Natural Language Processing

Sequence labeling

Yulia Tsvetkov

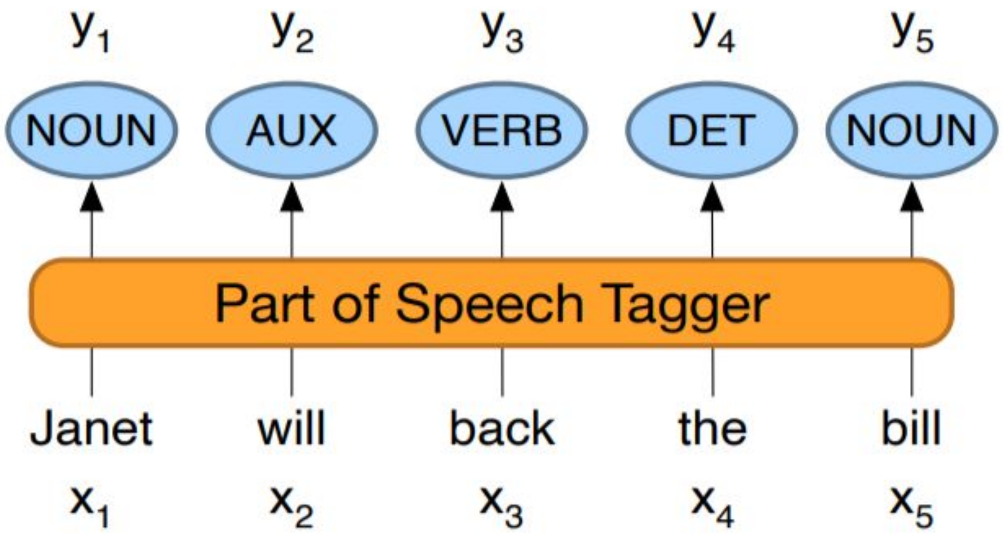
yuliats@cs.washington.edu

PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING

Announcements

- HW 2 review

Part of speech tagging



Why POS tagging

- Goal: resolve ambiguities
- Text-to-speech
  - record, lead, protest
- Lemmatization
  - saw/V → see, saw/N → saw
- Machine translation – the meaning of a word depends on its POS tag
  - "what is your address" vs. "we should address this comment"
  - "grand challenge" vs. "challenge the team"
- Sentiment analysis – adjectives are the major opinion holders
  - "good" vs. "bad", "excellent" vs. "terrible"
- Preprocessing for harder disambiguation problems
  - syntactic parsing
  - semantic parsing

Parts of speech

- Open classes
  - nouns
  - verbs
  - adjectives
  - adverbs

- Closed classes
  - prepositions
  - determiners
  - pronouns
  - conjunctions
  - auxiliary verbs

Parts of speech, more fine-grained classes

- Open classes
  - nouns
    - proper
    - common
      - count
      - mass
  - verbs
  - adjectives
  - adverbs
    - directional
    - degree
    - manner
    - temporal

Actually, I ran home *extremely quickly yesterday*

Parts of speech, closed classes

prepositions: on, under, over, near, by, at, from, to, with
particles: up, down, on, off, in, out, at, by
determiners: a, an, the
conjunctions: and, but, or, as, if, when
pronouns: she, who, I, others
auxiliary verbs: can, may, should, are
numerals: one, two, three, first, second, third

Part of speech tagsets

- Penn treebank tagset (Marcus et al., 1993)

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	: ; ... – -

Example of POS tagging

- There are 7 children there.

There/PRON are/AUX 70/NUM children/NOUN there/ADV ./PUNC

- Preliminary findings were reported in today's NY TIMES.

Preliminary/ADJ findings/NOUN were/AUX reported/VERB in/ADP today/NOUN 's/PART NY/PROPN TIMES/PROPN ./PUNC

The Universal Dependencies

Universal Dependencies

Universal Dependencies (UD) is a framework for consistent annotation of grammar (parts of speech, morphological features, and syntactic dependencies) across different human languages. UD is an open community effort with over 300 contributors producing more than 150 treebanks in 90 languages. If you're new to UD, you should start by reading the first part of the Short Introduction and then browsing the annotation guidelines.

- Short introduction to UD
- UD annotation guidelines
- More information on UD:
  - How to contribute to UD
  - Tools for working with UD
  - Discussion on UD
  - UD-related events
- Query UD treebanks online:
  - SETS treebank search maintained by the University of Turku
  - PML Tree Query maintained by the Charles University in Prague
  - Kontext maintained by the Charles University in Prague
  - Grew-match maintained by Inria in Nancy
  - INESS maintained by the University of Bergen
- Download UD treebanks

Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

The Universal Dependencies

	tag	tag name	description	examples
open class	ADJ	Adjective	noun modifiers descrfibing properties	red, young, awesome
	ADV	Adverb	verb modifiers of time, place, manner	very, slowly, home (as in “go home”), yesterday
	NOUN	Noun	words for persons, places, things, etc.	algorithm, cat, mango, beauty
	VERB	Verb	words for actions and processes	draw, provide, run
	PROPN	Proper noun	name of a person, organization, location, etc.	Regina, IBM, Vietnam
	INTJ	Interjection	exclamation, greeting, yes/no response, etc.	oh, um, yes, hello
closed class	ADP	Adposition	(preposition/postposition): marks a noun’s spatial, temporal, or other relation	in, on, by, under
	AUX	Auxiliary	helping a verb making tense, aspect, mood, etc.	can, may, should, are
	CCONJ	Coordinating conjunction	joins two phrases or clauses	and, or, but
	DET	Determiner	marks noun phrase properties	a, an, the, this
	NUM	Numeral		one, two, first, second
	PART	Particle	a preposition-like form used together with a verb	up, down, on, off, in, out, at, by
	PRON	Pronoun	a shorthand for referring to an entity or event	you, she, who, I others
	SCONJ	Suborinating conjunction	joins a main clause with a subordinate clause such as sentential complement	that, which
other	PUNCT	Punctuation		; ! ? ()
	SYM	Symbol	various symbols	$ %
	X	Other		all the rest...

Universal Dependencies tagset (Nivre et al., 2016) (17 tags)

Ambiguities in POS tags

Types:		WSJ	Brown
Unambiguous	(1 tag)	44,432 (**86%**)	45,799 (**85%**)
Ambiguous	(2+ tags)	7,025 (**14%**)	8,050 (**15%**)

Ambiguities in POS tags

		WSJ	Brown
Types:			
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%**)

POS-tagging is a disambiguation task

- look at this small building in the back/NN
- the main player in the team took a back/JJ seat
- a majority of senators will back/VERB the decision
- we should enable the country to buy back/PART its debt
- I was twenty-one back/ADV then

- try yourself with an online POS tagger, e.g., https://parts-of-speech.info/

POS tagging algorithms

- We will introduce classic and state-of-the-art sequence labeling approaches
  - Hidden Markov Model (HMM) + Viterbi, generative
  - RNNs + Conditional Random Field (CRF), discriminative

Most frequent class baseline

- A baseline: assign each word its most probable (frequent) tag

- Always compare a classifier against a baseline at least as good as the most frequent class baseline

- The WSJ training corpus and test on sections 22-24 of the same corpus the most-frequent-tag baseline achieves an accuracy of 92.34%.

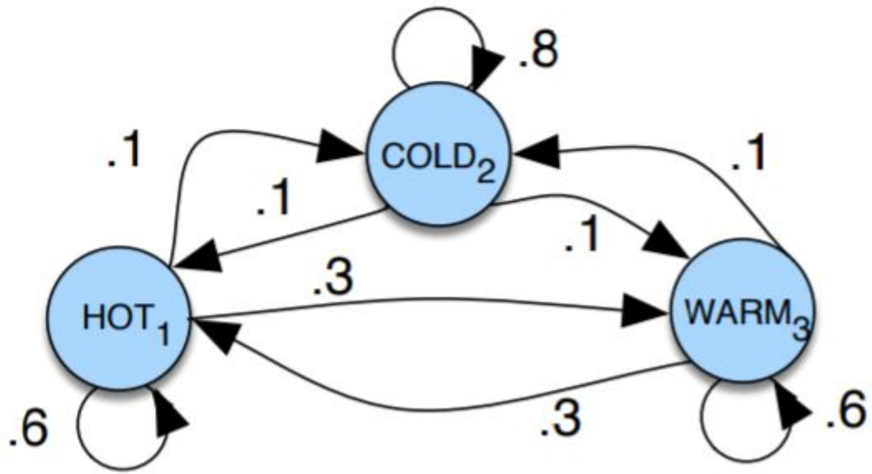
- 97% tag accuracy achievable by most algorithms (HMMs, MEMMs, neural networks, rule-based algorithms)

Sequence labeling as text classification

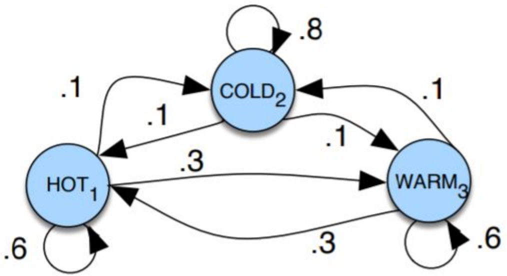
$$\hat{y}_i = \underset{y \in \mathcal{L}}{\operatorname{argmax}}\, s(\boldsymbol{x}, i, y)$$

Generative sequence labeling: Hidden Markov Models

Markov Chain: weather



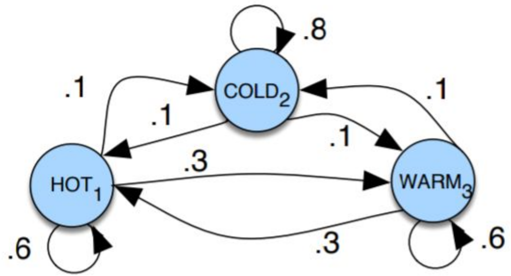
Markov Chain: weather



- A model that assigns probabilities to sequences of random variables
  - Each variable can take a value from some finite set
- Markov assumption: only the current state matters for predicting the next state, all states before the current have no impact
- For a set of variables: $q_1, \quad q_2, \quad \ldots, \quad q_i$

$$P(q_i = a|q_1, \ldots, q_{i-1}) = P(q_i = a|q_{i-1})$$

Markov Chain: weather



- Given initial probabilities of [0.1, 0.7, 0.2] compute the probability for:
  - HOT WARM WARM HOT
  - HOT COLD HOT COLD

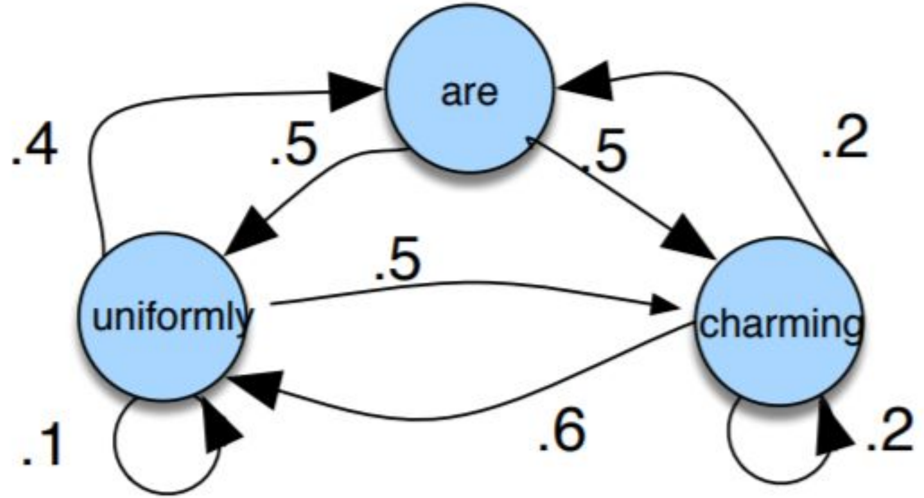
Markov chain

- Formally, a Markov chain specified by the following components

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

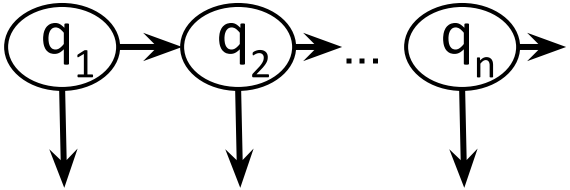
Markov Chain: words

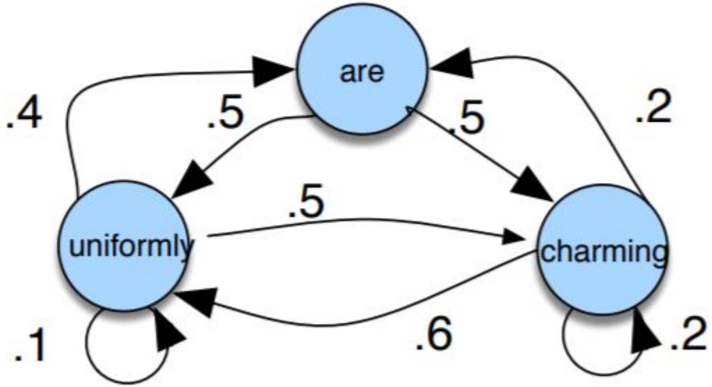
$$\pi = [0.1, 0.7, 0.2]$$



- only the current state matters for predicting the next state, all states before the current have no impact

Markov Chain: words

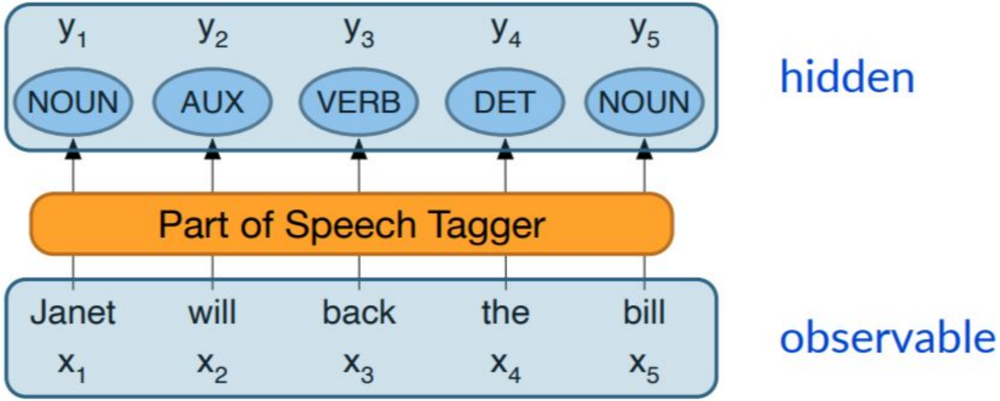




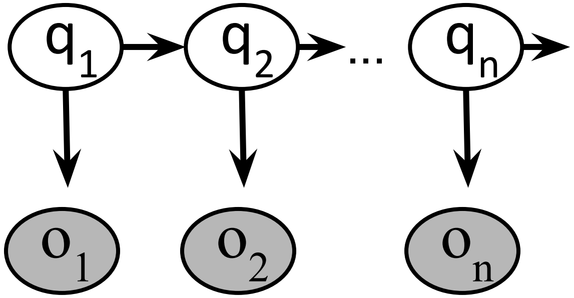
- only the current state matters for predicting the next state, all states before the current have no impact

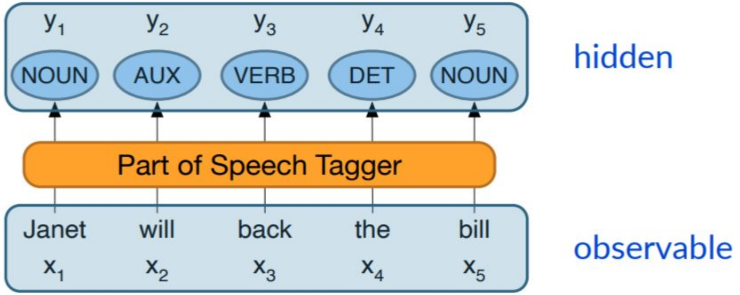
Hidden Markov Models

- We use a Markov chain for computing P for a sequence of observable events
- In many cases the events we are interested in are hidden
  - e.g., we don't observe POS tags in a text



Hidden Markov Models

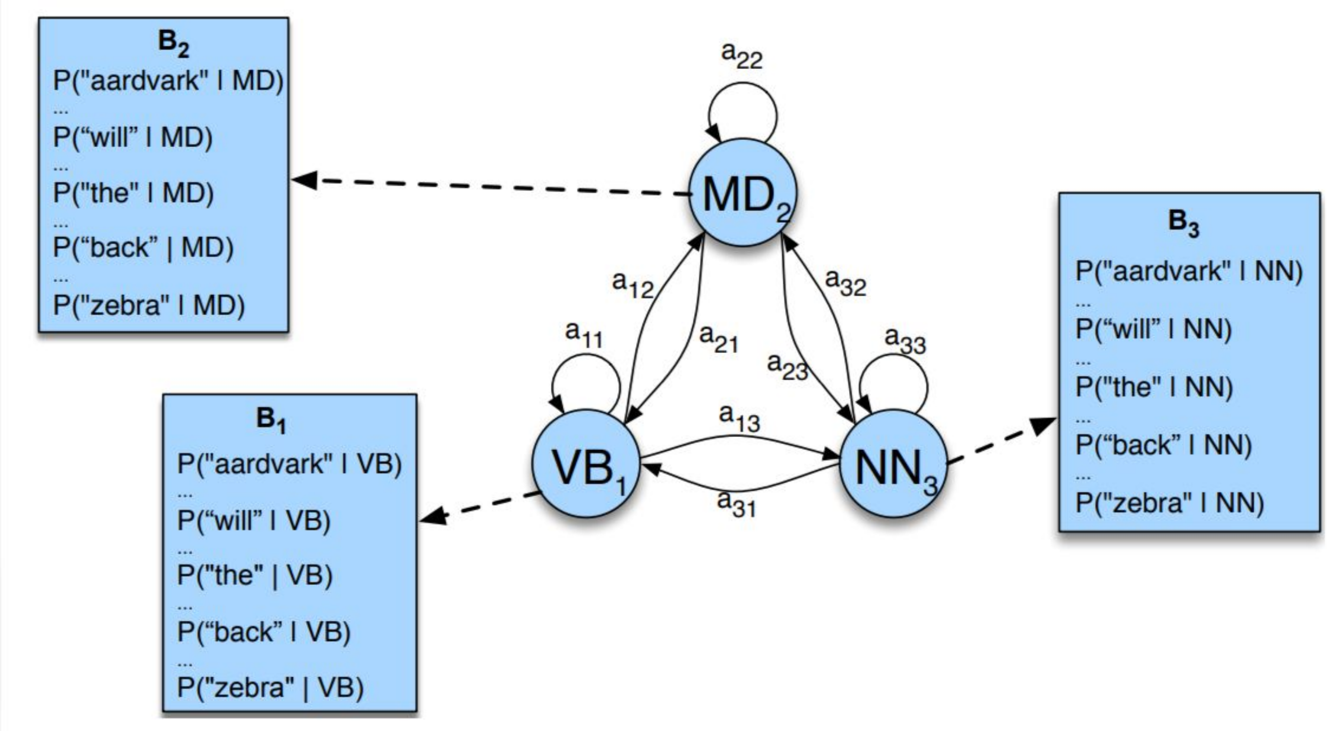




Markov Assumption: $P(q_i|q_1...q_{i-1}) = P(q_i|q_{i-1})$

Output Independence: $P(o_i|q_1 \dots q_i, \dots, q_T, o_1, \dots, o_i, \dots, o_T) = P(o_i|q_i)$

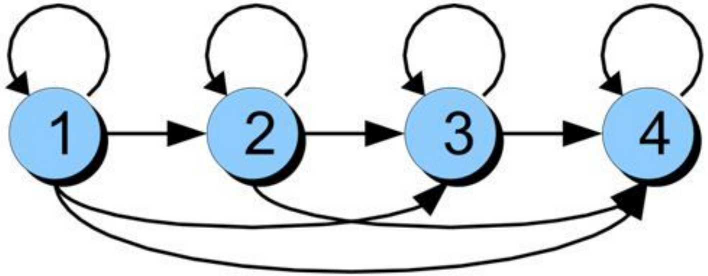
Hidden Markov Models



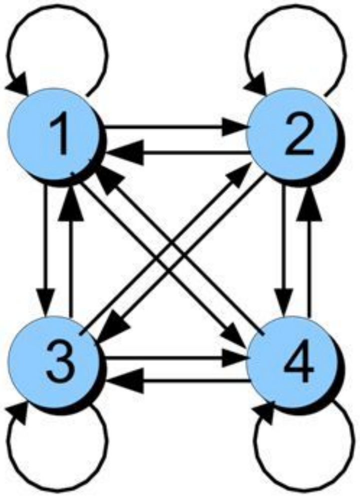
Hidden Markov Models (HMMs)

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

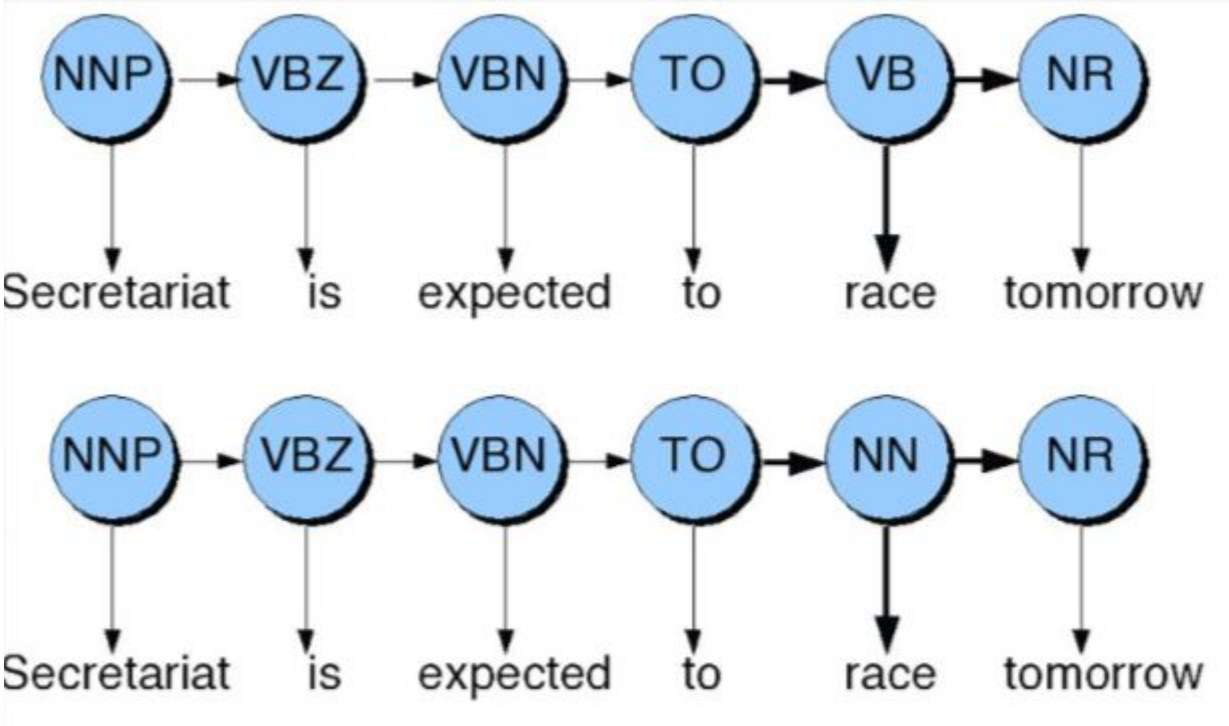
Types of HMMs



Bakis = left-to-right



Ergodic = fully-connected


NNP
VBZ
VBN
TO
VB
NR
Secretariat
is
expected
to
race
tomorrow
NNP
VBZ
VBN
TO
NN
NR
Secretariat
is
expected
to
race
tomorrow

HMMs in language technologies

- Part-of-speech tagging (Church, 1988; Brants, 2000)
- Named entity recognition (Bikel et al., 1999) and other information extraction tasks
- Text chunking and shallow parsing (Ramshaw and Marcus, 1995)
- Word alignment of parallel text (Vogel et al., 1996)
- Acoustic models in speech recognition (emissions are continuous)
- Discourse segmentation (labeling parts of a document)

HMM parameters

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

HMMs: algorithms

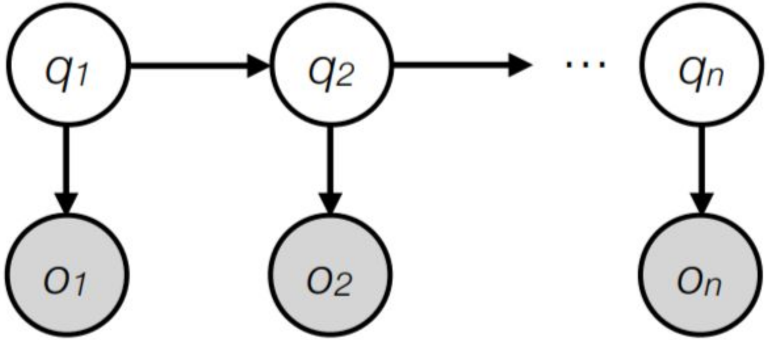
Forward	**Problem 1 (Likelihood):**	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Viterbi	**Problem 2 (Decoding):**	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Forward-backward; Baum-Welch	**Problem 3 (Learning):**	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

Forward	**Problem 1 (Likelihood):**	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Viterbi	**Problem 2 (Decoding):**	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Forward-backward; Baum-Welch	**Problem 3 (Learning):**	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

HMM tagging as decoding

Decoding: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n \mid w_1^n)$$



HMM tagging as decoding

Decoding: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n \mid w_1^n)$$

$$= \underset{t_1^n}{\operatorname{argmax}}\, \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)}$$

HMM tagging as decoding

Decoding: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n)$$

$$= \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n \mid t_1^n) P(t_1^n)}{P(w_1^n)}$$

$$= \underset{t_1^n}{\operatorname{argmax}} P(w_1^n \mid t_1^n) P(t_1^n)$$

HMM tagging as decoding

Decoding: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n \mid w_1^n)$$

$$= \underset{t_1^n}{\operatorname{argmax}}\, \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$= \underset{t_1^n}{\operatorname{argmax}}\, P(w_1^n \mid t_1^n)P(t_1^n)$$

simplifying assumptions:

HMM tagging as decoding

Decoding: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n \mid w_1^n)$$

$$= \underset{t_1^n}{\operatorname{argmax}}\, \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$= \underset{t_1^n}{\operatorname{argmax}}\, P(w_1^n \mid t_1^n)P(t_1^n)$$

simplifying assumptions:

$$P(w_1^n \mid t_1^n) \approx \prod_{i=1}^{n} P(w_i \mid t_i)$$

HMM tagging as decoding

Decoding: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n)$$

$$= \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$= \underset{t_1^n}{\operatorname{argmax}} P(w_1^n \mid t_1^n)P(t_1^n)$$

simplifying assumptions:

$$P(w_1^n \mid t_1^n) \approx \prod_{i=1}^{n} P(w_i \mid t_i)$$

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i \mid t_{i-1})$$

HMM tagging as decoding

Decoding: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \underbrace{P(w_i \mid t_i)}_{\text{emission, } B} \underbrace{P(t_i \mid t_{i-1})}_{\text{transition, } A}$$

HMM tagging as decoding

Decoding: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \underbrace{P(w_i \mid t_i)}_{\text{emission, } B} \underbrace{P(t_i \mid t_{i-1})}_{\text{transition, } A}$$

How many possible choices?

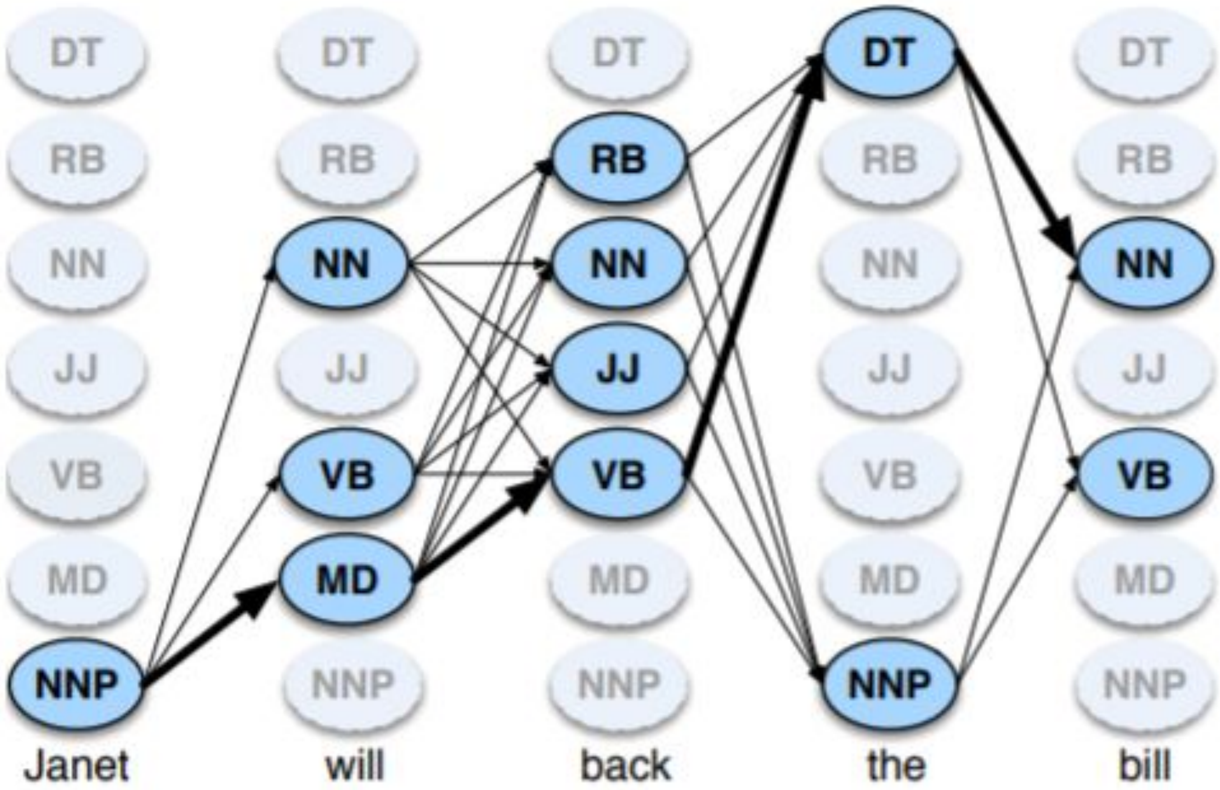
Part of speech tagging example

	I	suspect	the	present	forecast	is	pessimistic	.
noun	•	•	•	•	•	•		
adj.		•		•	•		•	
adv.				•				
verb		•		•	•	•		
num.	•							
det.			•					
punc.								•

With this very simple tag set, $7^8 = 5.7$ million labelings.
(Even restricting to the possibilities above, 288 labelings.)

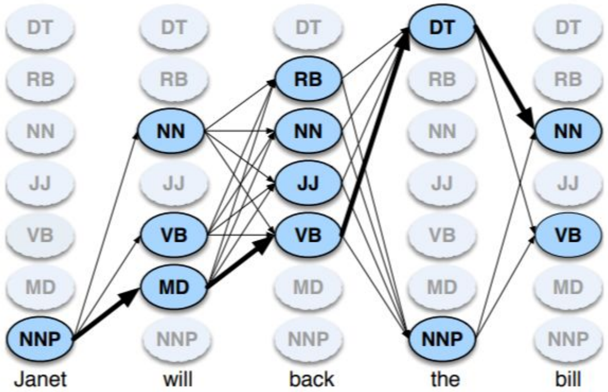
The Viterbi algorithm

The Viterbi algorithm



The Viterbi algorithm

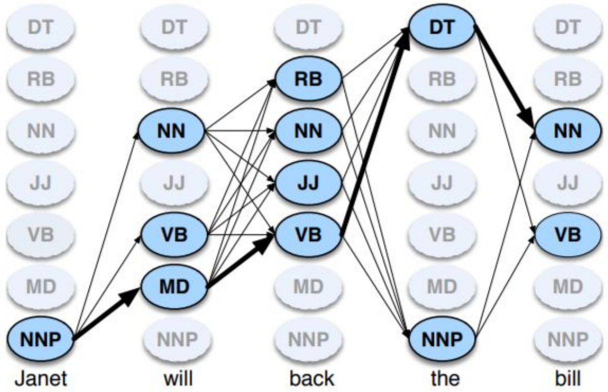
$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j



$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

The Viterbi algorithm

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

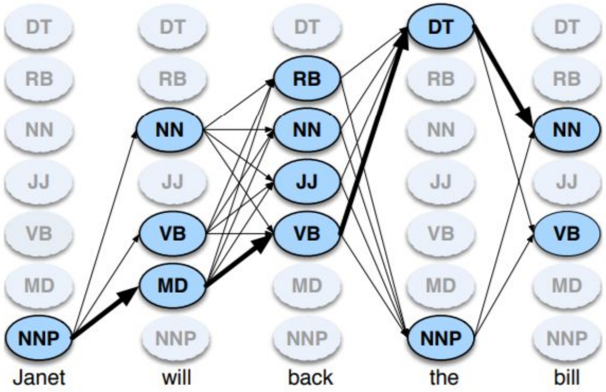


$$v_t(j) = \max_{i=1}^{N} \boxed{v_{t-1}(i)}\, a_{ij}\, b_j(o_t)$$

previous
Viterbi path
probability

The Viterbi algorithm

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j



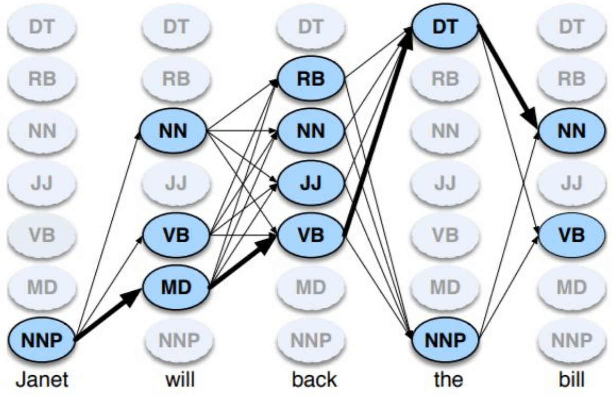
transition probability

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

previous Viterbi path probability

The Viterbi algorithm

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j



transition probability

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

state observation likelihood

previous Viterbi path probability

The 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**

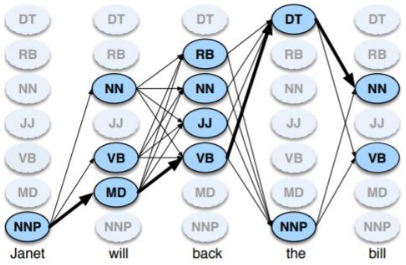
$$viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$$
$$backpointer[s,1] \leftarrow 0$$

for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

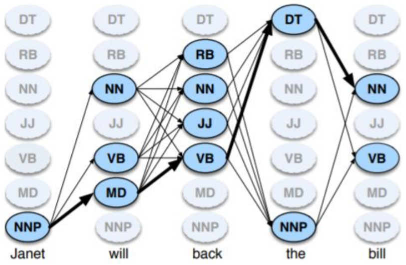
$$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$$
$$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$$

bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*



The 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**
$\quad viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
$\quad backpointer[s,1] \leftarrow 0$

initialization

for each time step t **from** 2 **to** T **do**
for each state s **from** 1 **to** N **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} \; viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} \; viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

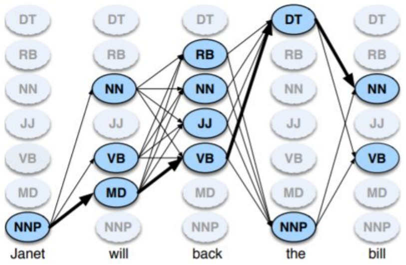
$$bestpathprob \leftarrow \max_{s=1}^{N} \; viterbi[s,T]$$

$$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} \; viterbi[s,T]$$

bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

The 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

$\quad viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$

$\quad backpointer[s,1] \leftarrow 0$

for each time step *t* **from** 2 **to** *T* **do**

recursion

$\quad$ **for** each state *s* **from** 1 **to** *N* **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$$

$$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$$

bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time

return *bestpath*, *bestpathprob*

The Viterbi algorithm

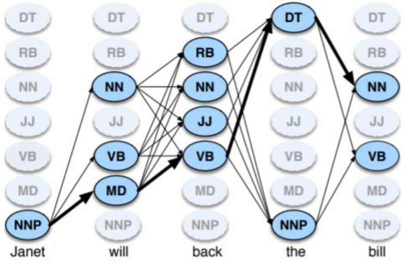
function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

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

initialization

recursion

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$



The Viterbi algorithm

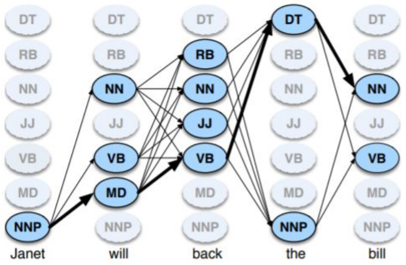
function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do**
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$ ← $v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

initialization

recursion

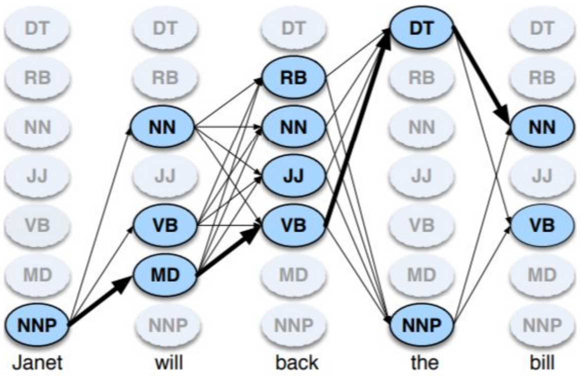
termination



The Viterbi algorithm

	NNP	MD	VB	JJ	NN	RB	DT
$<s>$	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

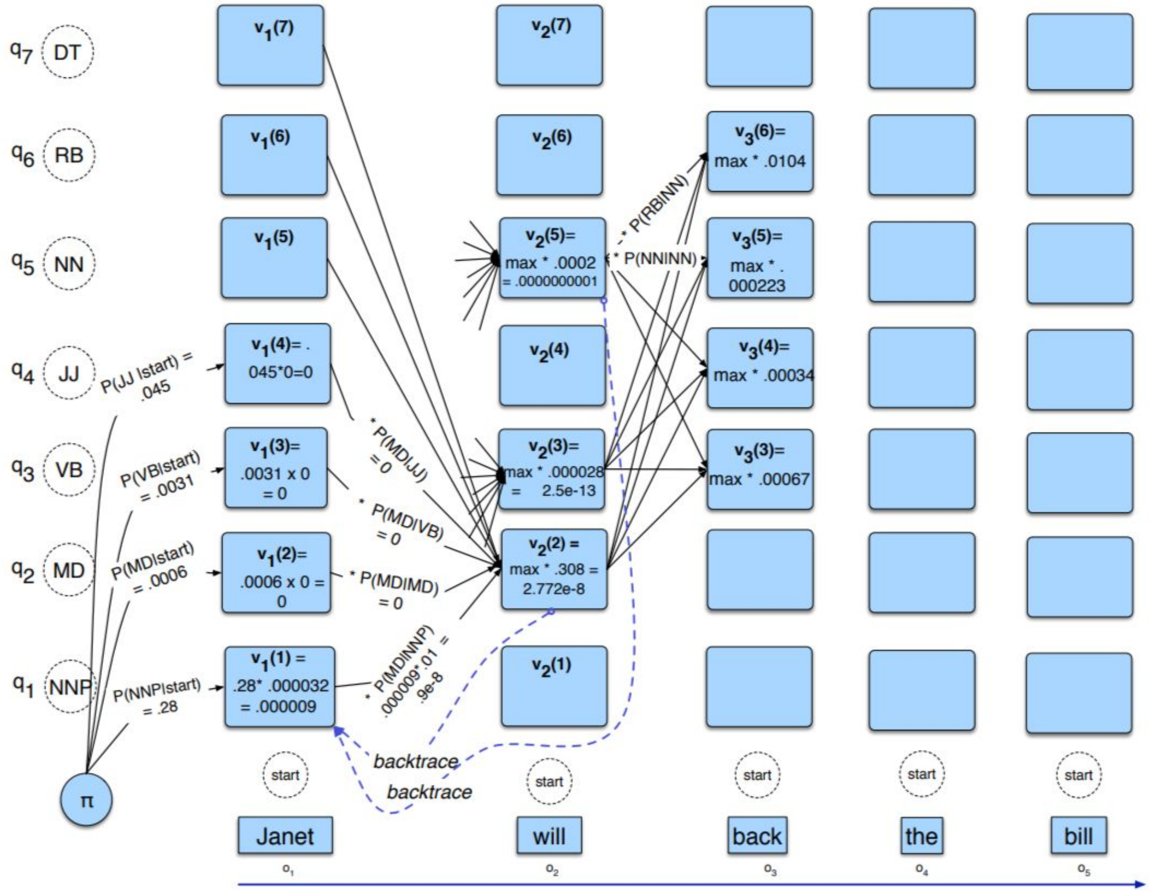
	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0



A

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

B


q7 DT
q6 RB
q5 NN
q4 JJ
q3 VB
q2 MD
q1 NNP
π
P(JJ |start) = .045
P(VB|start) = .0031
P(MD|start) = .0006
P(NNP|start) = .28
v1(7)
v1(6)
v1(5)
v1(4)= . 045*0=0
v1(3)= .0031 x 0 = 0
v1(2)= .0006 x 0 = 0
v1(1) = .28* .000032 = .000009
* P(MD|JJ) = 0
* P(MD|VB) = 0
* P(MD|MD) = 0
* P(MD|NNP) .000009*.01 = .9e-8
v2(7)
v2(6)
v2(5)= max * .0002 = .0000000001
v2(4)
v2(3)= max * .000028 = 2.5e-13
v2(2) = max * .308 = 2.772e-8
v2(1)
* P(RB|NN)
* P(NN|NN)
v3(6)= max * .0104
v3(5)= max * . 000223
v3(4)= max * .00034
v3(3)= max * .00067
backtrace
backtrace
start
start
start
start
start
Janet
will
back
the
bill
o1
o2
o3
o4
o5

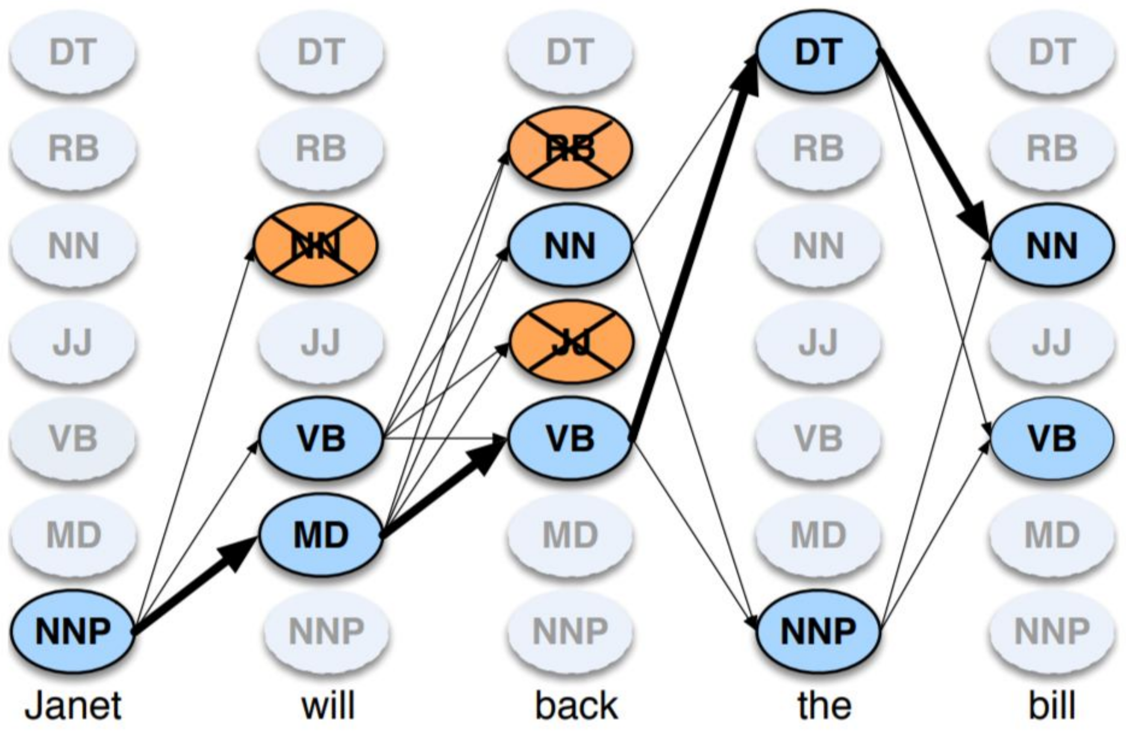
The Viterbi algorithm

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

Computational complexity in N and T?

Beam search



HMMs: algorithms

Forward	**Problem 1 (Likelihood):**	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Viterbi	**Problem 2 (Decoding):**	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Forward-backward; Baum-Welch	**Problem 3 (Learning):**	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

From J&M

HMMs: algorithms

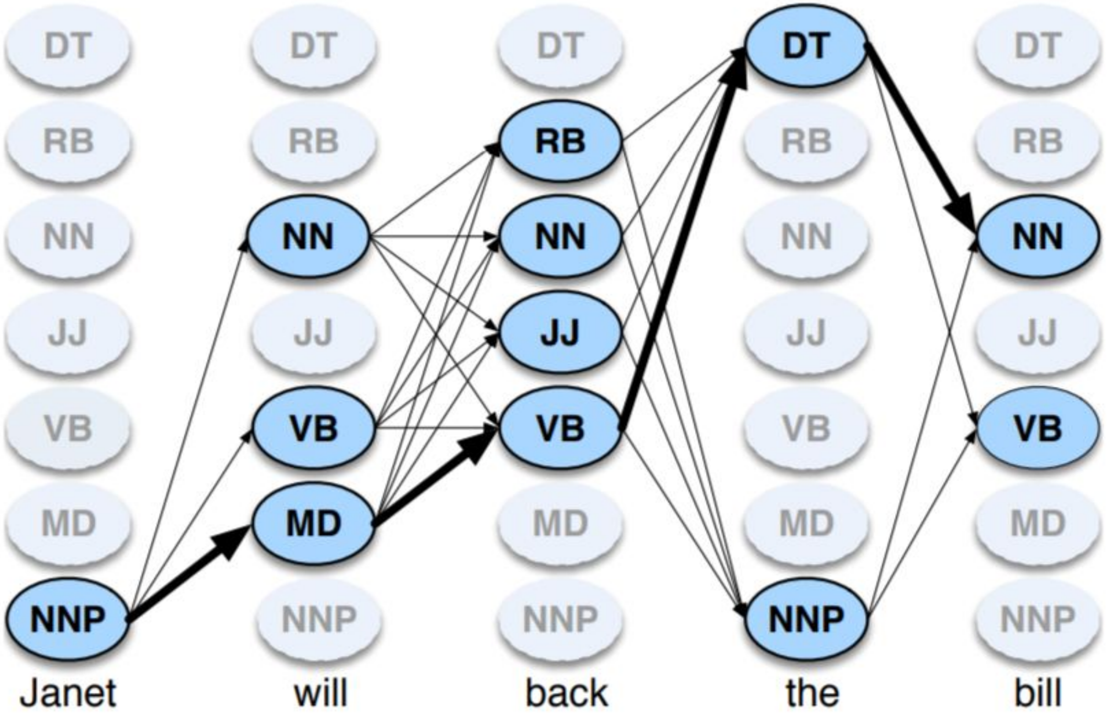
Forward	**Problem 1 (Likelihood):**	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Viterbi	**Problem 2 (Decoding):**	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Forward-backward; Baum-Welch	**Problem 3 (Learning):**	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

From J&M

The Forward algorithm

- Just sum instead of max!

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$



Viterbi

- n-best decoding
- relationship to sequence alignment

Citation	Field
Viterbi (1967)	information theory
Vintsyuk (1968)	speech processing
Needleman and Wunsch (1970)	molecular biology
Sakoe and Chiba (1971)	speech processing
Sankoff (1972)	molecular biology
Reichert et al. (1973)	molecular biology
Wagner and Fischer (1974)	computer science