable Y_t takes values from $\{s_1, s_2, s_3, s_4\}$

Random variable X_t takes values from $\{w_1, w_2, w_3, w_4, w_5, \ldots\}$

HMM
Graphical Model

Hidden states
or
Tags



Random variable Y_t takes values from $\{s_1, s_2, s_3, s_4\}$

Observations
or
Words

Random variable X_t takes values from $\{w_1, w_2, w_3, w_4, w_{5, \ldots}\}$

Need Parameters:

Start state probabilities: $P(Y_1 = s_k)$

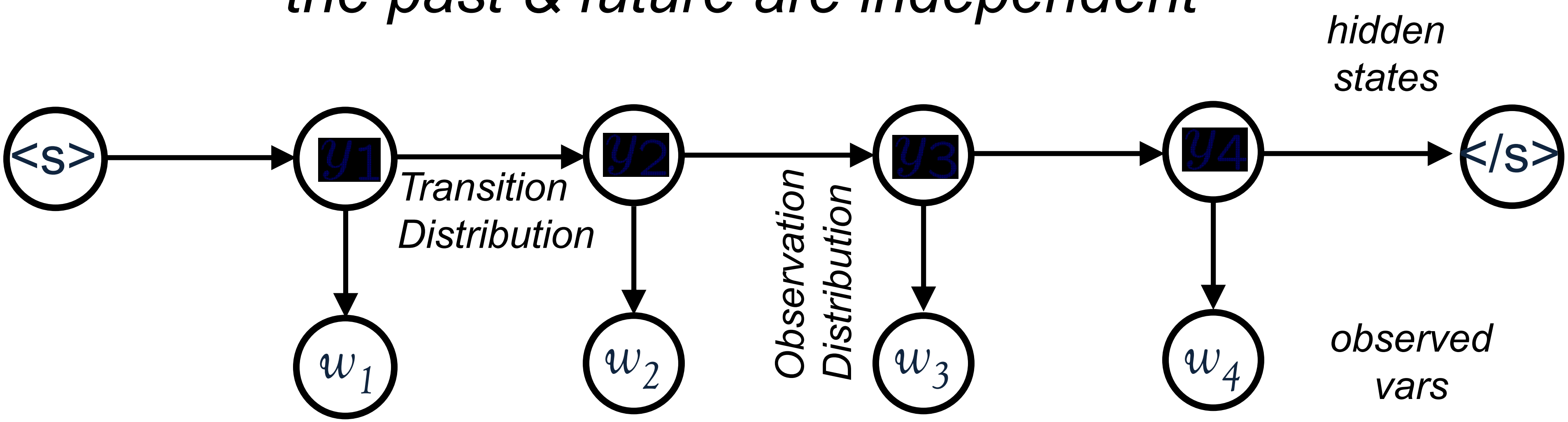
Transition probabilities: $P(Y_t = s_i \mid Y_{t-1} = s_k)$

Observation probabilities: $P(X_t = w_j \mid Y_t = s_k)$

Hidden Markov Models for Text

- Just another graphical model…

"Conditioned on the present,
the past & future are independent"



$$P(\vec{w}, \vec{y}) = \prod_{t=1}^{T+1} q(y_t \mid y_{t-1}) \prod_{t=1}^{T} e(w_t \mid y_t)$$

HMM Generative Process

- We can easily sample sequences pairs:

  $\mathbf{X_{1:n}, Y_{1:n}}$

- Sample initial state: <s>
- For i = 1 ... n
  - Sample $\mathbf{y_i}$ from the distribution $q(y_i|y_{i-1})$
  - Sample $\mathbf{x_i}$ from the distribution $e(w_i|y_i)$
- Sample </s> from $q(</s>|y_i)$

Example: POS Tagging

- Setup:
  - states $S = \{DT, NNP, NN, ... \}$ are the POS tags
  - Observations W in V are words
  - Transition dist'n $q(y_i|y_{i-1})$ models the tag sequences
  - Observation dist'n $e(w_i|y_i)$ models words given their POS

Neighboring states

Current word

Subtlety: not dependent on neighboring words directly
influence thru neighboring tags.

- Most important task: tagging
  - Decoding: find the most likely tag sequence for words w

$$\arg\max_{y1 \ldots yn} P(y_1, \ldots, y_n | w_1, \ldots, w_n)$$

Trigram HMMs

$$\cancel{P(\vec{w},\vec{y}) = \prod_{t=1}^{T} q(y_t \mid y_{t-1}) e(w_t \mid y_t)}$$

$$P(\vec{w},\vec{y}) = \prod_{t=1}^{T+1} q(y_t \mid y_{t-1}, y_{t-2}) \prod_{t=1}^{T} e(w_t \mid y_t)$$

- y_0=y_{-1}=<s>. y_{T+1}=</s>
- Parameters
  - q(s|u,v) for s ϵ S U {</s>}, u,v ϵ S U {<s>}
  - e(w|s) for w ϵ V and s ϵ S

Parameter Estimation

Counting & Smoothing

$$q(y_t \mid y_{t-1}, y_{t-2}) = \lambda_1 \frac{c(y_{t-2}, y_{t-1}, y_t)}{c(y_{t-2}, y_{t-1})} + \lambda_2 \frac{c(y_{t-1}, y_t)}{c(y_{t-1})} + \lambda_3 \frac{c(y_t)}{N}$$

$$\sum_i \lambda_i = 1$$

$$e(w_t \mid y_t) = \frac{c(w_t, y_t)}{c(y_t)}$$

Bad idea: zeros!

how to smooth a really low freq word?

Low Frequency Words

- Test sentence:
  - Astronaut Sujay M. Kulkarni decided not to leave the tricky spot, manning a tough situation by himself.
- Intuition
  - manning likely a verb. Why?
    - “-ing”
  - Sujay likely a noun. Why?
    - Capitalized in the middle of a sentence

Low Frequency Words Solution

- Split vocabulary into two sets:
  – frequent (count >k) and infrequent

- Map low frequency words into a
  – small, finite set
  – using word's orthographic features

Words → Orthographic Features

- (Bikel et al 1999) for NER task

Word Feature	Example Text	Intuition
twoDigitNum	90	Two-digit year
fourDigitNum	1990	Four digit year
containsDigitAndAlpha	A8956-67	Product code
containsDigitAndDash	09-96	Date
containsDigitAndSlash	11/9/89	Date
containsDigitAndComma	23,000.00	Monetary amount
containsDigitAndPeriod	1.00	Monetary amount, percentage
otherNum	456789	Other number
allCaps	BBN	Organization
capPeriod	M.	Person name initial
firstWord	*first word of sentence*	No useful capitalization information
initCap	Sally	Capitalized word
lowerCase	can	Uncapitalized word
other	,	Punctuation marks, all other words

- Features computed in order.

Example

- Training data
  - Astronaut/NN Sujay/NNP M./NNP Kulkarni/NNP decided/VBD not/RB to/TO leave/VB the/DT tricky/JJ spot/NN ,/, manning/VBG a/DT tough/JJ situation/NN by/IN himself/PRP .
  - firstword/NN initCap/NNP capPeriod/NNP initCap/NNP decided/VBD not/RB to/TO leave/VB the/DT tricky/JJ spot/NN ,/, endinING/VBG a/DT tough/JJ situation/NN by/IN himself/PRP .

HMM Inference

- Decoding: most likely sequence of hidden states
  - Viterbi algorithm
- Evaluation: prob. of observing an obs. sequence
  - Forward Algorithm (very similar to Viterbi)
- Marginal distribution: prob. of a particular state
  - Forward-Backward

Decoding Problem

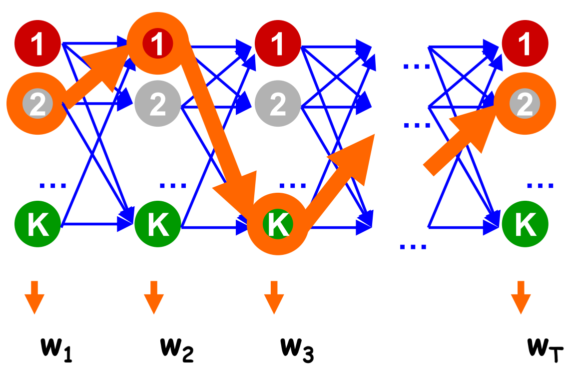
Given $w=w_1 \ldots w_T$ and HMM θ, what is "best" parse $y_1 \ldots y_T$?

Several possible meanings of 'solution'

1. States which are individually most likely
2. Single best state sequence

We want ***sequence*** $y_1 \ldots y_T$,
such that $P(y|w)$ is maximized

$$y^* = \text{argmax}_y \, P(\, y|w \,)$$



Most Likely Sequence

- Problem: find the most likely (Viterbi) sequence under the model
  - Given model parameters, we can score any sequence pair

NNP	VBZ	NN	NNS	CD	NN	.
Fed	raises	interest	rates	0.5	percent	.

$$P(\mathbf{Y}_{1:T+1}, \mathbf{w}_{1:T}) = q(\text{NNP}|<s>,<s>)\ q(\text{Fed}|\text{NNP})\ P(\text{VBZ}|<s>,\text{NNP})\ P(\text{raises}|\text{VBZ})\ P(\text{NN}|\text{NNP},\text{VBZ}) \ldots\ldots$$

  - In principle, we're done – list all possible tag sequences, score each one, pick the best one (the Viterbi state sequence)

2T+1 operations per sequence

NNP VBZ NN NNS CD NN → logP = -23

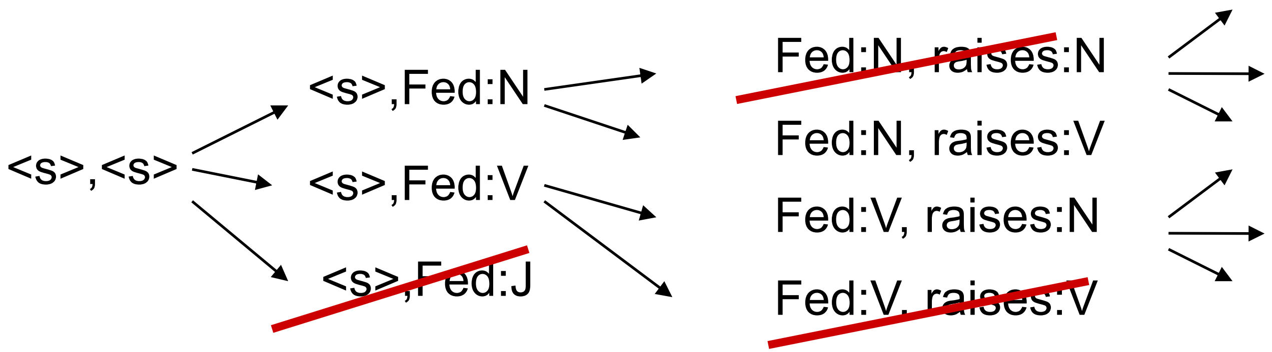
NNP NNS NN NNS CD NN → logP = -29

NNP VBZ VB NNS CD NN → logP = -27

$|Y|^T$ tag sequences!

Finding the Best Trajectory

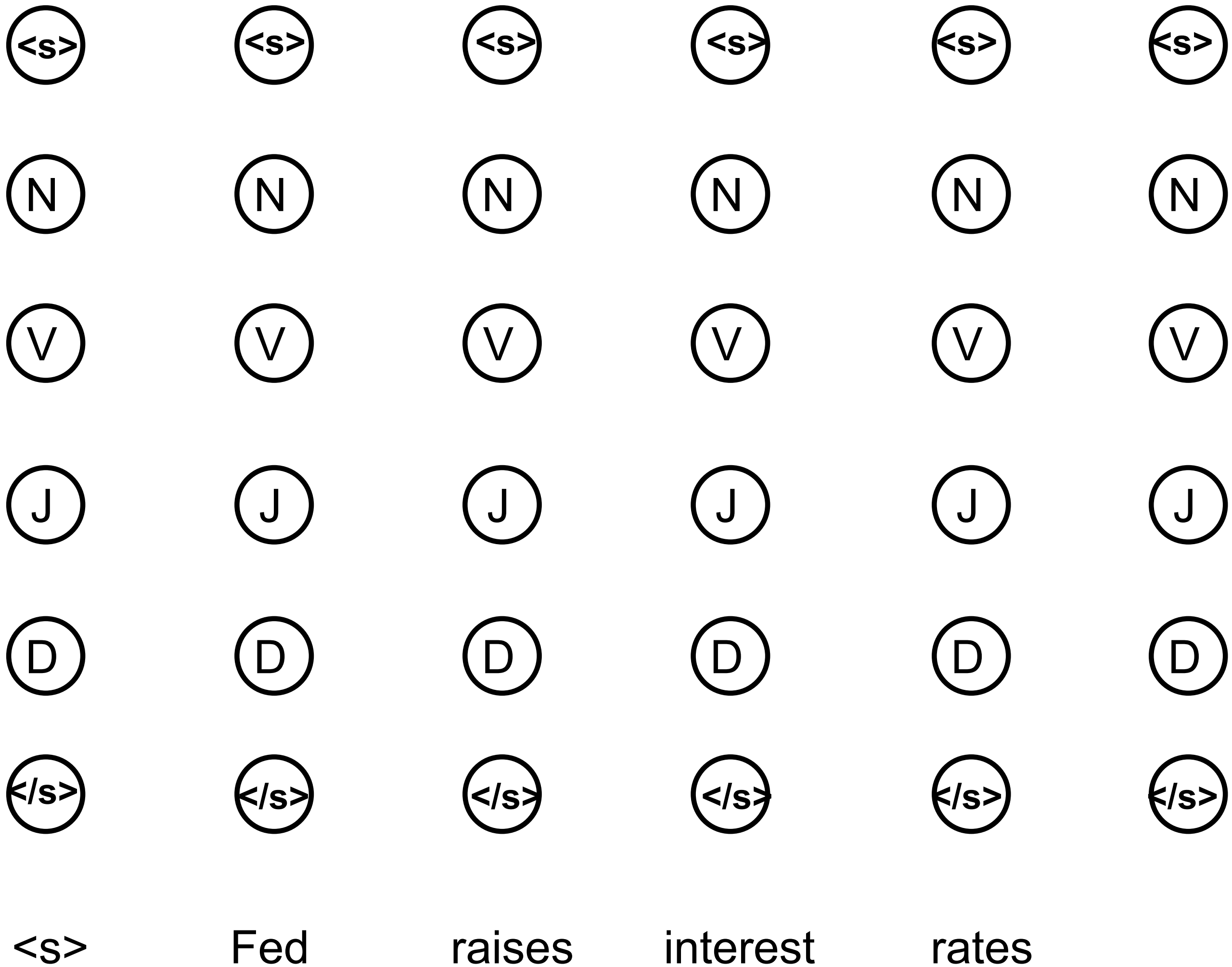
- Brute Force: Too many trajectories (state sequences) to list
- Option 1: Beam Search



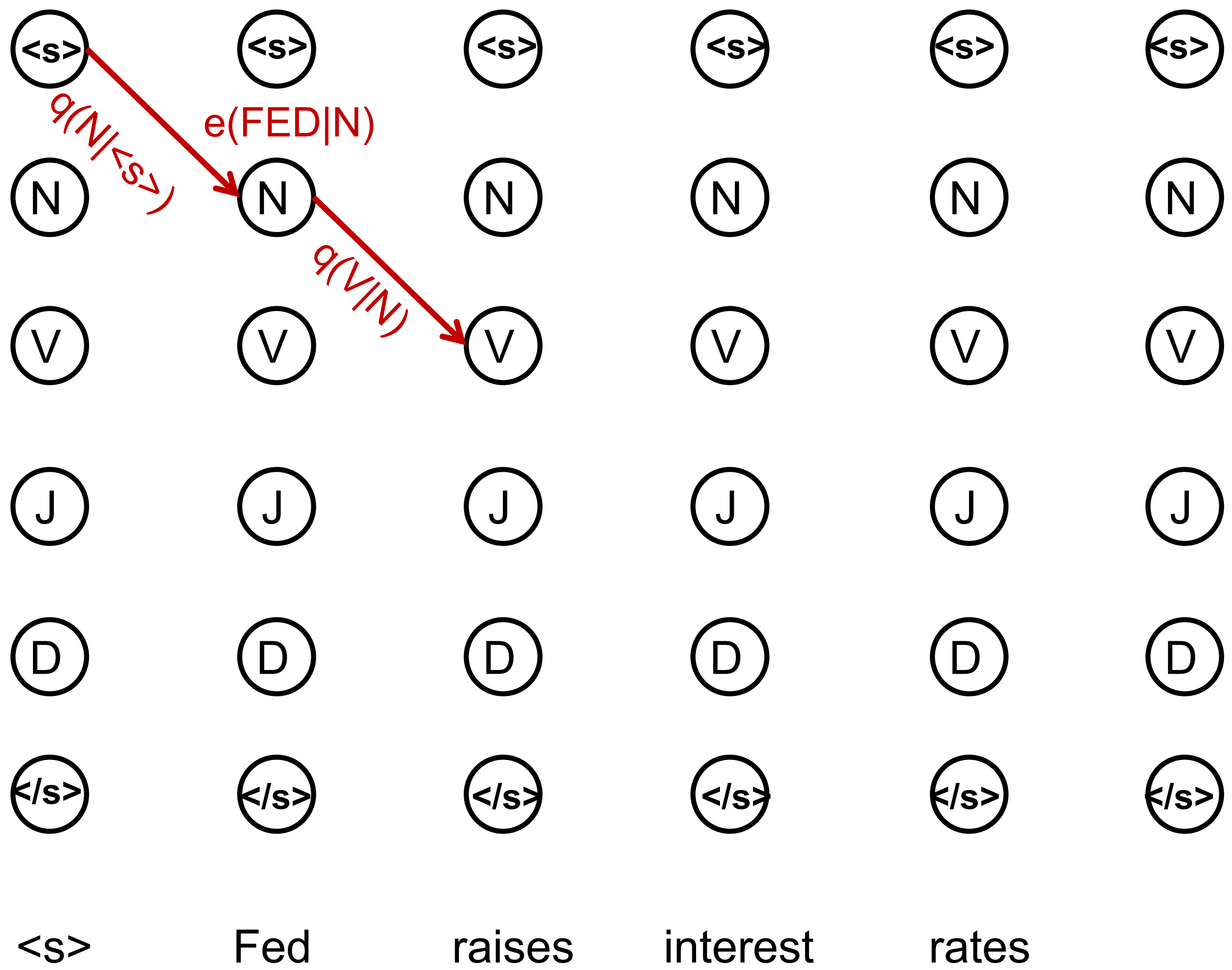
  - A beam is a set of partial hypotheses
  - Start with just the single empty trajectory
  - At each derivation step:
    - Consider all continuations of previous hypotheses
    - Discard most, keep top k

- Beam search works ok in practice
  - … but sometimes you want the optimal answer
  - … and there's often a better option than naïve beams

State Lattice / Trellis (Bigram HMM)



State Lattice / Trellis (Bigram HMM)



Dynamic Programming (Bigram)

- Decoding: $$\vec{y}^* = \arg\max_{\vec{y}} P(\vec{y} \mid \vec{w}) = \arg\max_{\vec{y}} P(\vec{w}, \vec{y})$$
$$= \arg\max_{\vec{y}} \prod_{t=1}^{T+1} q(y_t \mid y_{t-1}) \prod_{t=1}^{T} e(w_t \mid y_t)$$
- First consider how to compute max
- Define $\delta_i(y_i) = \max_{y[1:i-1]} P(y_{[1..i]}, w_{[1..i]})$
  - probability of ***most likely*** state sequence ending with tag y_i, given observations $w_1, \ldots, w_i$

$$\delta_i(y_i) = \max_{y[1:i-1]} e(w_i \mid y_i) q(y_i \mid y_{i-1}) P(y_{[1..i-1]}, w_{[1..i-1]})$$

$$= e(w_i \mid y_i) \max_{y_{i-1}} q(y_i \mid y_{i-1}) \max_{y[1:i-2]} P(y_{[1..i-1]}, w_{[1..i-1]})$$

$$= e(w_i \mid y_i) \max_{y_{i-1}} q(y_i \mid y_{i-1}) \delta_{i-1}(y_{i-1})$$

Viterbi Algorithm for Bigram HMMs

- Input: $w_1,\ldots,w_T$, model parameters q() and e()
- Initialize: $\delta_0(\langle s \rangle) = 1$
- For k=1 to T do
  - For (y') in all possible tagset

$$\delta_i(y') = e(w_i \mid y') \max_{y} q(y' \mid y)\delta_{i-1}(y)$$

- Return

$$\max_{y'} q(</s> \mid y')\delta_T(y')$$

returns only the optimal value

keep backpointers

Viterbi Algorithm for Bigram HMMs

- Input: $w_1,\ldots,w_T$, model parameters q() and e()
- Initialize: $\delta_0(\text{<s>},\text{<s>}) = 1$
- For k=1 to T do
  - For (y') in all possible tagset

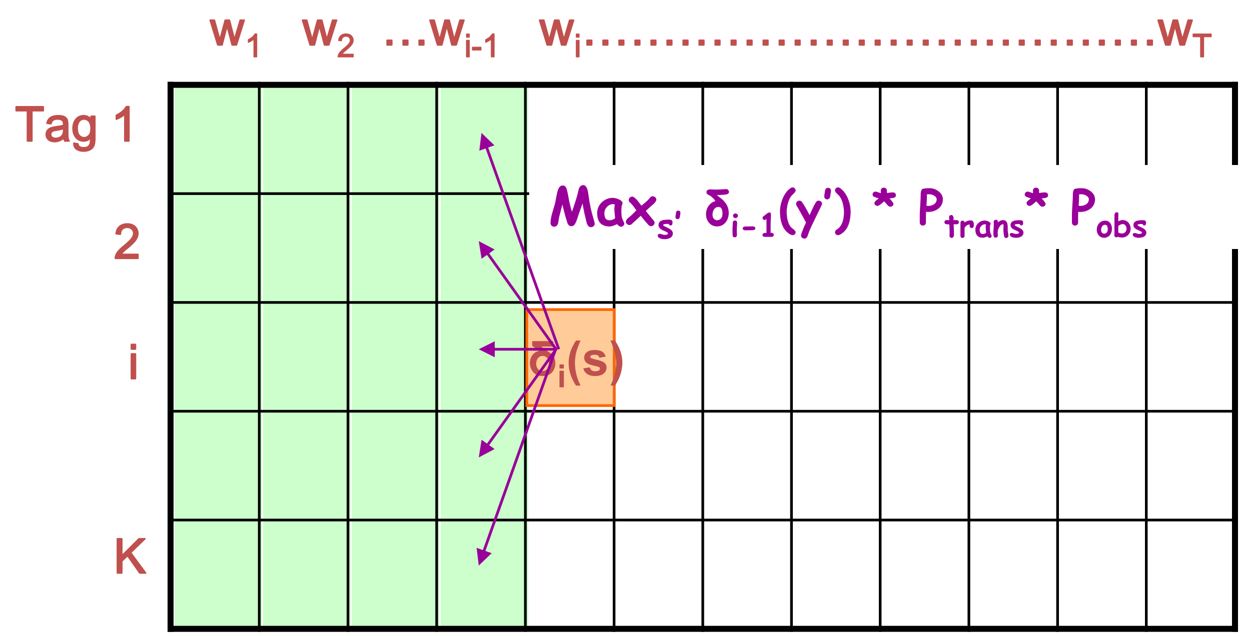
$$\delta_i(y') = e(w_i \mid y') \max_{y} q(y' \mid y)\delta_{i-1}(y)$$

$$bp_i(y') = e(w_i \mid y') \arg\max_{y} q(y' \mid y)\delta_{i-1}(y)$$

- Set $y_T = \arg\max_{y'} q(</s> \mid y')\delta_T(y')$
- For k=T-1 to 1 do
  - Set $y_k = bp_k(y_{k+1})$
- Return y[1..T]

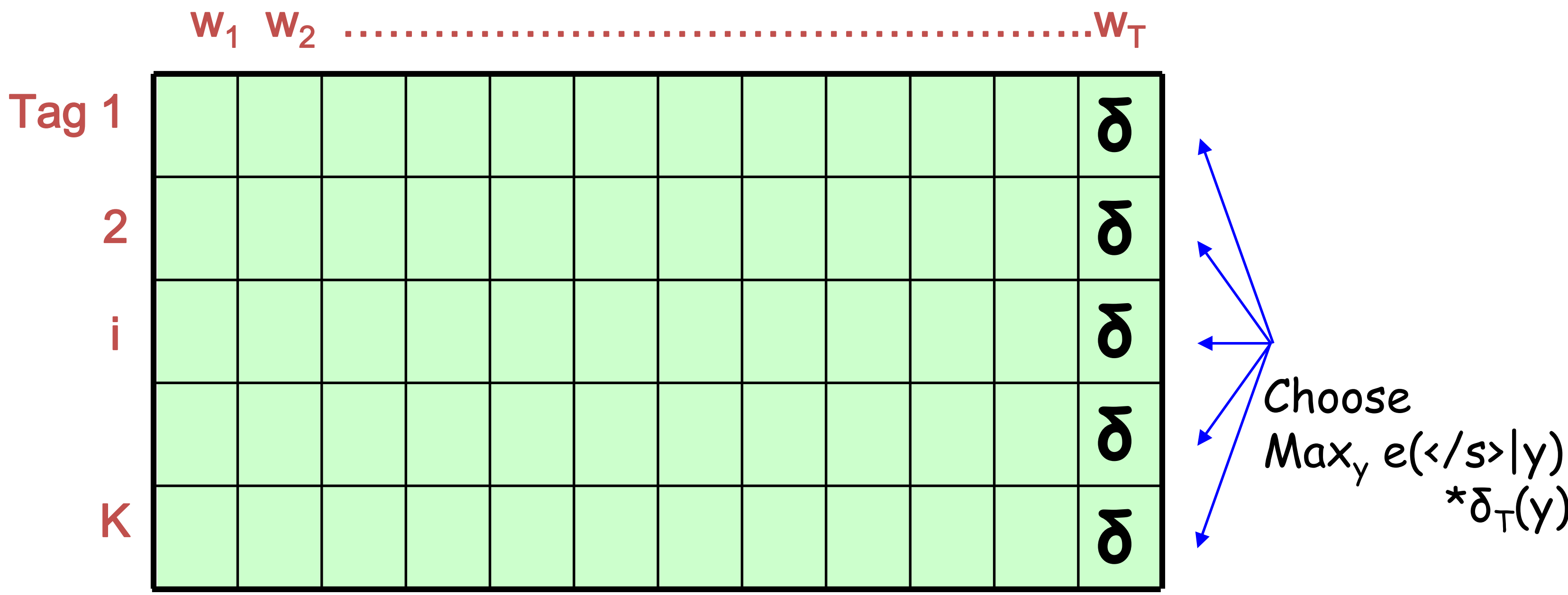
Time: $O(|Y|^2T)$
Space: $O(|Y|T)$

Viterbi Algorithm for Bigram HMMs

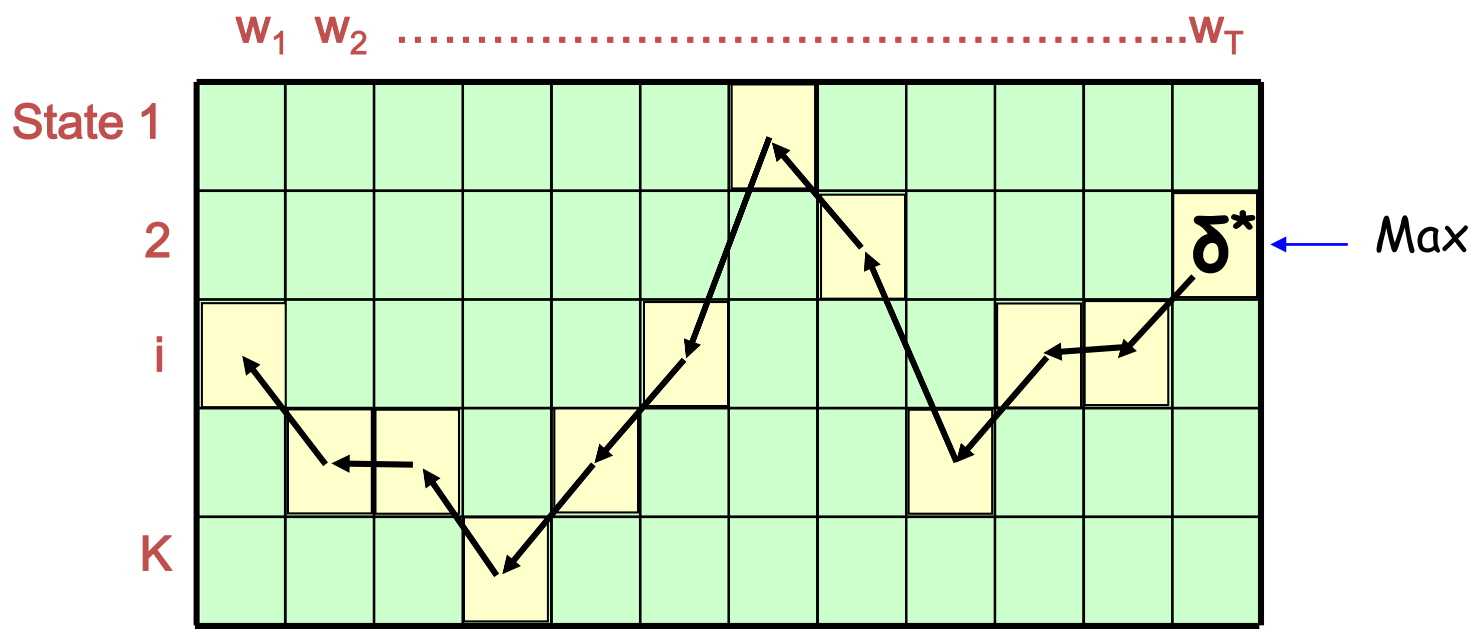


Remember: $\delta_i(y)$ = probability of most likely tag seq ending with y at time i

Terminating Viterbi



Terminating Viterbi



How did we compute δ^*?

$\text{Max}_{s'}\ \delta_{T-1}(y') * P_{trans} * P_{obs}$

Now Backchain to Find Final Sequence

Time: $O(|Y|^2 T)$

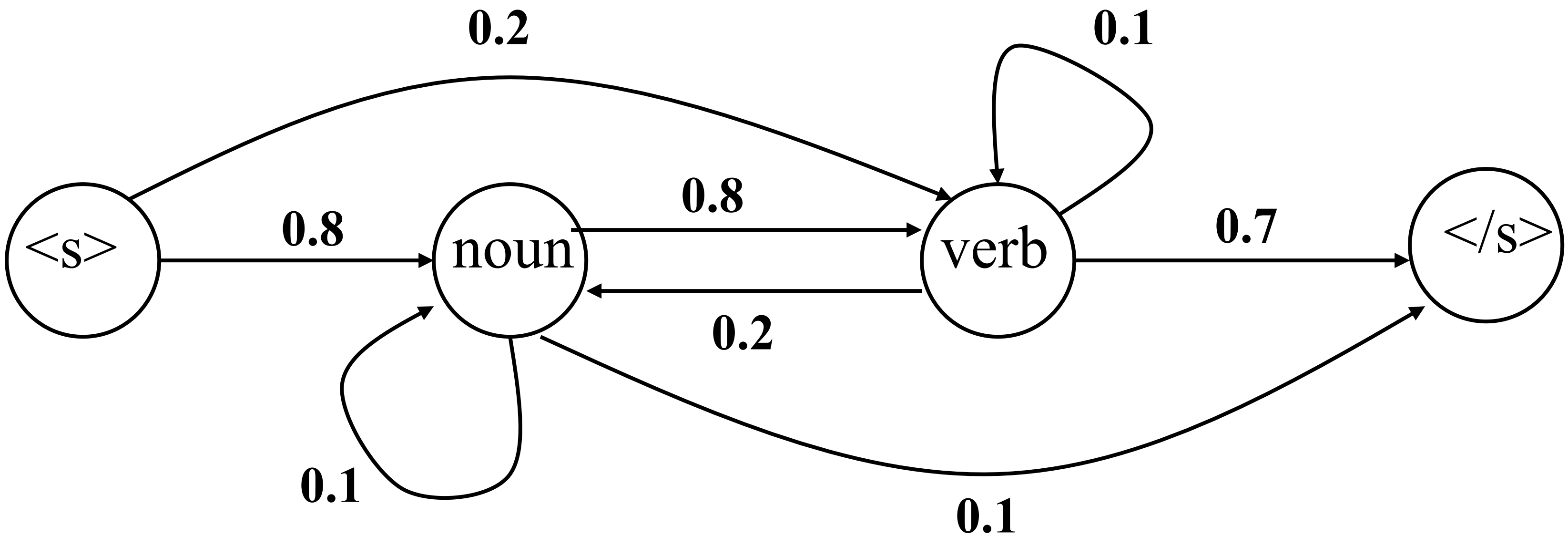
Space: $O(|Y|T)$

Linear in length of sequence

Example

Fish sleep.

Example: Bigram HMM

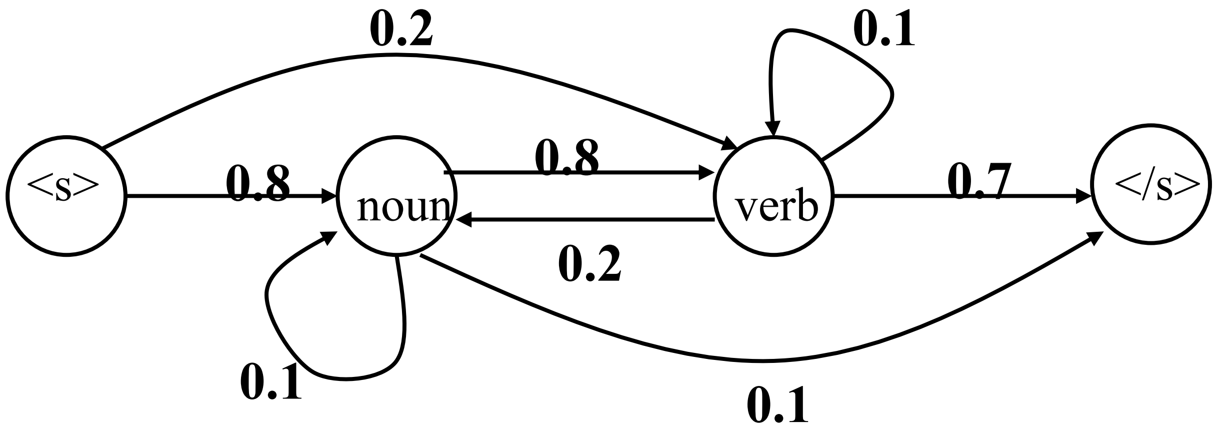


Data

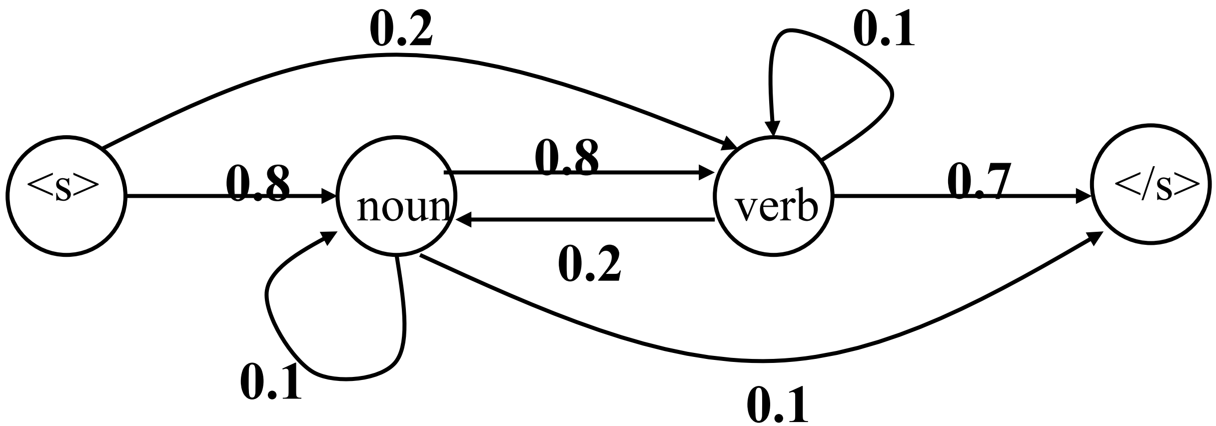
- A two-word language: “fish” and “sleep”
- Suppose in our training corpus,
  - “fish” appears 8 times as a noun and 5 times as a verb
  - “sleep” appears twice as a noun and 5 times as a verb
- Emission probabilities:
  - Noun
    - P(fish | noun) : 0.8
    - P(sleep | noun) : 0.2
  - Verb
    - P(fish | verb) : 0.5
    - P(sleep | verb) : 0.5

Viterbi Probabilities

	0	1	2	3
start				
verb				
noun				
end				

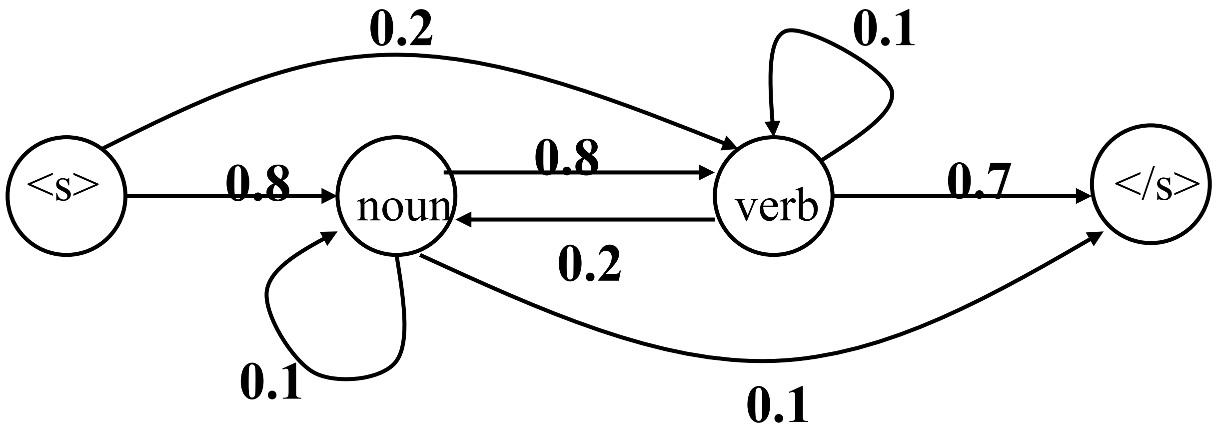


	0	1	2	3
start	1			
verb	0			
noun	0			
end	0			



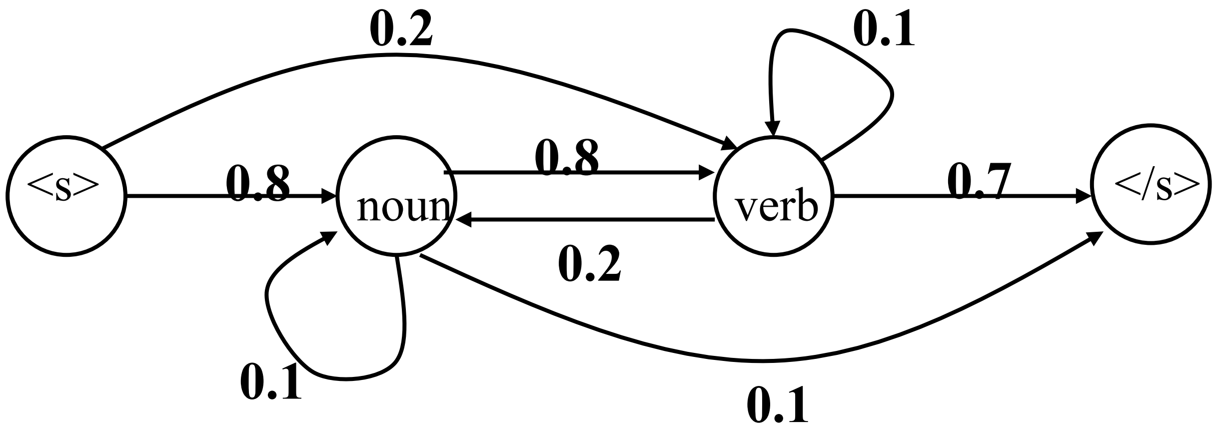
Token 1: fish

	0	1	2	3
start	1	0		
verb	0	.2 * .5		
noun	0	.8 * .8		
end	0	0		



Token 1: fish

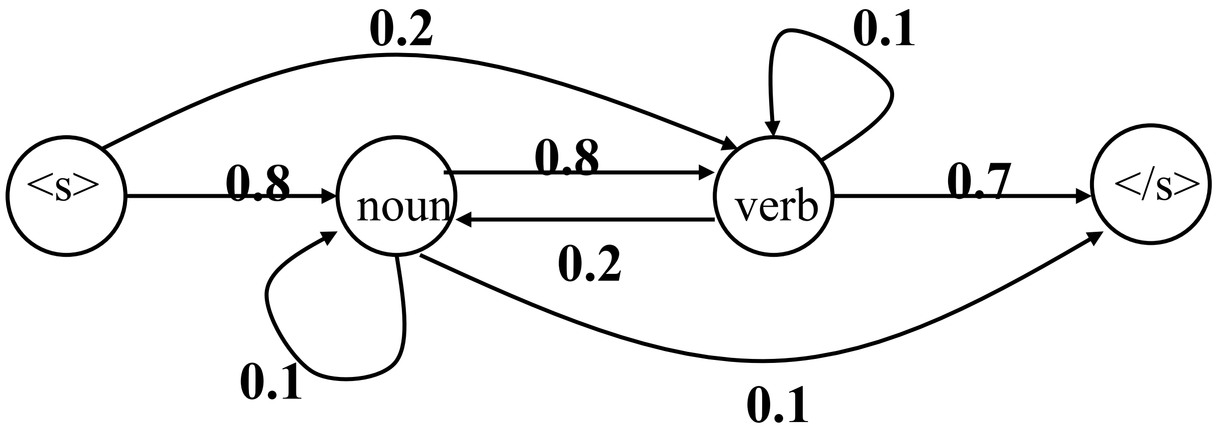
	0	1	2	3
start	1	0		
verb	0	.1		
noun	0	.64		
end	0	0		



Token 2: sleep

(if ‘fish’ is verb)

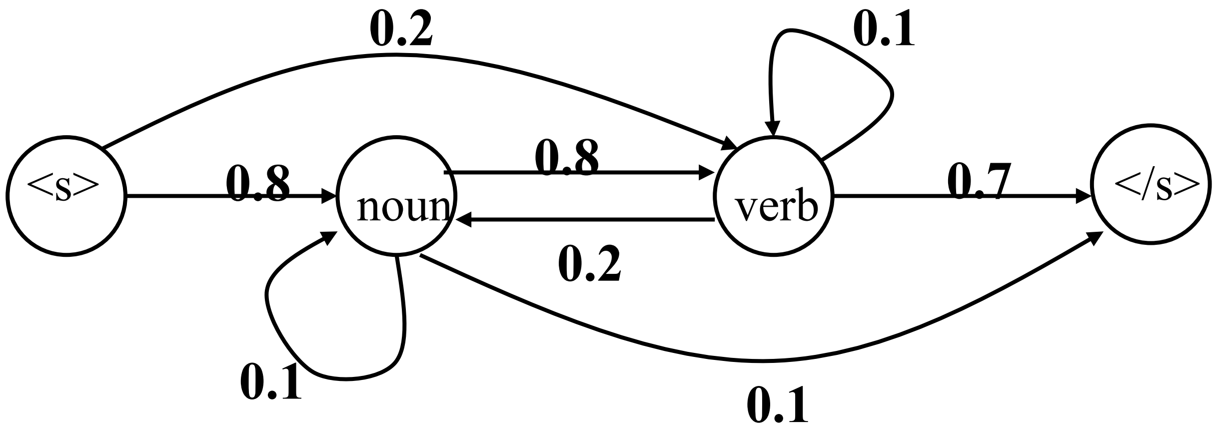
	0	1	2	3
start	1	0	0	
verb	0	.1	.1*.1*.5	
noun	0	.64	.1*.2*.2	
end	0	0	-	



Token 2: sleep

(if 'fish' is verb)

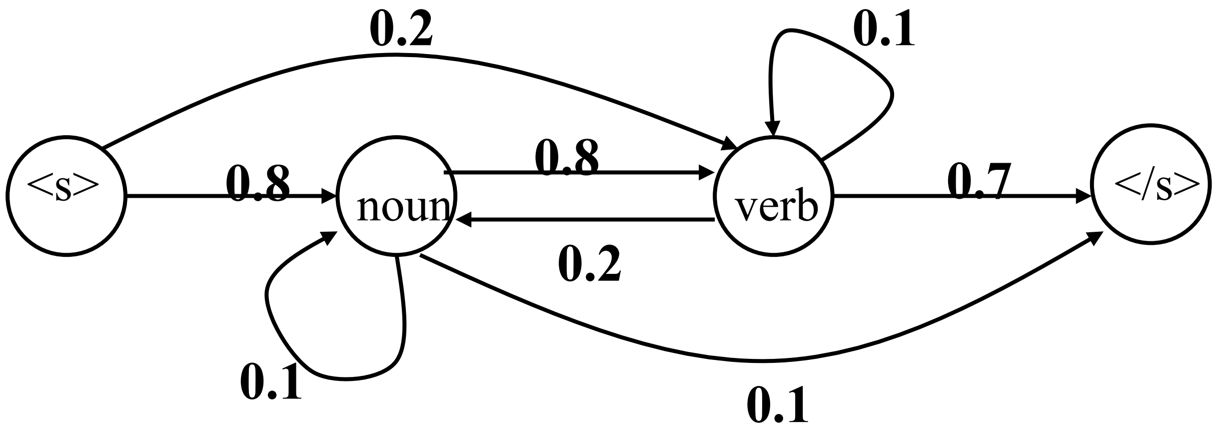
	0	1	2	3
start	1	0	0	
verb	0	.1	.005	
noun	0	.64	.004	
end	0	0	-	



Token 2: sleep

(if 'fish' is a noun)

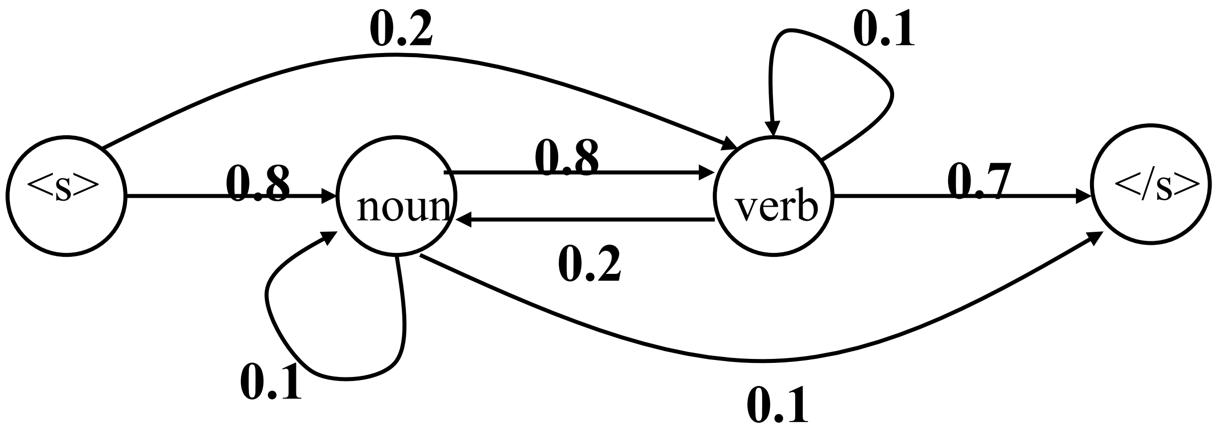
	0	1	2	3
start	1	0	0	
verb	0	.1	.005 .64*.8*.5	
noun	0	.64	.004 .64*.1*.2	
end	0	0	-	



Token 2: sleep

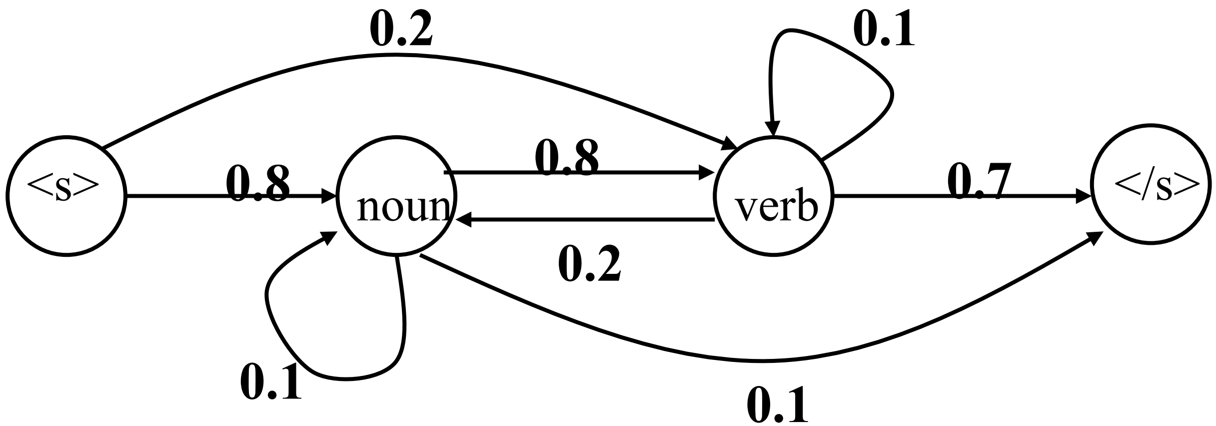
(if ‘fish’ is a noun)

	0	1	2	3
start	1	0	0	
verb	0	.1	.005 .256	
noun	0	.64	.004 .0128	
end	0	0	-	



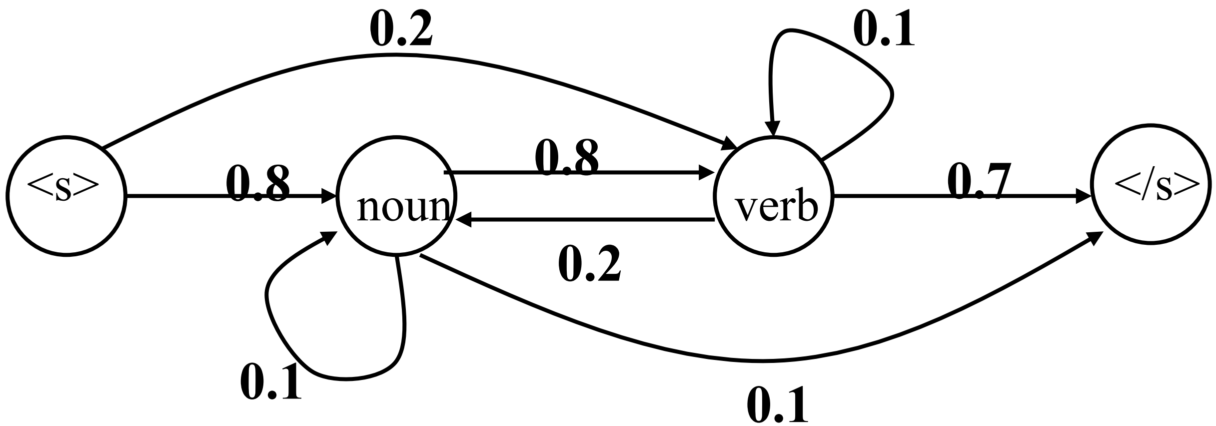
Token 2: sleep
take maximum,
set back pointers

	0	1	2	3
start	1	0	0	
verb	0	.1	~~.005~~ .256	
noun	0	.64	~~.004~~ .0128	
end	0	0	-	



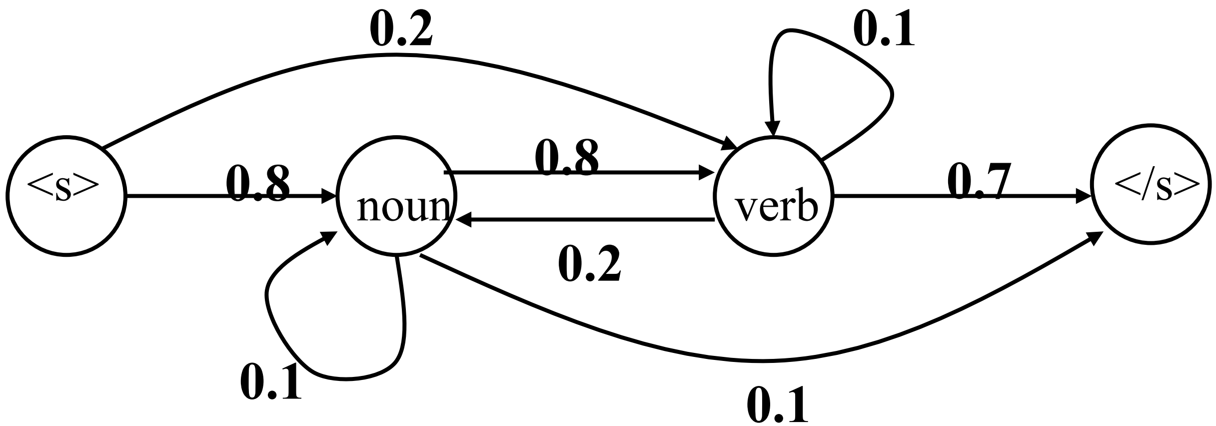
Token 2: sleep
take maximum,
set back pointers

	0	1	2	3
start	1	0	0	
verb	0	.1	.256	
noun	0	.64	.0128	
end	0	0	-	



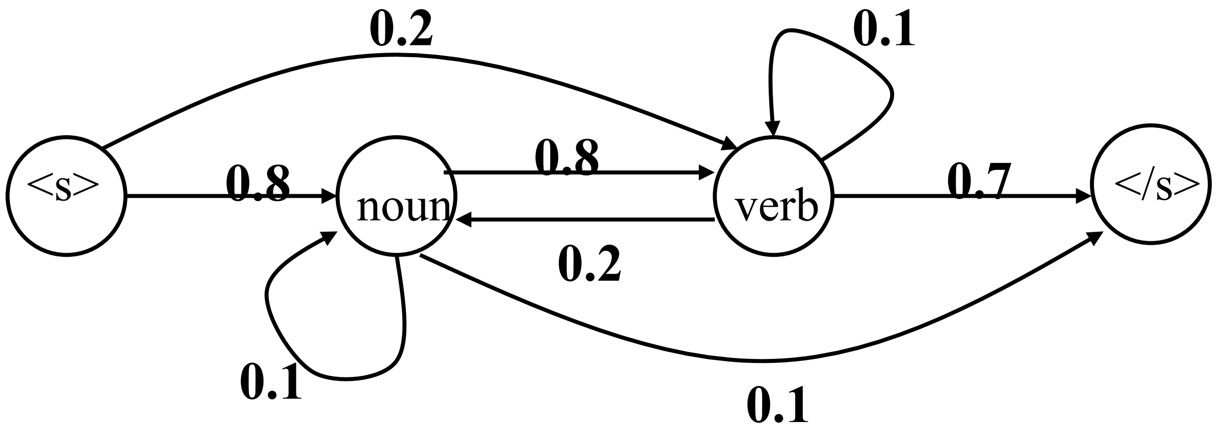
Token 3: end

	0	1	2	3
start	1	0	0	0
verb	0	.1	.256	-
noun	0	.64	.0128	-
end	0	0	-	.256*.7 .0128*.1



Token 3: end
take maximum,
set back pointers

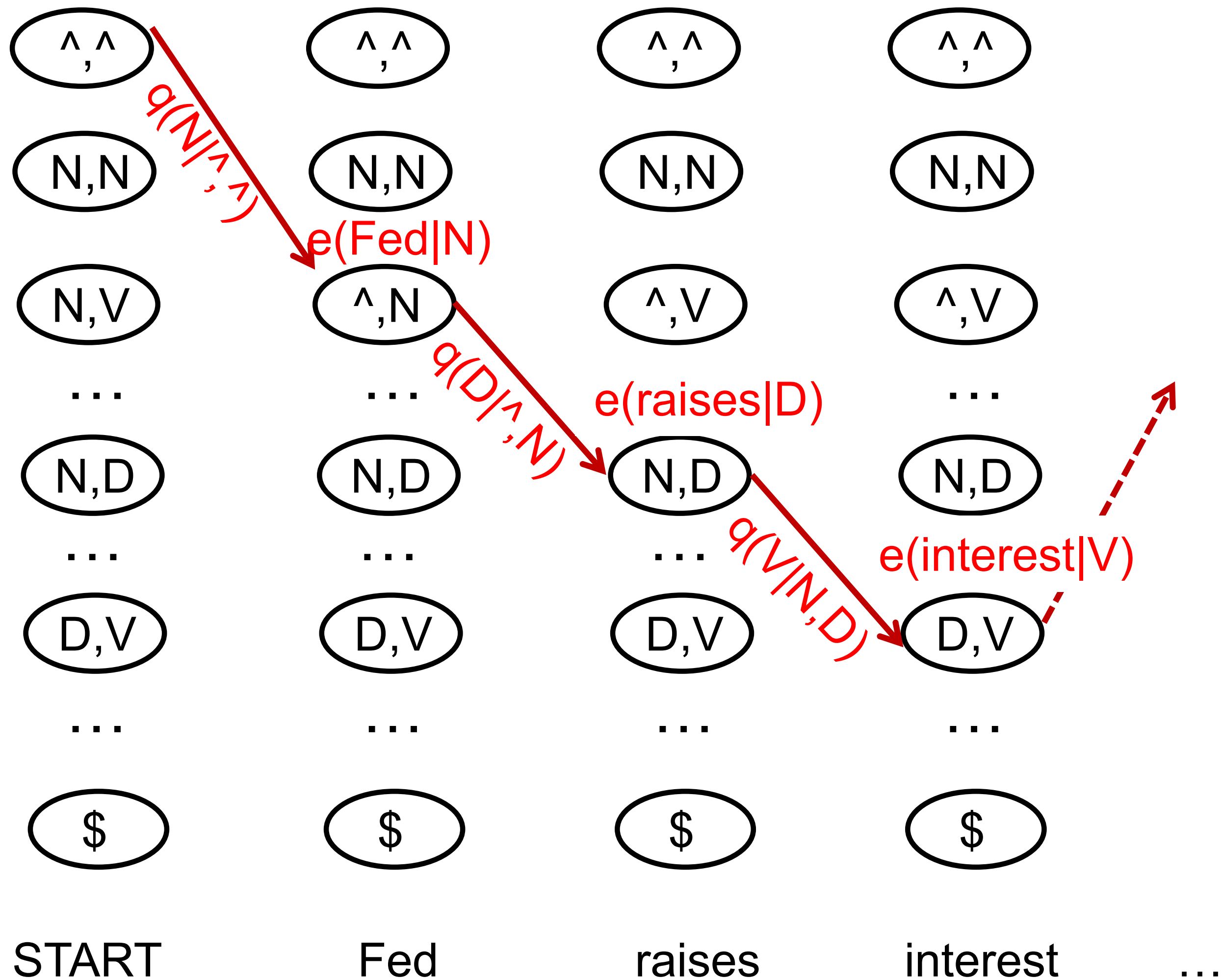
	0	1	2	3
start	1	0	0	0
verb	0	.1	.256	-
noun	0	.64	.0128	-
end	0	0	-	.256*.7 ~~.0128*.1~~



Decode:
fish = noun
sleep = verb

	0	1	2	3
start	1	0	0	0
verb	0	.1	.256	-
noun	0	.64	.0128	-
end	0	0	-	.256*.7

State Lattice / Trellis (Trigram HMM)



Dynamic Programming (Trigram)

- Decoding: $$\vec{y}^* = \arg\max_{\vec{y}} P(\vec{y} \mid \vec{w}) = \arg\max_{\vec{y}} P(\vec{w}, \vec{y})$$ $$= \arg\max_{\vec{y}} \prod_{t=1}^{T+1} q(y_t \mid y_{t-1}, y_{t-2}) \prod_{t=1}^{T} e(w_t \mid y_t)$$
- First consider how to compute max
- Define $\delta_i(y_{i-1}, y_i) = \max_{y[1:i-2]} P(y_{[1..i]}, w_{[1..i]})$
  - probability of ***most likely*** state sequence ending with tags y_{i-1}, y_i, given observations $w_1, \ldots, w_i$

$$\delta_i(y_{i-1}, y_i) = \max_{y[1:i-2]} e(w_i \mid y_i) q(y_i \mid y_{i-2}, y_{i-1}) P(y_{[1..i-1]}, w_{[1..i-1]})$$

$$= e(w_i \mid y_i) \max_{y_{i-2}} q(y_i \mid y_{i-2}, y_{i-1}) \max_{y[1:i-3]} P(y_{[1..i-1]}, w_{[1..i-1]})$$

$$= e(w_i \mid y_i) \max_{y_{i-2}} q(y_i \mid y_{i-2}, y_{i-1}) \delta_{i-1}(y_{i-2}, y_{i-1})$$

Viterbi Algorithm for Trigram HMMs

- Input: $w_1,\ldots,w_T$, model parameters q() and e()
- Initialize: $\delta_0(<s>,<s>) = 1$
- For k=1 to T do
  - For (y',y'') in all possible tagset

$$\delta_i(y',y'') = e(w_i \mid y'')\max_{y} q(y'' \mid y,y')\delta_{i-1}(y,y')$$

- Return

$$\max_{y',y''} q(</s> \mid y',y'')\delta_T(y',y'')$$

returns only the optimal value

keep backpointers

Viterbi Algorithm for Trigram HMMs

- Input: $w_1,\ldots,w_T$, model parameters q() and e()
- Initialize: $\delta_0(\text{<s>},\text{<s>}) = 1$
- For k=1 to T do
  - For (y',y'') in all possible tagset

$$\delta_i(y',y'') = e(w_i \mid y'') \max_{y} q(y'' \mid y, y')\delta_{i-1}(y,y')$$

$$bp_i(y',y'') = e(w_i \mid y'') \arg\max_{y} q(y'' \mid y, y')\delta_{i-1}(y,y')$$

- Set $y_{T-1}, y_T = \arg\max_{y',y''} q(</s> \mid y', y'')\delta_T(y',y'')$
- For k=T-2 to 1 do
  - Set $y_k = bp_k(y_{k+1}, y_{k+2})$
- Return y[1..T]

Time: $O(|Y|^3 T)$
Space: $O(|Y|^2 T)$

Overview: Accuracies

- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%

Most errors on unknown words

- TnT (Brants, 2000):
  - A carefully smoothed trigram tagger
  - Suffix trees for emissions
  - 96.7% on WSJ text
  - Upper bound: ~98%

Common Errors

- Common errors [from Toutanova & Manning 00]

	JJ	NN	NNP	NNPS	RB	RP	IN	VB	VBD	VBN	VBP	Total
JJ	0	**177**	**56**	0	**61**	2	5	10	15	**108**	0	488
NN	**244**	0	**103**	0	12	1	1	29	5	6	19	525
NNP	**107**	**106**	0	**132**	5	0	7	5	1	2	0	427
NNPS	1	0	**110**	0	0	0	0	0	0	0	0	142
RB	**72**	21	7	0	0	16	**138**	1	0	0	0	295
RP	0	0	0	0	**39**	0	**65**	0	0	0	0	104
IN	11	0	1	0	**169**	**103**	0	1	0	0	0	323
VB	17	**64**	9	0	2	0	1	0	4	7	**85**	189
VBD	10	5	3	0	0	0	0	3	0	**143**	2	166
VBN	**101**	3	3	0	0	0	0	3	**108**	0	1	221
VBP	5	34	3	1	1	0	2	**49**	6	3	0	104
Total	626	536	348	144	317	122	279	102	140	269	108	3651

NN/JJ NN
official knowledge

VBD RP/IN DT NN
made up the story

RB VBD/VBN NNS
recently sold shares

Issues with HMMs for POS Tagging

- Slow for long sentences
- Only one feature for less frequent words
- No features for frequent words

- Why not try a feature rich classifier?
  - MaxEnt?

Feature-based tagger

- Can do surprisingly well just looking at a word by itself:
  - Word   the: the $\rightarrow$ DT
  - Lowercased word   Importantly: importantly $\rightarrow$ RB
  - Prefixes   unfathomable: un- $\rightarrow$ JJ
  - Suffixes   Importantly: -ly $\rightarrow$ RB
  - Capitalization   Meridian: CAP $\rightarrow$ NNP
  - Word shapes   35-year: d-x $\rightarrow$ JJ

- Then build a maxent (or whatever) model to predict tag
  - Maxent P(y|w): 93.7% overall / 82.6% unknown

Overview: Accuracies

- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(t|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - Upper bound: ~98%

How to improve supervised results?

- Build better features!

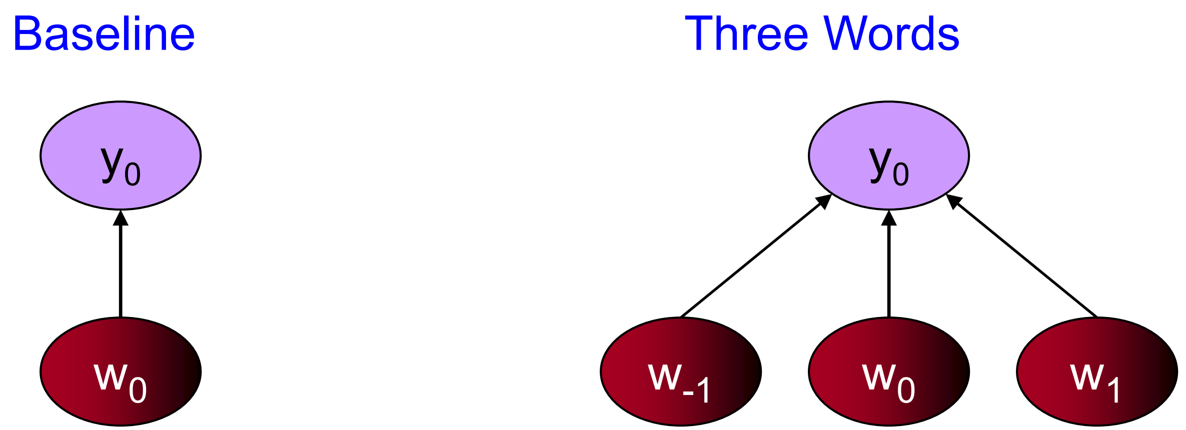
RB
PRP VBD **IN** RB IN PRP VBD .
They left as soon as he arrived .

  - We could fix this with a feature that looked at the next word

JJ
NNP NNS VBD VBN .
Intrinsic flaws remained undetected .

  - We could fix this by linking capitalized words to their lowercase versions

Tagging Without Sequence Information



Model	Features	Token	Unknown
Baseline	56,805	**93.69%**	82.61%
3Words	239,767	**96.57%**	86.78%

Using words only in a straight classifier works as well as a basic sequence model!!

Overview: Accuracies

- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(y|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - Maxent (local nbrs): 96.8% / 86.8%
  - Upper bound: ~98%

Discriminative Sequence Taggers

- Maxent $P(y|w)$ is too local
  - completely ignores sequence labeling problem
  - and predicts independently

- Discriminative Sequence Taggers
  - Feature rich
  - neighboring labels can guide tagging process
  - Example: Max Entropy Markov Models (MEMM), Linear Perceptron

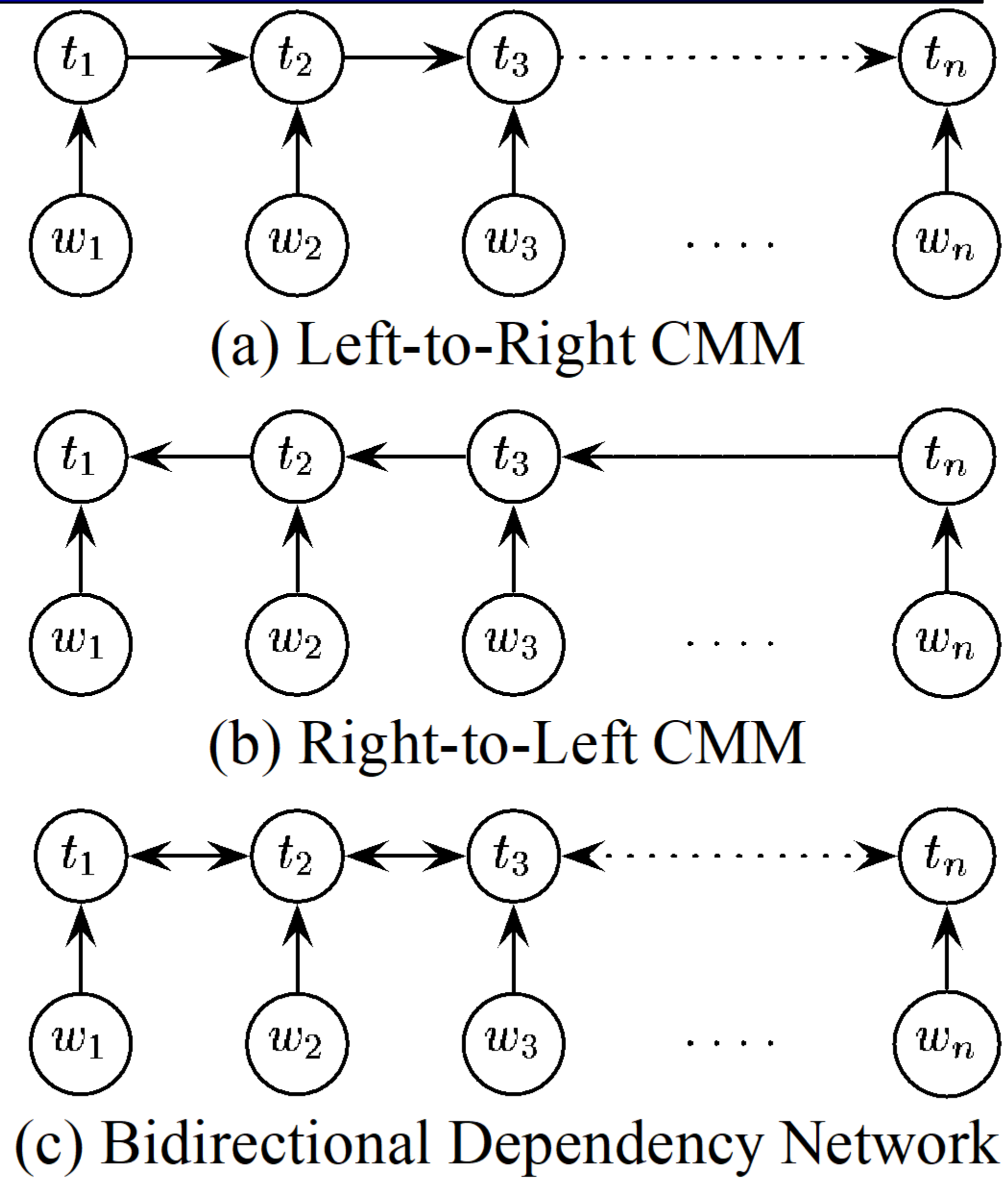
Overview: Accuracies

- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(y|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - Maxent (local nbrs): 96.8% / 86.8%
  - MEMMs: 96.9% / 86.9%
  - Linear Perceptron: 96.7% / ??

  - Upper bound: ~98%

Cyclic Network [Toutanova et al 03]

- Train two MEMMs, multiple together to score
- And be very careful
  - Tune regularization
  - Try lots of different features
  - See paper for full detail



Overview: Accuracies

- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(y|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - Maxent (local nbrs): 96.8% / 86.8%
  - MEMMs: 96.9% / 86.9%
  - Linear Perceptron: 96.7% / ??
  - Cyclic tagger: 97.2% / 89.0%

  - Upper bound: ~98%

Overview: Accuracies

- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(y|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - Maxent (local nbrs): 96.8% / 86.8%
  - MEMMs: 96.9% / 86.9%
  - Linear Perceptron: 96.7% / ??
  - Cyclic tagger: 97.2% / 89.0%
  - Maxent+Ext ambig. 97.4% / 91.3%
  - Upper bound: ~98%

Summary of POS Tagging

For tagging, the change from generative to discriminative model **does not by itself** result in great improvement

One profits from models for specifying dependence on **overlapping features of the observation** such as spelling, suffix analysis, etc.

An MEMM allows integration of rich features of the observations, but can suffer strongly from assuming independence from following observations; this effect can be relieved by adding dependence on following words

This additional power (of the MEMM ,CRF, Perceptron models) has been shown to result in improvements in accuracy

The **higher accuracy** of discriminative models comes at the price of **much slower training**

Simple MaxEnt models perform close to state of the art

What does it say about the sequence labeling task?

Domain Effects

- Accuracies degrade outside of domain
  - Up to triple error rate
  - Usually make the most errors on the things you care about in the domain (e.g. protein names)

- Open questions
  - How to effectively exploit unlabeled data from a new domain (what could we gain?)
  - How to best incorporate domain lexica in a principled way (e.g. UMLS specialist lexicon, ontologies)