Schoolhouse Rock

Reminders

QUIZ 5 IS DUE TONIGHT BY 11:59PM (NO LATE DAYS)

HW6 IS DUE ON WEDNEDAY

Part of Speech Tagging

JURAFSKY AND MARTIN CHAPTER 8

Ancient Greek tag set

(c. 100 BC)

Noun

Verb

Pronoun

Preposition

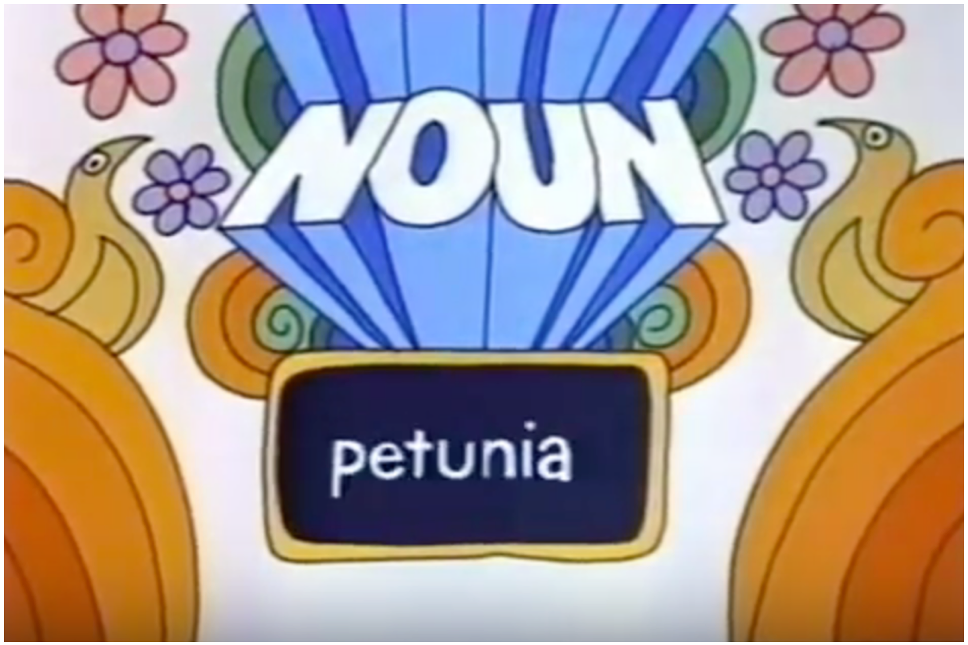
Adverb

Conjunction

Participle

Article

Schoolhouse Rock tag set



(c. 1970)

Noun

Verb

Pronoun

Preposition

Adverb

Conjunction

~~Participle~~

~~Article~~

Adjective

Interjection

Word classes

Every word in the vocabulary belongs to one or more of these word classes.

Assigning the classes to words in a sentence is called part of speech (POS) tagging.

Many words can have multiple POS tags.
Can you think of some?

Open classes

Four major classes:

1. Noun
2. Verbs
3. Adjectives
4. Adverbs

English has all four but not every language does.

Nouns

Person, place or thing.

Proper nouns: names of specific entities or people.

Common nouns

- **Count nouns -** allow grammatical enumeration, occurring in both singular and plural.
- **Mass nouns -** conceptualized as homogenous groups. Cannot be pluralized. Can appear without determiners even in singular form.

Verbs

Words describing actions and processes.

English verbs have inflectional markers.

3rd person singular		
Non-3rd person singular		
Progressive (ing)		
Past		

Verbs

Words describing actions and processes.

English verbs have inflectional markers.

	Root: compute	suffix
3rd person singular	He/she/it computes	+s
Non-3rd person singular	They/you/I compute	__
Progressive (ing)	Computing	+ing
Past	Computed	+ed

Adjectives

Word that describe properties or qualities.

Matthew Anderson
@MattAndersonNYT

Things native English speakers know, but don't know we know:

> adjectives in English absolutely have to be in this order: opinion-size-age-shape-colour-origin-material-purpose Noun. So you can have a lovely little old rectangular green French silver whittling knife. But if you mess with that word order in the slightest you'll sound like a maniac. It's an odd thing that every English speaker uses that list, but almost none of us could write it out. And as size comes before colour, green great dragons can't exist.

78.3K 4:26 AM - Sep 3, 2016

52.8K people are talking about this

Adverb

Modify verbs or whole verb phrases or other words like adjectives

	Examples
Locatives	here, home, uphill
Degree	Very, extremely, extraordinarily, somewhat, not really, --ish
Manner	slowly, quickly, softly, gently, alluringly
Temporal	yesterday, Monday, last semester

Closed Classes

numerals	one, two, *n*th, first, second, …
prepositions	of, on, over, under, to, from, around
determiners	indefinite: some, a, an definite: the, this, that, the
pronouns	she, he, it, they, them, who, whoever, whatever
conjunctions	and, or, but
particles (preposition joined to a verb)	knocked **over**
auxiliary verbs	was

Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	**SYM**	symbol	*+, %, &*
CD	cardinal number	*one, two*	**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	proposition/sub-conj	*of, in, by*	**VBG**	verb gerund	*eating*
JJ	adjective	*yellow*	**VBN**	verb past participle	*eaten*
JJR	comparative adjective	*bigger*	**VBP**	verb non-3sg pres	*eat*
JJS	superlative adjective	*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, singular or mass	*llama*	**WP$**	possessive wh-	*whose*
NNS	noun, plural	*llamas*	**WRB**	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	**$**	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	**#**	pound sign	*#*
PDT	predeterminer	*all, both*	**“**	left quote	*‘ or “*
POS	possessive ending	*‘s*	**”**	right quote	*’ or ”*
PRP	personal pronoun	*I, you, we*	**(**	left parenthesis	*[, (, {, <*
PRP$	possessive pronoun	*your, one’s*	**)**	right parenthesis	*],), }, >*

POS Tagging

Words are ambiguous, so tagging must resolve disambiguate.

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%**)

The amount of tag ambiguity for word types in the Brown and WSJ corpora from the Treebank-3 (45-tag) tagging. These statistics include punctuation as words, and assume words are kept in their original case.

Some words have up to 6 tags

	Sentence	Tag
1	Earnings took a **back** seat	
2	A small yard in the **back**	
3	Senators **back** the bill	
4	He started to **back** towards the door	
5	To **buy back** stock.	
6	I was young **back** then.	

Corpora with manual POS tags

Brown corpus – 1 million words of 500 written English texts from different genres.

WSJ corpus – 1 million words from the Wall Street Journal

Switchboard corpus – 2 million words of telephone conversations

The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

There/EX are/VBP 70/CD children/NNS **there/RB**

Most frequent class baseline

Many words are easy to disambiguate, because their different tags aren't equally likely.

Simplistic baseline for POS tagging: given an ambiguous word, choose the tag which is most frequent in the training corpus.

Most Frequent Class Baseline: Always compare a classifier against a baseline at least as good as the most frequent class baseline (assigning each token to the class it occurred in most often in the training set).

How good is the baseline?

This lets us know how hard the task is (and how much room for improvement real models have).

Accuracy for POS taggers is measured as the percent of tags that are correctly labeled when compared to human labels on a test set.

Most Frequent Class Baseline:	92%
State of the art in POS tagging:	97%

(Much harder for other languages and other genres)

Hidden Markov Models (HMMs)

The HMM is a probabilistic **sequence model**.

A sequence model assigns a label to each unit in a sequence, mapping a sequence of observations to a sequence of labels.

Given a sequence of words, an HMM computes a probability distribution over a sequence of POS tags.

Sequence Models

A **sequence model** or **sequence classifier** is a model whose job is to assign a label or class to each unit in a sequence, thus mapping a sequence of observations to a sequence of labels.

A Hidden Markov Model (HMM) is a probabilistic sequence model: given a sequence of words, it computes a probability distribution over possible sequences of labels and chooses the best label sequence.

What is hidden?

We used a Markov model in n-gram LMs. This kind of model is sometimes called a Markov chain. It is useful when we need to compute a probability for a sequence of observable events.

In many cases the events we are interested in are **not observed** directly. We don't see part-of-speech tags in a text. We just see words, and need to infer the tags from the word sequence.

We call the tags **hidden** because they are **not observed**.

HMMs for tagging

Basic equation for HMM tagging

$$\hat{t}_1^N = \arg\max_{t_1^N} P(t_1^N | w_1^N)$$

Find the best (hidden) **tag sequence** $\boldsymbol{t_1^N}$, given an (observed) **word sequence** $\boldsymbol{w_1^N}$
where N = number of words in the sequence

Use Bayes rule

$$= \arg\max_{t_1^N} \frac{P\left(w_1^N \middle| t_1^N\right) P(t_1^N)}{P(w_1^N)}$$

$$= \arg\max_{t_1^N} P(w_1^N | t_1^N) P(t_1^N)$$

Simplifying Assumptions

1. **Output Independence:** Probability of a word only depends on its own tag, and it is independent of neighboring word and tags

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

2. **Markov assumption**: The probability of a tag depends only on previous tag, not the whole tag sequence.

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

Simplifying Assumptions

1. **Output Independence:** Probability of a word only depends on its own tag, and it is independent of neighboring word and tags

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

Emission probability

2. **Markov assumption**: The probability of a tag depends only on previous tag, not the whole tag sequence.

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

Transition probability

Combining:

$$\hat{\boldsymbol{t}}_1^N = \mathbf{arg\,max}_{t_1^N}\ \boldsymbol{P}(\boldsymbol{t}_1^N|\boldsymbol{w}_1^N) \approx \mathbf{arg\,max}_{t_1^N} \prod_{i=1}^{N} \boldsymbol{P}(\boldsymbol{w}_i|\boldsymbol{t}_i)\ \boldsymbol{P}\,(\boldsymbol{t}_i|\boldsymbol{t}_{i-1})$$

HMM Tagger Components

Transition probability

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

In the WSJ corpus, a modal verb (MD) occurs 13,124 times. 10,471 times the MD is followed by a verb (VB). Therefore,

$$P(VB|MD) = \frac{10{,}471}{13{,}124} = .80$$

Transition probabilities are sometimes called the **A probabilities**.

HMM Tagger Components

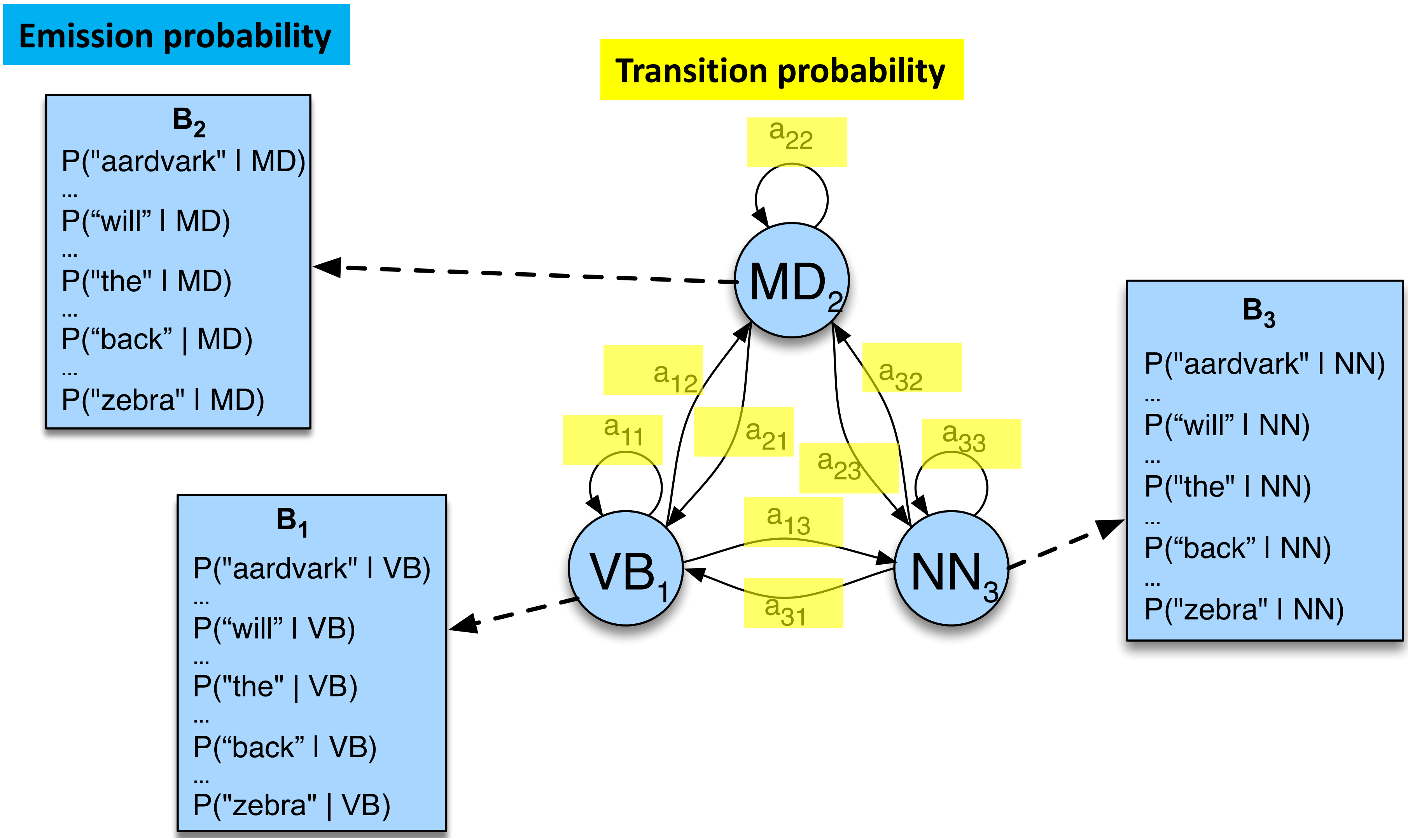
Emission probability

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

Of the 13,124 occurrences of modal verbs (MD) in the WSJ corpus, the word ***will*** represents 4,046 of the words tagged as MD.

$$P(will|MD) = \frac{4{,}046}{13{,}124} = .31$$

Emission probabilities are sometimes called the **B probabilities**.



Emission probability
Transition probability
B2
P("aardvark" I MD)
...
P("will" I MD)
...
P("the" I MD)
...
P("back" | MD)
...
P("zebra" I MD)
a22
MD2
a12
a21
a32
a11
a23
a33
a13
a31
VB1
NN3
B1
P("aardvark" I VB)
...
P("will" I VB)
...
P("the" | VB)
...
P("back" I VB)
...
P("zebra" | VB)
B3
P("aardvark" I NN)
...
P("will" I NN)
...
P("the" I NN)
...
P("back" I NN)
...
P("zebra" I NN)

HMM decoding

For a model with hidden variables, the task of determining the hidden variables sequence corresponding to the sequence of observations is called "**decoding**".

Decoding: Given an HMM $\lambda = (A, B)$ and a sequence of observations $O = w_1, w_2, ..., w_T$, find the most probable sequence of states $Q = t_1 t_2 t_3 ... t_T$.

$$\hat{t}_1^N = \mathbf{arg\,max}_{t_1^N} P(w_1^N | t_1^N) P(t_1^N)$$

HMM decoding

Input:	Let	us	learn	about	HMMs
Output Best Labels:	VB	PRP	VB	IN	NNP

Compute probability for **all possible sequence of labels:**

Let	us	learn	about	HMMs	
VB	PRP	VB	IN	NNP	p=0.45
IN	VB	VB	NN	DT	p=0.03
		…			
PRP	.	NN	IN	WP	p=0.00006

How many label sequences?

T observations

Input:	Let	us	learn	about	HMMs
	CC	CC	CC	CC	CC
	CD	CD	CD	CD	CD
	DT	DT	DT	DT	DT
	EX	EX	EX	EX	EX
	FW	FW	FW	FW	FW
	IN	IN	IN	IN	IN
	JJ	JJ	JJ	JJ	JJ
	JJR	JJR	JJR	JJR	JJR
N states	JJS	JJS	JJS	JJS	JJS
	LS	LS	LS	LS	LS
	MD	MD	MD	MD	MD
	NN	NN	NN	NN	NN
	NNS	NNS	NNS	NNS	NNS
	NNP	NNP	NNP	NNP	NNP
	NNPS	NNPS	NNPS	NNPS	NNPS
	PDT	PDT	PDT	PDT	PDT

How many label sequences?

T observations

Input: Let us learn about HMMs

Let	us	learn	about	HMMs
CC	CC	CC	CC	CC
CD	CD	CD	CD	CD
DT	DT	DT	DT	DT

For POS tagging a sentence of length T = 5, and number of states (tags) = 45

$$N^T = 60{,}466{,}176$$

N states

JJS	JJS	JJS	JJS	JJS
LS	LS	LS	LS	LS
MD	MD	MD	MD	MD
NN	NN	NN	NN	NN
NNS	NNS	NNS	NNS	NNS
NNP	NNP	NNP	NNP	NNP
NNPS	NNPS	NNPS	NNPS	NNPS
PDT	PDT	PDT	PDT	PDT

Dynamic Programming

Coined by Richard Bellman in 1940s

> "My boss, Secretary of Defense, actually had a pathological fear and hatred of the word 'research'. *Dynamic* has a very interesting property as an adjective, and that it's impossible to use the word dynamic in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible!"

Method for solving complex problems by breaking them down into simpler sub-problems and storing their solutions

Technique of storing solutions to sub-problems instead of recomputing them is called "**memoization**"

Dynamic Programming

Fibonacci Series

$$fib(n) = fib(n-1) + fib(n-2)$$

- fib(5)
- fib(4) + fib(3)
- (fib(3) + fib(2)) + (fib(2) + fib(1))
- ((fib(2) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))
- (((fib(1) + fib(0)) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))

Instead of calling fib(3) multiple times, we should store it and lookup instead of recomputing

Viterbi Algorithm

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
  $viterbi[s,1] \leftarrow \pi_s \, * \, b_s(o_1)$
  $backpointer[\mathrm{s},1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[\mathrm{s,t}] \leftarrow \max_{s'=1}^{N} \; viterbi[s',t-1] \, * \, a_{s',s} \, * \, b_s(o_t)$
    $backpointer[\mathrm{s,t}] \leftarrow \operatorname{argmax}_{s'=1}^{N} \; viterbi[s',t-1] \, * \, a_{s',s} \, * \, b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} \; viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} \; viterbi[s,T]$; termination step
bestpath $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Viterbi Algorithm

function VITERBI(*observations* of len *T*,*state-graph* of len *N*) **returns** *best-path*, *path-prob*

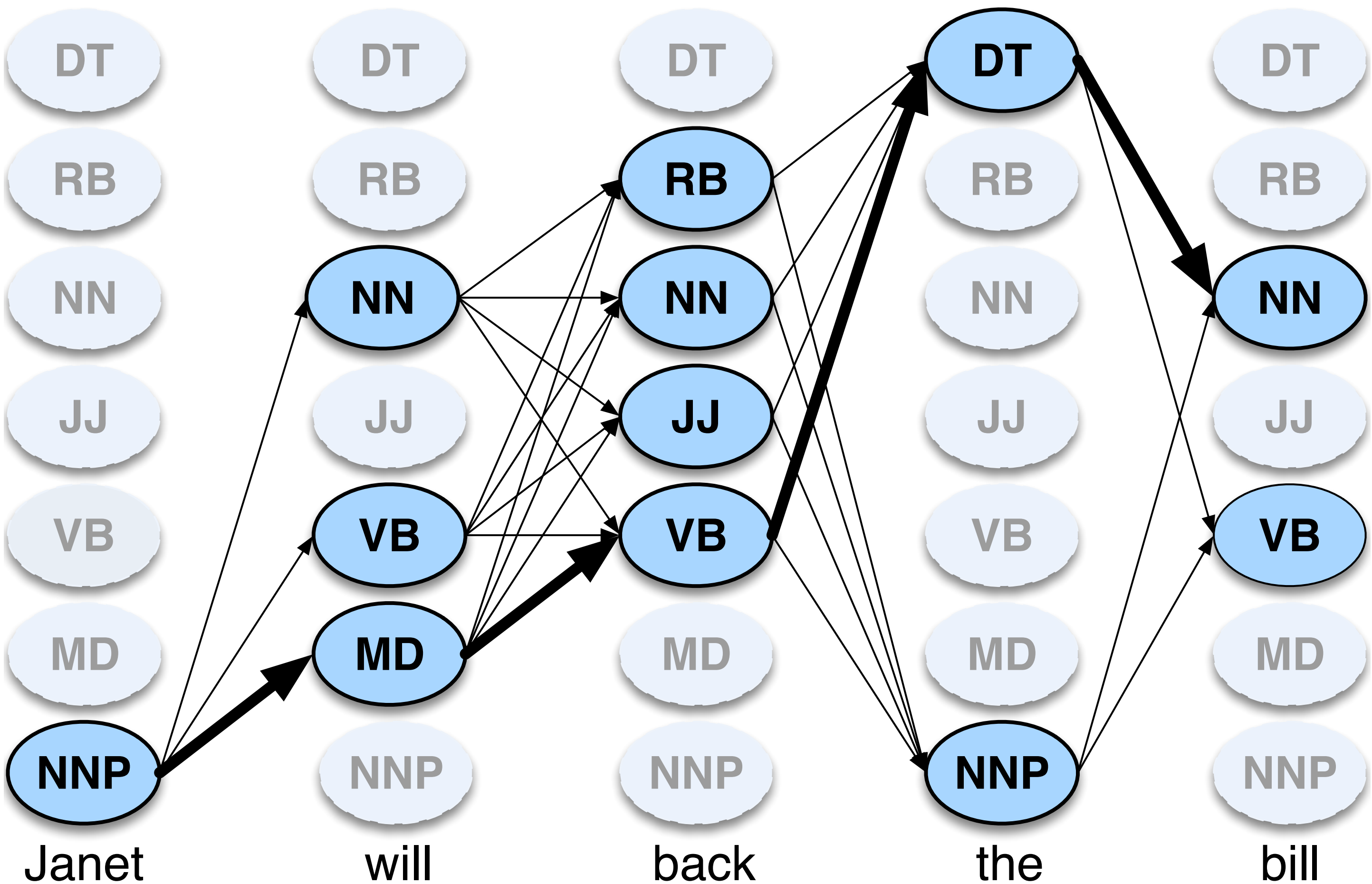
The complexity of the Viterbi algorithm for this HMM is $O(T * N^2)$.
So POS tagging a sentence of length T = 5 with N = 45 states (tags) goes from:

$N^T = 60{,}466{,}176$

to computations

$T * N^2 = 10{,}125$ computations!

Viterbi Lattice



Trigram HMMs

So far, we had a bigram assumption. The probability of a tag depends only on previous tag, not the whole tag sequence.

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

We could extend it to a trigram model

$$P(t_1^N) \approx \prod_{i=1}^{N} p\,(t_i | t_{i-1}, t_{i-2})$$

Trigram HMMs

So far, we had a bigram assumption. The probability of a tag depends only on previous tag, not the whole tag sequence.

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

We could extend it to a trigram model

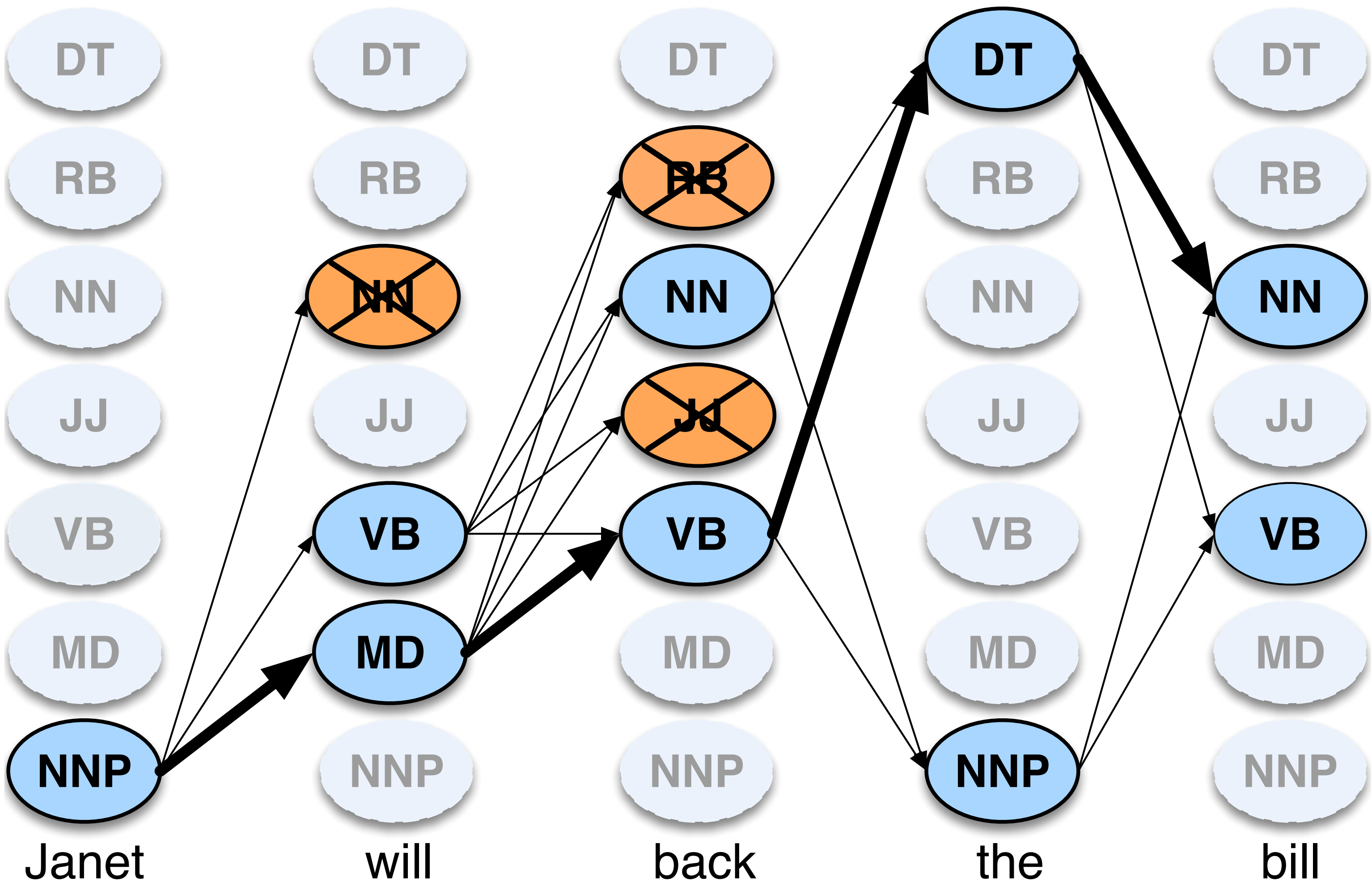
The complexity of the trigram HMM increases from $O(N^2T)$ to $O(N^3T)$. The number of states (N) gets larger since we have to compare every pair of 45 tags, instead of just each tag, so we have 45^3 = **91,125 computations per column.**

Beam Search

One common solution to the complexity problem is the use of **beam search decoding**. Instead of keeping the entire column of states at each time point *t*, beam search just keeps the best few hypothesis.

At time *t* this requires computing the Viterbi score for each of the *N* cells, sorting the scores, and **keeping only the best-scoring states**. The rest are pruned out and not continued forward to time *t*+1.

Beam Search



Unknown words

To achieve high accuracy with POS taggers, it is also important to have a good model for dealing with **unknown words**.

Proper names and acronyms are created very often, and even new common nouns and verbs enter the language at a surprising rate.

Unknown words

One useful feature for distinguishing parts of speech is **word shape** (proper nouns start with a capital).

The strongest feature is **morphology**.

Words that end in

- **-s** tend to be **plural nouns (NNS)**
- **-ed** tend to be **past participles (VBN)**
- **-able** tend to be **adjectives (JJ)**
- and so on

Learning suffix model

Store the final letter sequence (suffixes) for up to 10 letters.

For each such sequence, record the probability of the tag that it was associated with during training.

Use back-off to smooth these probabilities for. Successively shorter sequences.

Trigram HMM with unknown word handling:	96.7%
State-of-the-art neural network POS tagging:	97%

Maximum Entropy Markov Models

Could we add features like word shape and suffixes directly into the model in a clean way? We had this for classification with **logistic regression**. But it's not a sequence model, since it assigns a class to a single observation.

We can turn it into a **discriminative sequence model** by running it on successive words, using the class assigned to the prior word as a feature in the classification of the next word. This is called a Maximum Entropy Markov Model **(MEMM).**

MEMMs v HMMs

HMM:

$$\hat{T} = argmax_T \, P(T|W)$$

$$= argmax_T \, P(W|T)P(T)$$

$$= argmax_T \prod_i P\,(word_i|tag_i) \prod_i P\,(tag_i|tag_{i\,-1})$$

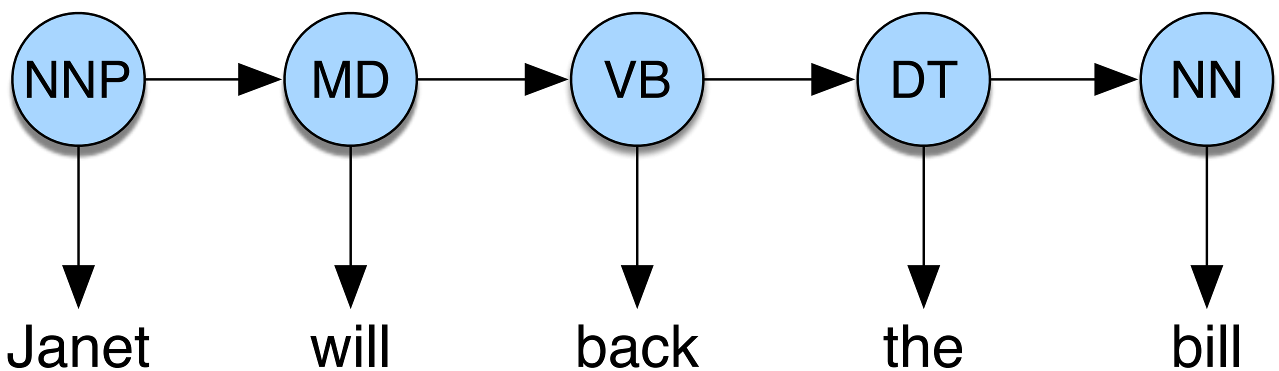
MEMM:

$$\hat{T} = argmax_T \, P(T|W)$$

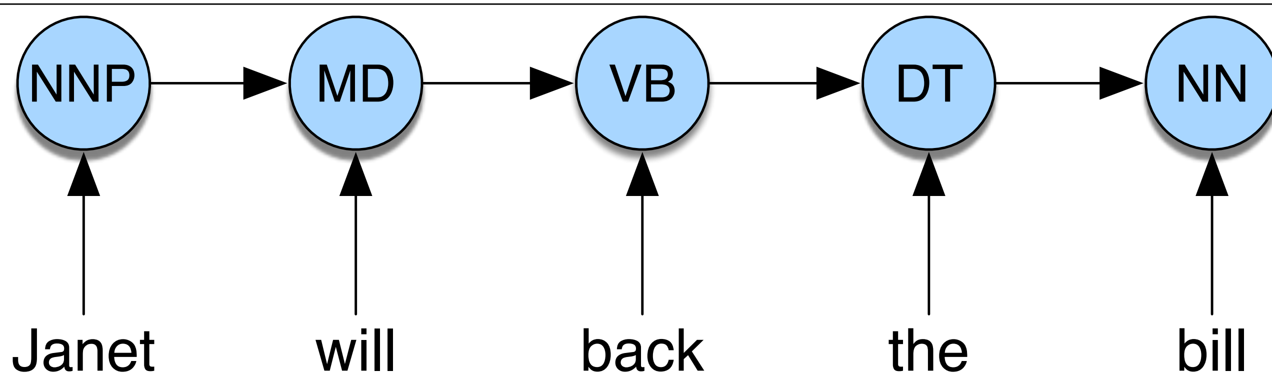
$$= argmax_T \prod_i P\,(tag_i|word_i, tag_{i\,-1})$$

MEMMs v HMMs

HMM:

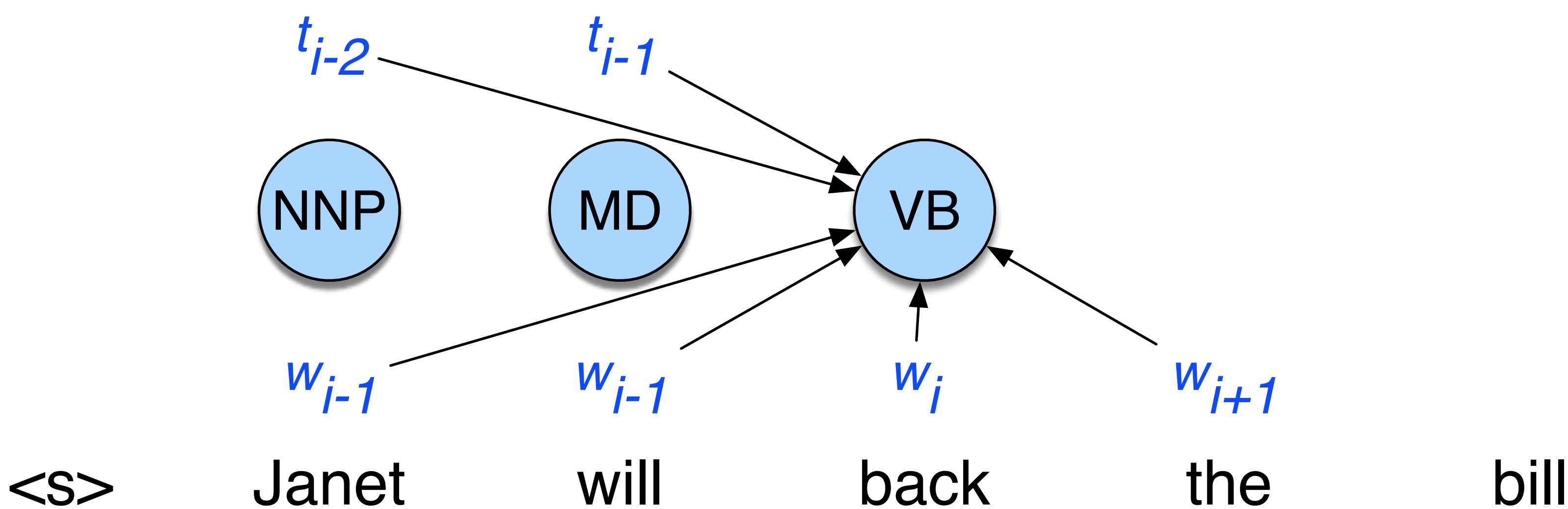


MEMM:



Features in a MEMM

We can build MEMMs that don't just condition on w_i and t_{i-1}. It is easy to incorporate lots of features in a discriminative sequence model.



Feature templates

A basic MEMM part-of-speech tagger conditions on the observation word it- self, neighboring words, and previous tags, and various combinations, using feature templates like the following

$$< t_i, w_{i-2} >, < t_i, w_{i-1} >, < t_i, w_i >, < t_i, w_{i+1} >, < t_i, w_{i+2} >$$
$$< t_i, t_{i-1} >, < t_i, t_{i-2}, t_{i-1} >$$
$$< t_i, t_{i-1}, w_i >, < t_i, w_{i-1}, w_i >, < t_i, w_i, w_{i+1} >$$

Janet/NNP will/MD back/VB the/DT bill/NN, when w_i is the word *back*

$t_i = VB$ and $w_{i-2} = Janet$

$t_i = VB$ and $w_{i-1} = will$

$t_i = VB$ and $w_i = back$

$t_i = VB$ and $w_{i+1} = the$

$t_i = VB$ and $w_{i+2} = bill$

$t_i = VB$ and $t_{i-1} = MD$

$t_i = VB$ and $t_{i-1} = MD$ and $t_{i-2} = NNP$

$t_i = VB$ and $w_i = back$ and $w_{i+1} = the$

Features for unknown words

w_i contains a particular prefix (from all prefixes of length ≤ 4)

w_i contains a particular suffix (from all suffixes of length ≤ 4)

w_i contains a number

w_i contains an upper-case letter

w_i contains a hyphen

w_i is all upper case

$w_i's$ word shape

$w_i's$ short word shape

w_i is upper case and has a digit and a dash (like *CFC-12*)

w_i is upper case and followed within 3 words by Co., Inc., etc.

Features for *well-dressed*

$\text{prefix}(w_i) = w$

$\text{prefix}(w_i) = w\text{e}$

$\text{prefix}(w_i) = w\text{el}$

$\text{prefix}(w_i) = well$

$\text{suffix}(w_i) = ssed$

$\text{suffix}(w_i) = sed$

$\text{suffix}(w_i) = ed$

$\text{suffix}(w_i) = d$

$\text{h}as - hyphen(w_i)$

$\text{word} - shape(w_i) = xxxx\ - xxxxxxx$

$short - word - shape(w_i) = x\ - x$

Morphologically Rich Languages

Both morphologically rich and highly inflectional languages are challenging since they have a large vocabulary: a 250,000 word token corpus of Hungarian has **more than twice as many word types** as a similarly sized corpus of English.

For these languages, POS taggers need to label words with case and gender information as well, resulting in **novel tagsets in the form of sequences of morphological tags** rather than a single tag.

Ex. Üzerinde parmak **izin** kalmiş *(iz + Noun + A3sg + P2sg + Nom)*