

AI6122 Text Data Management & Analysis

Topic: POS and HMM



Topics

- Word classes
- Part of speech tagging
- Use HMMs for POS tagging



Word Classes: Parts of Speech

- Parts of speech (POS)
 - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc.
 - Also Called: parts-of-speech, lexical categories, word classes, morphological classes, lexical tags...

N	noun	chair, bandwidth, pacing
V	verb	study, debate, munch
ADJ	adjective	purple, tall, ridiculous
ADV	adverb	unfortunately, slowly
P	preposition	of, by, to
PRO	pronoun	I, me, mine
DET	determiner	the, a, that, those

POS Tagging

- The process of assigning a part-of-speech or lexical class marker to each word in a sequence.
 - First step of a vast number of practical tasks
- Information extraction
 - Finding names, e.g., people, organization -- N.
- Machine Translation
 - E.g. result/N, result/V -> kyol-kwa/N, kyol-kwa-lul-ne-da/V
- Parsing
 - Helpful to know parts of speech before you start parsing, e.g. subject-verb-object

WORD	tag
the	DET
koala	N
put	V
the	DET
keys	N
on	P
the	DET
table	N

POS Tagging: Choosing a Tagset

- To do POS tagging, we need to choose a standard **set of tags**
 - Could pick very coarse tagsets (e.g., N, V, Adj, Adv.)
 - More commonly used set is the finer grained, “Penn TreeBank tagset”, 45 tags

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	<i>and, but, or</i>	SYM	symbol	<i>+, %, &</i>
CD	cardinal number	<i>one, two, three</i>	TO	“to”	<i>to</i>
DT	determiner	<i>a, the</i>	UH	interjection	<i>ah, oops</i>
EX	existential ‘there’	<i>there</i>	VB	verb, base form	<i>eat</i>
FW	foreign word	<i>mea culpa</i>	VBD	verb, past tense	<i>ate</i>
IN	preposition/sub-conj	<i>of, in, by</i>	VBG	verb, gerund	<i>eating</i>
JJ	adjective	<i>yellow</i>	VBN	verb, past participle	<i>eaten</i>
JJR	adj., comparative	<i>bigger</i>	VBP	verb, non-3sg pres	<i>eat</i>
JJS	adj., superlative	<i>wildest</i>	VBZ	verb, 3sg pres	<i>eats</i>
LS	list item marker	<i>1, 2, One</i>	WDT	wh-determiner	<i>which, that</i>
MD	modal	<i>can, should</i>	WP	wh-pronoun	<i>what, who</i>
NN	noun, sing. or mass	<i>llama</i>	WP\$	possessive wh-	<i>whose</i>
NNS	noun, plural	<i>llamas</i>	WRB	wh-adverb	<i>how, where</i>
NNP	proper noun, singular	<i>IBM</i>	\$	dollar sign	<i>\$</i>
NNPS	proper noun, plural	<i>Carolinas</i>	#	pound sign	<i>#</i>
PDT	predeterminer	<i>all, both</i>	“	left quote	<i>‘ or “</i>
POS	possessive ending	<i>’s</i>	”	right quote	<i>’ or ”</i>
PRP	personal pronoun	<i>I, you, he</i>	(left parenthesis	<i>[, (, {, <</i>
PRP\$	possessive pronoun	<i>your, one’s</i>)	right parenthesis	<i>],), }, ></i>
RB	adverb	<i>quickly, never</i>	,	comma	<i>,</i>
RBR	adverb, comparative	<i>faster</i>	.	sentence-final punc	<i>. ! ?</i>
RBS	adverb, superlative	<i>fastest</i>	:	mid-sentence punc	<i>: ; ... - -</i>
RP	particle	<i>up, off</i>			

Using the Penn Tagset

- The/**DT** grand/**JJ** jury/**NN** commented/**VBD** on/**IN** a/**DT** number/**NN** of/**IN** other/**JJ** topics/**NNS** ./.
- Prepositions and subordinating conjunctions marked IN (“although/**IN** I/**PRP**..”)
- Except the preposition “to” is just marked “**TO**”.

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	<i>and, but, or</i>	SYM	symbol	<i>+, %, &</i>
CD	cardinal number	<i>one, two, three</i>	TO	“to”	<i>to</i>
DT	determiner	<i>a, the</i>	UH	interjection	<i>ah, oops</i>
EX	existential ‘there’	<i>there</i>	VB	verb, base form	<i>eat</i>
FW	foreign word	<i>mea culpa</i>	VBD	verb, past tense	<i>ate</i>
IN	preposition/sub-conj	<i>of, in, by</i>	VBG	verb, gerund	<i>eating</i>
JJ	adjective	<i>yellow</i>	VCN	verb, past participle	<i>eaten</i>
JJR	adj., comparative	<i>bigger</i>	VBP	verb, non-3sg pres	<i>eat</i>
JJS	adj., superlative	<i>wildest</i>	VBZ	verb, 3sg pres	<i>eats</i>
LS	list item marker	<i>1, 2, One</i>	WDT	wh-determiner	<i>which, that</i>
MD	modal	<i>can, should</i>	WP	wh-pronoun	<i>what, who</i>
NN	noun, sing. or mass	<i>llama</i>	WP\$	possessive wh-	<i>whose</i>
NNS	noun, plural	<i>llamas</i>	WRB	wh-adverb	<i>how, where</i>
NNP	proper noun, singular	<i>IBM</i>	\$	dollar sign	<i>\$</i>
NNPS	proper noun, plural	<i>Carolinas</i>	#	pound sign	<i>#</i>
PDT	predeterminer	<i>all, both</i>	“	left quote	<i>‘ or “</i>
POS	possessive ending	<i>’s</i>	”	right quote	<i>’ or ”</i>
PRP	personal pronoun	<i>I, you, he</i>	(left parenthesis	<i>[, (, {, <</i>
PRP\$	possessive pronoun	<i>your, one’s</i>)	right parenthesis	<i>],), }, ></i>
RB	adverb	<i>quickly, never</i>	,	comma	<i>,</i>
RBR	adverb, comparative	<i>faster</i>	.	sentence-final punc	<i>. ! ?</i>
RBS	adverb, superlative	<i>fastest</i>	:	mid-sentence punc	<i>: ; ... --</i>
RP	particle	<i>up, off</i>			

POS Tagging

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



How Hard is POS Tagging? Measuring Ambiguity

- Number of word types with different levels of POS ambiguity from the Brown corpus

Many **ambiguous** words appear **frequently** in corpus

	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 (<i>well, beat</i>)	32
7 tags	2 (<i>still, down</i>)	6 (<i>well, set, round, open, fit, down</i>)
8 tags		4 (<i>'s, half, back, a</i>)
9 tags		3 (<i>that, more, in</i>)

POS Tagging as Sequence Classification

- **Input:** We are given a sentence (an “observation” or “sequence of observations”)
- **Output:** What is the best sequence of tags that corresponds to this sequence of observations?

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

- Probabilistic view:
 - Consider all possible sequences of tags
 - Out of this universe of sequences, choose **the tag sequence** which is **most probable** given the observation sequence of n words $w_1 \dots w_n$.


Road to HMMs

- Out of all sequences of n tags $t_1 \dots t_n$ the single tag sequence such that $P(t_1 \dots t_n | w_1 \dots w_n)$ is highest.


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

- Hat $\hat{}$ means “our estimate of the best one”
 - $\arg \max_x f(x)$ means “the x such that $f(x)$ is maximized”
-
- But how to compute this value?
 - Bayesian inference: Use Bayes rule to transform this equation into a set of other probabilities that are easier to compute

Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$


$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n) \quad \longrightarrow \quad \hat{t}_1^n = \operatorname{argmax}_{t_1^n} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$$


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

Likelihood and Prior

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} \overbrace{P(w_1^n | t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

The **/DT** yellow **/JJ** hat **/NN**

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

The probability of a word appearing depends only on **its own POS tag** $P(\text{the} | \text{DT})$

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

The probability of a tag appearing depends only **on the previous tag** $P(\text{NN} | \text{JJ})$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n) \approx \operatorname{argmax}_{t_1^n} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$

Two Kinds of Probabilities

- Tag transition probabilities $p(t_i|t_{i-1})$

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

- Example: determiners likely to precede adjectives and nouns
 - That/DT flight/NN
 - The/DT yellow/JJ hat/NN
- So we expect $P(NN|DT)$ and $P(JJ|DT)$ to be high, but not $P(DT|JJ)$ to be high
- Compute $P(NN|DT)$ by counting in a labeled corpus:

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Two Kinds of Probabilities

- Word likelihood probabilities $p(w_i|t_i)$

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

- Example: VBZ (3rd person singular present verb) likely to be “is”
- Compute $P(is|VBZ)$ by counting in a labeled corpus:

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

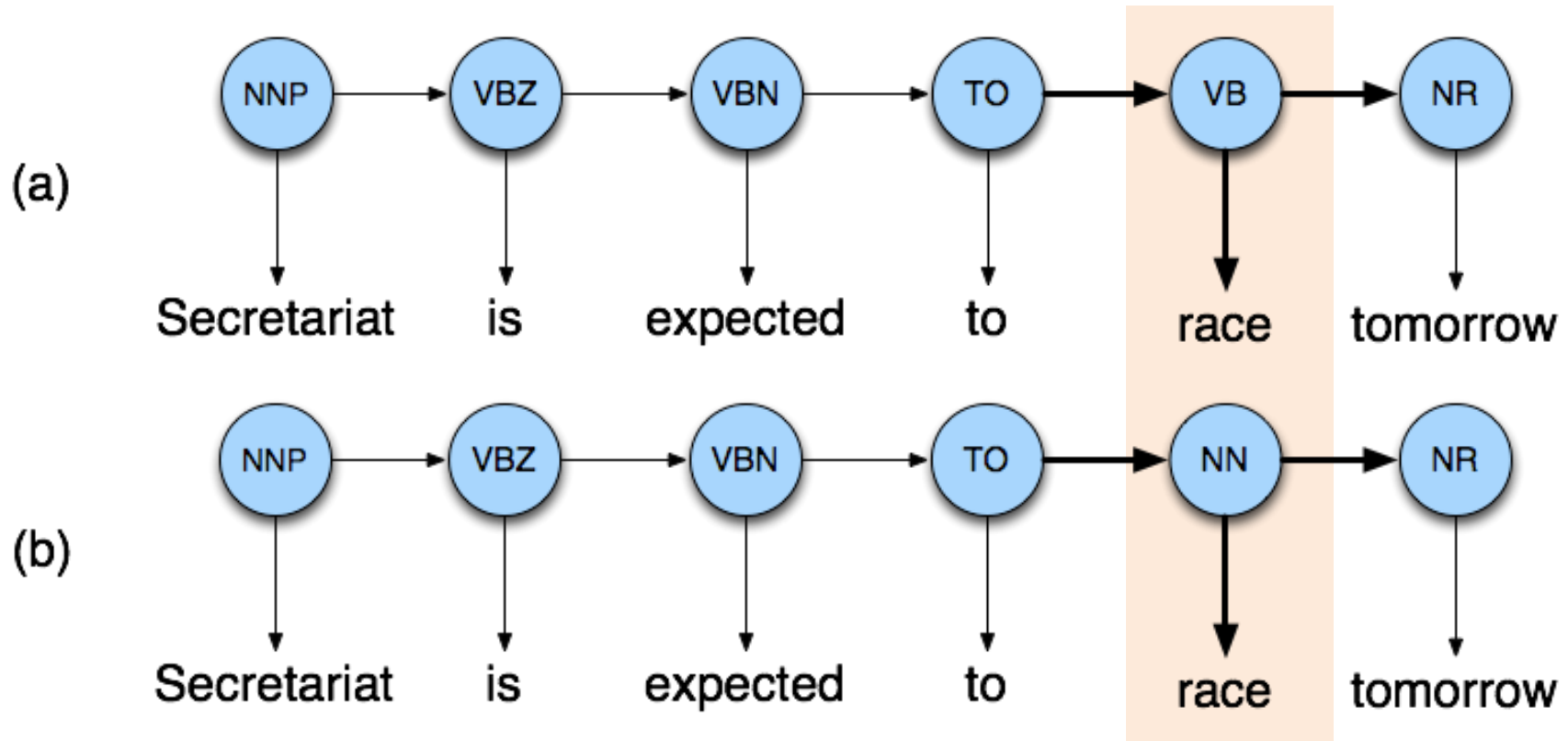
Example: The Verb “race”

- Secretariat/**NNP** is/**VBZ** expected/**VBN** to/**TO** **race**/**VB** tomorrow/**NR**
- People/**NNS** continue/**VB** to/**TO** inquire/**VB** the/**DT** reason/**NN** for/**IN** the/**DT** **race**/**NN** for/**IN** outer/**JJ** space/**NN**
- How do we pick the right tag for word “race” in each sentence?



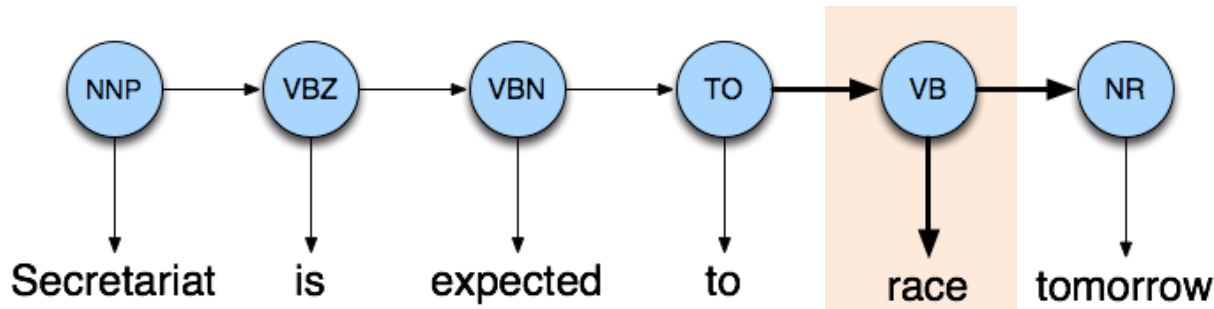
Disambiguating “race”

- Assuming tags of other words are known (for this example)



Calculating estimated probability

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n) \approx \operatorname{argmax}_{t_1^n} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$



$p(\text{Secretariat} | \text{NNP}) * p(\text{NNP} | \text{Start})$
 $* p(\text{is} | \text{VBZ}) * p(\text{VBZ} | \text{NNP})$
 $* p(\text{expected} | \text{VBN}) * p(\text{VBN} | \text{VBZ})$
 $* p(\text{to} | \text{TO}) * p(\text{TO} | \text{VBN})$
 $* p(\text{race} | \text{VB}) * p(\text{VB} | \text{TO})$
 $* p(\text{tomorrow} | \text{NR}) * p(\text{NR} | \text{VB})$

Example

- From a training corpus, we know
 - $P(NN|TO) = .00047$ $P(VB|TO) = .83$ $P(race|NN) = .00057$
 - $P(race|VB) = .00012$ $P(NR|VB) = .0027$ $P(NR|NN) = .0012$
- How to use the above information to do POS tagging?
 - $P(VB|TO)P(NR|VB)P(race|VB) = .00000027$
 - $P(NN|TO)P(NR|NN)P(race|NN) = .00000000032$
- So we (correctly) choose verb for race in this sentence

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n) \approx \operatorname{argmax}_{t_1^n} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$

Summary

- Parts of speech, Tagsets, and tagging
- Next: HMM Tagging
 - Hidden Markov Models
 - Viterbi decoding
- The next two slides are about linguistics and are for your references



Open and Closed Classes

Skip

- Closed class: a small(ish) fixed membership
 - Usually **function words** (short common words which play a role in grammar)
 - prepositions: *on, under, over, ...*
 - particles: *up, down, on, off, ...*
 - determiners: *a, an, the, ...*
 - pronouns: *she, who, I, ..*
 - conjunctions: *and, but, or, ...*
 - auxiliary verbs: *can, may should, ...*
 - numerals: *one, two, three, third, ...*
- Open class: new ones can be created all the time
 - English has 4: Nouns, Verbs, Adjectives, Adverbs
 - Many languages have these 4, but not all!
 - Nouns are typically where the bulk of the action is with respect to new items



Open Class Words

Skip

- Nouns
 - Proper nouns (Boulder, Granby, Beyoncé) -- 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 things
 - **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)
 - With differing patterns of regularity



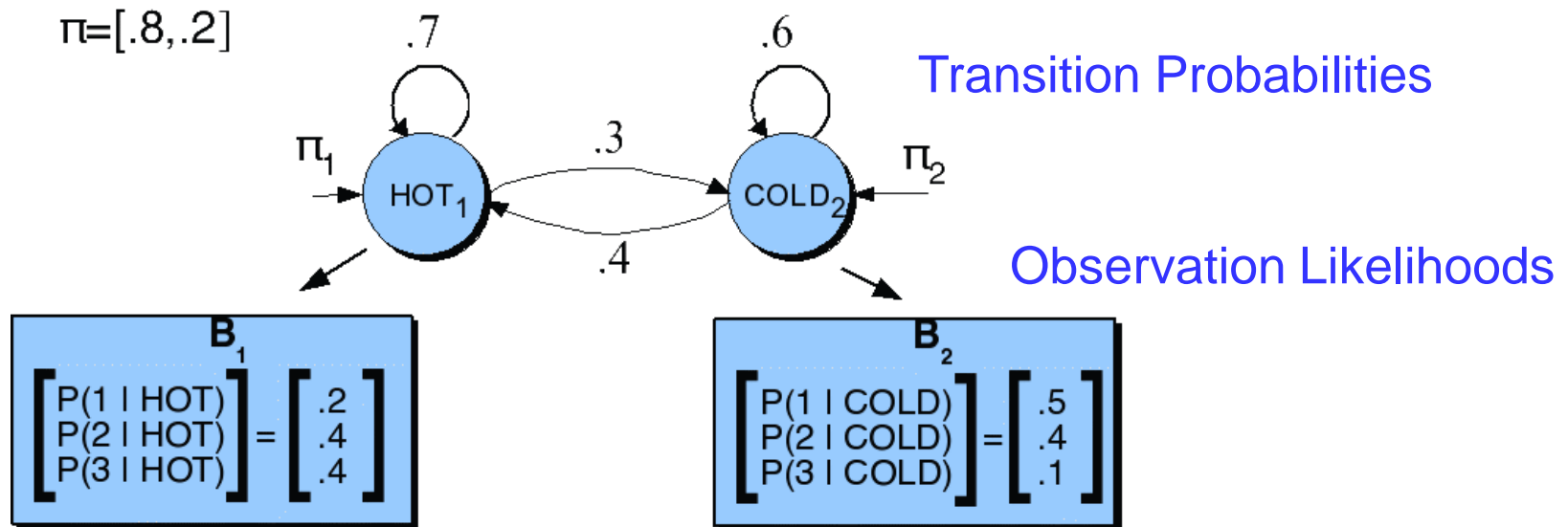
HMM for Ice Cream

- You are a climatologist in the year 2799 studying global warming
- You can't find any records of the weather in Singapore for summer of 2018
- But you find your grandma's diary which lists how many ice-creams she ate every date that summer
- Your job: figure out whether each day was cold/hot



Example of sequence prediction

- Can the number of ice cream eaten be used to **predict** the weather?
 - Ice cream observation sequence: 2,1,3,2,2...
 - Weather Sequence: H,C,H,H,C...



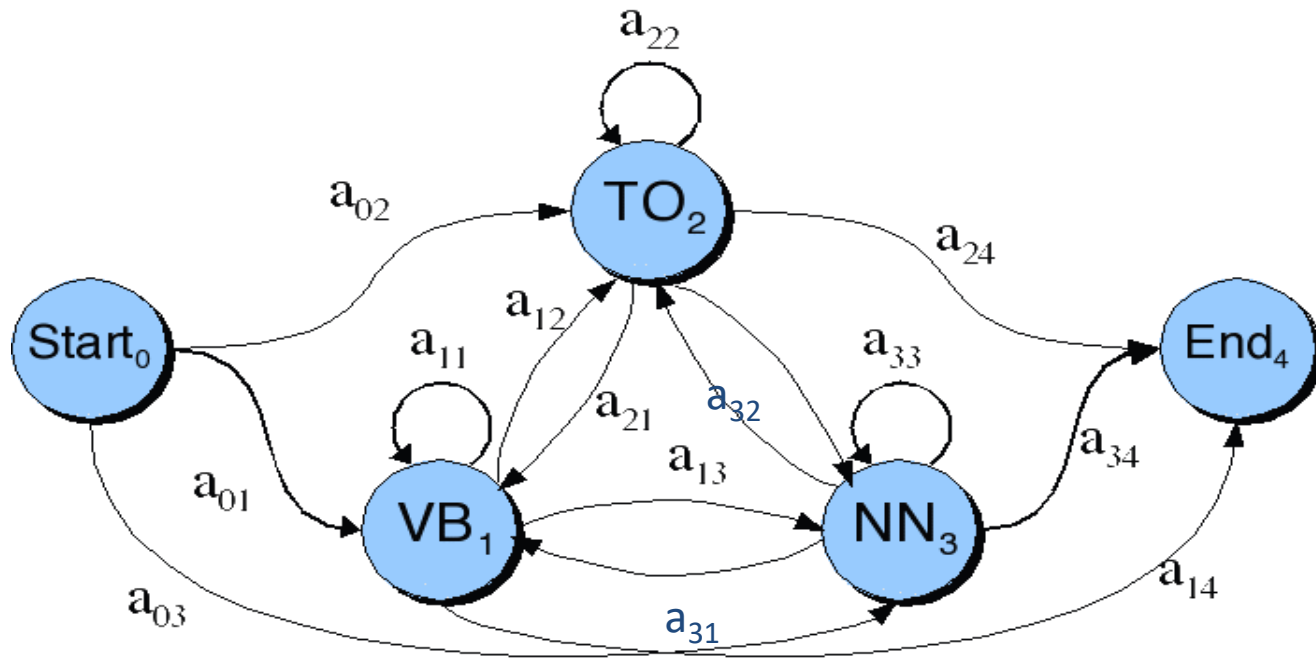
Hidden Markov Models

- What we've described with these two kinds of probabilities is a Hidden Markov Model (HMM)
 - Transition Probabilities
 - Observation Likelihoods
- Formalizing HMM:
 - A **weighted finite-state automaton** where each arc is associated with a probability
 - The probability indicates how likely a path is to be taken

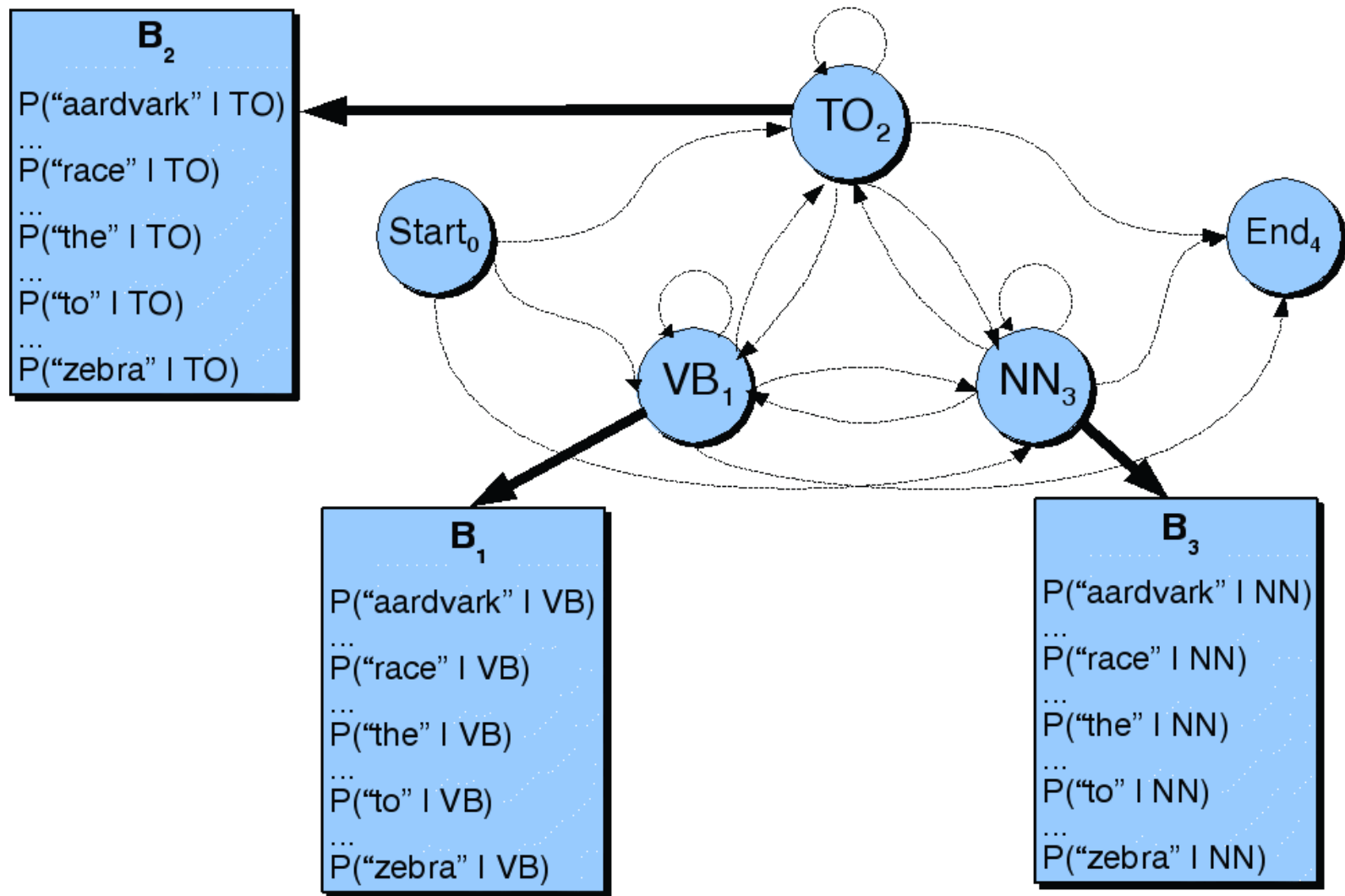


Transition Probabilities

- The sum of the probabilities leaving any arc must sum to one
 - For example, $a_{01} + a_{02} + a_{03} = 1$

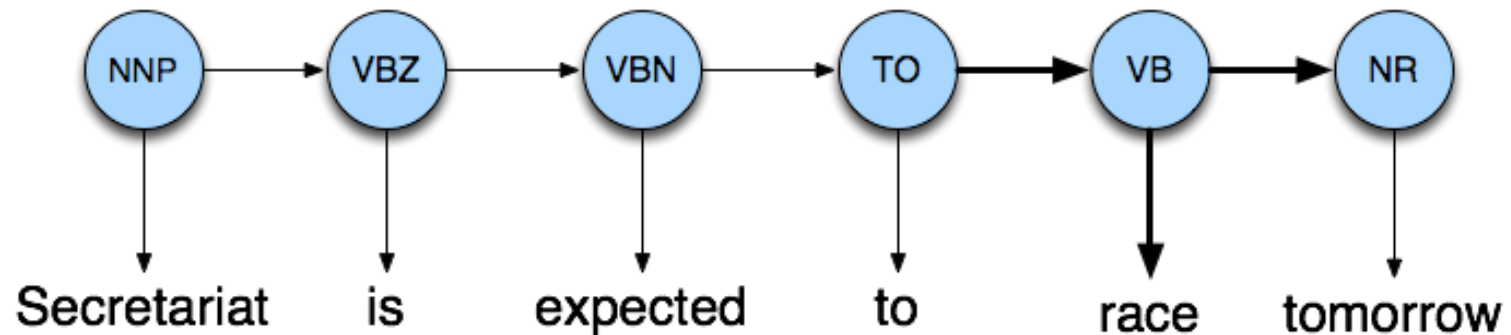


Observation Likelihoods



Hidden Markov Model

- In part-of-speech tagging
 - The input symbols are **words**
 - But the hidden states are **part-of-speech tags**



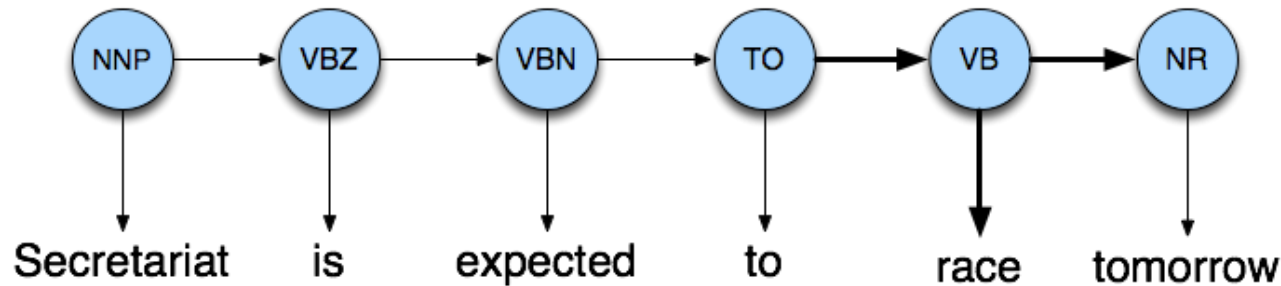
- It has many other applications
 - Named entity recognition, gene prediction, etc

Hidden Markov Models

- States $Q = q_1, q_2 \dots q_N$; and the start and end states q_0, q_F
- Observations $O = o_1, o_2 \dots o_T$;
 - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, \dots v_V\}$
 - s_i : the state of the i -th observation;
 - q_0, q_F are not associated with observations
- Transition probabilities: Transition probability matrix $A = \{a_{ij}\}$;
 - $a_{ij} = P(s_t = j | s_{t-1} = i) \quad 1 \leq i, j \leq N$
- Observation likelihoods: Output probability matrix $B = \{b_i(k)\}$;
 - $b_i(k) = P(X_t = o_k | s_t = i)$
- Special initial probability vector π ;
 - $\pi_i = P(s_1 = i) \quad 1 \leq i \leq N$

Hidden Markov Model

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n) \approx \operatorname{argmax}_{t_1^n} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$



$p(\text{Secretariat} | \text{NNP}) * p(\text{NNP} | \text{Start})$
 $* p(\text{is} | \text{VBZ}) * p(\text{VBZ} | \text{NNP})$
 $* p(\text{expected} | \text{VBN}) * p(\text{VBN} | \text{VBZ})$
 $* p(\text{to} | \text{TO}) * p(\text{TO} | \text{VBN})$
 $* p(\text{race} | \text{VB}) * p(\text{VB} | \text{TO})$
 $* p(\text{tomorrow} | \text{NR}) * p(\text{NR} | \text{VB})$

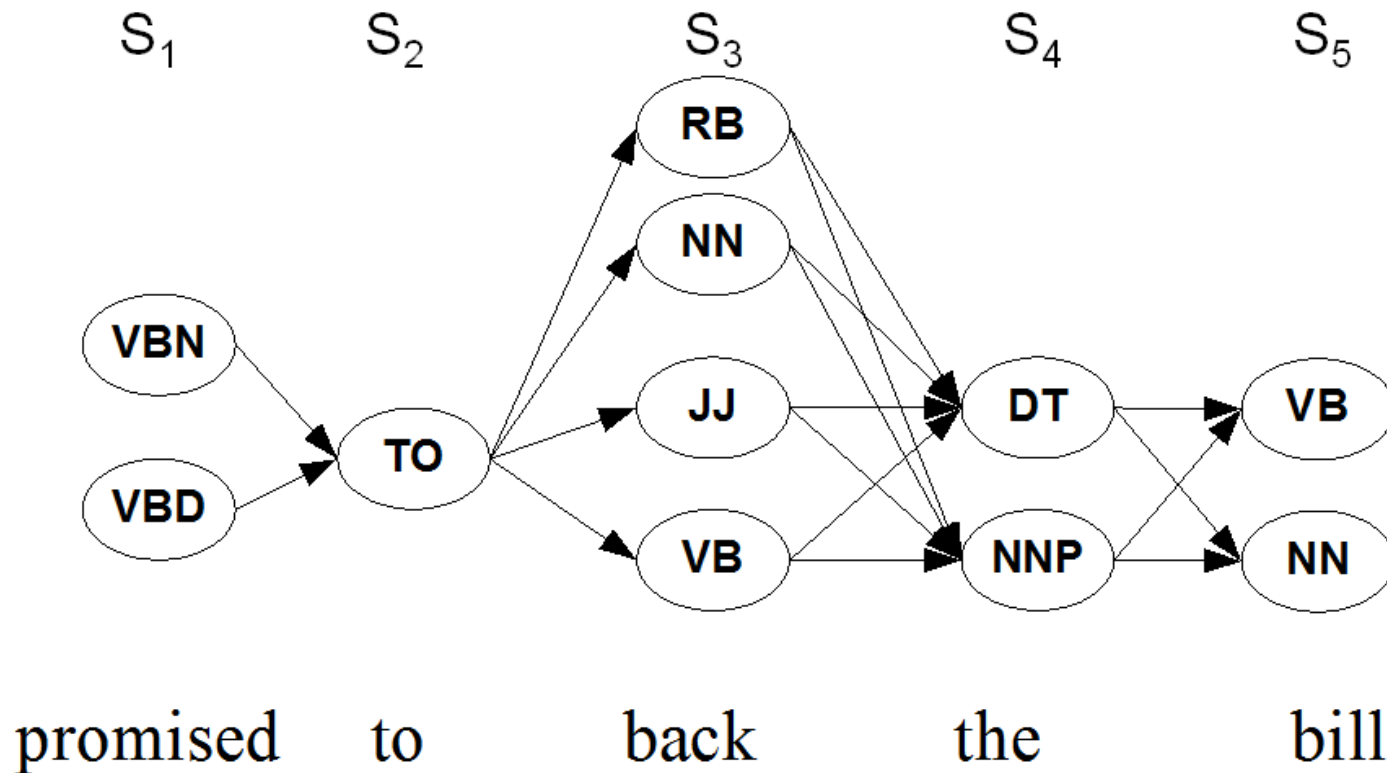
Decoding

- Now we have a complete model and we need to get

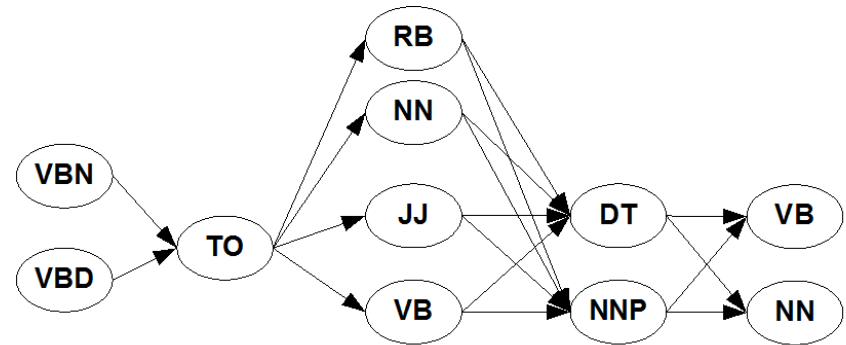
$$\hat{t}_1^n = \arg \max_{t_1^n} P(t_1^n | w_1^n)$$

- **Determine** sequences of variables, **given** sequence of observations
- We could just enumerate all paths given the input and use the model to assign probabilities to each.
 - Not a good idea. 1 2 -- HH, HC, CC, CH
 - N^T : N (number of states) T (size of sequence)
 - Dynamic programming helps us here

Example sentence

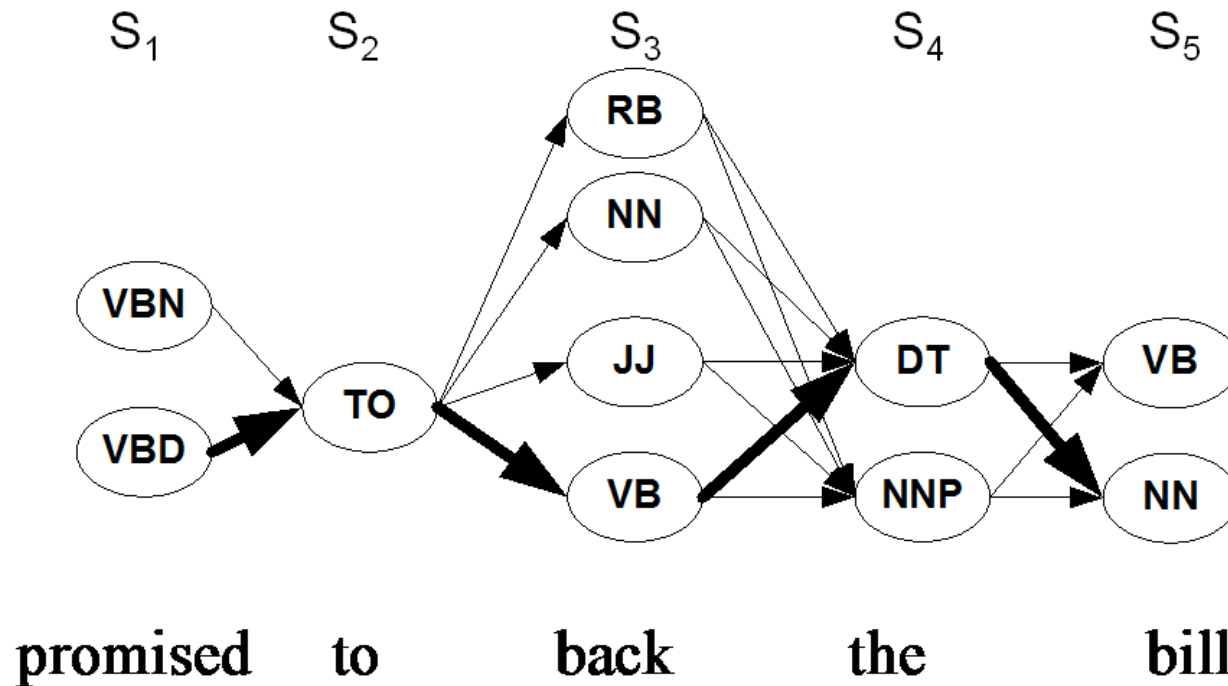


Enumerate all paths

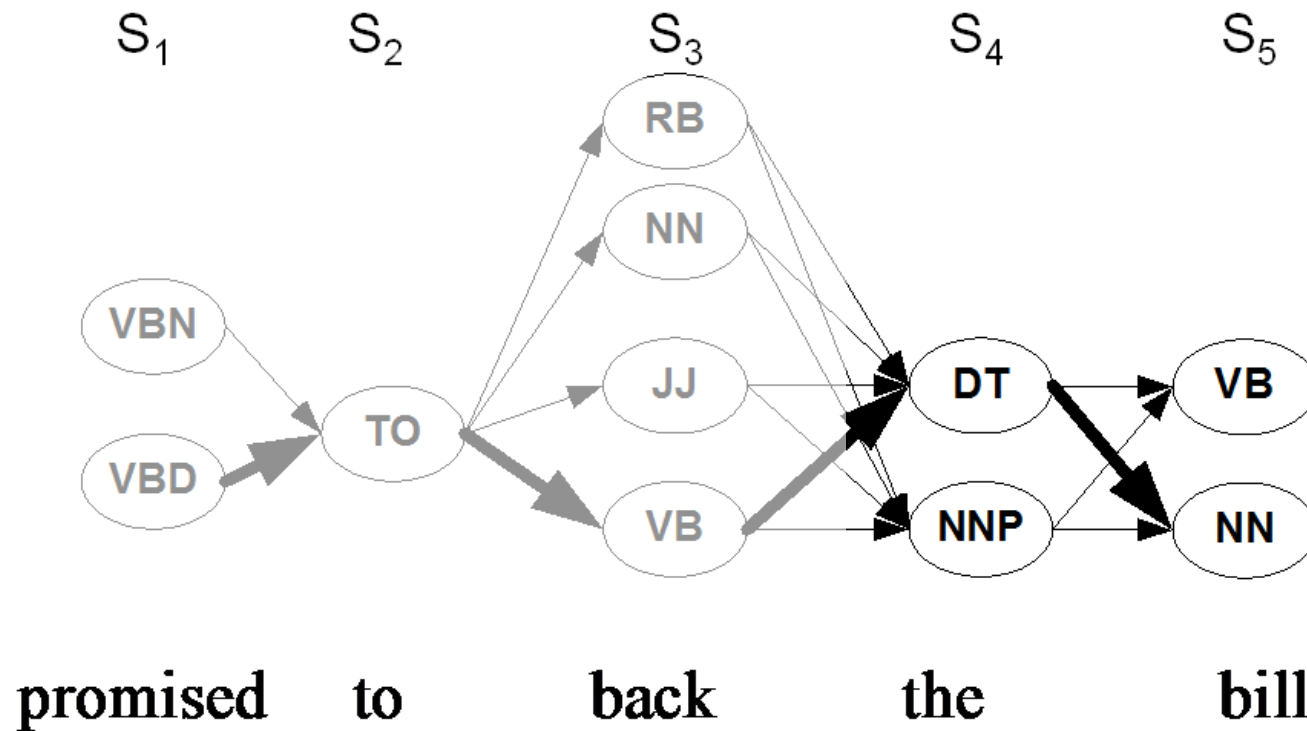


- VBN TO RB DT VB
- VBN TO RB DT NN
- VBN TO RB NNP VB
- VBN TO RB NNP NN
- VBN TO NN DT VB
- VBN TO NN DT NN
- VBN TO NN NNP VB
- VBN TO NN NNP NN
- VBN TO JJ DT VB
- VBN TO JJ DT NN
- VBN TO JJ NNP VB
- VBN TO JJ NNP NN
- VBN TO VB DT VB
- VBN TO VB DT NN
- VBN TO VB NNP VB
- VBN TO VB NNP NN
- VBD TO RB DT VB
- VBD TO RB DT NN
- VBD TO RB NNP VB
- VBD TO RB NNP NN
- VBD TO NN DT VB
- VBD TO NN DT NN
- VBD TO NN NNP VB
- VBD TO NN NNP NN
- VBD TO JJ DT VB
- VBD TO JJ DT NN
- VBD TO JJ NNP VB
- VBD TO JJ NNP NN
- VBD TO VB DT VB
- VBD TO VB DT NN
- VBD TO VB NNP VB
- VBD TO VB NNP NN

The best choice?



From DT to NN?



Intuition

- Consider a state sequence (tag sequence) that ends at time t with a particular tag i .
- The probability of that tag sequence can be broken into two parts
 - The probability of the **BEST** tag sequence up through $t - 1$
 - Multiplied **by the transition probability** from the tag at the end of the $t - 1$ sequence to i , and the observation probability of the word given tag i .
- Let j be the tag at the end of the $t - 1$ sequence, and W be the word at time t

$$\underset{v_t[i]}{Viterbi[i, t]} = Viterbi[j, t - 1] \times \underset{a_{j,i}}{p(i|j)} \times \underset{b_i(W)}{p(W|i)}$$

Consider paths ending with bill:NN

From S4, we have two paths P1, P2 to reach NN

- $p_1 = v_4[DT] * p(NN|DT) * p(bill|NN)$
- $p_2 = v_4[NNP] * p(NN|NNP) * p(bill|NN)$
- $v_5[NN] = \max(p_1, p_2)$
- $$v_4[DT] = \max(v_3[RB] * p(DT|RB) * p(the|DT),$$

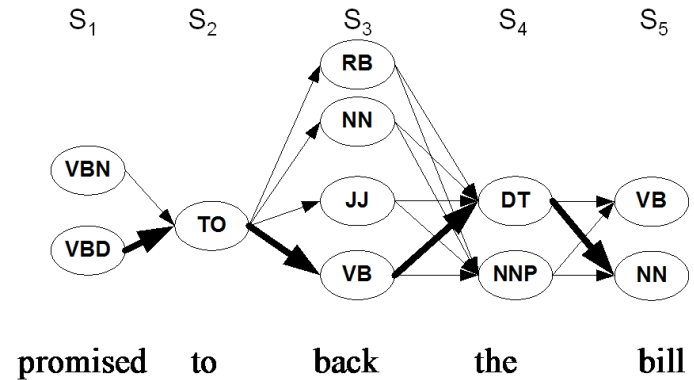
$$v_3[NN] * p(DT|NN) * p(the|DT),$$

$$v_3[JJ] * p(DT|JJ) * p(the|DT),$$

$$v_3[VB] * p(DT|VB) * p(the|DT))$$

$$= \max(v_3[RB] * p(DT|RB), v_3[NN] * p(DT|NN), v_3[JJ]$$

$$* p(DT|JJ), v_3[VB] * p(DT|VB)) * p(the|DT)$$

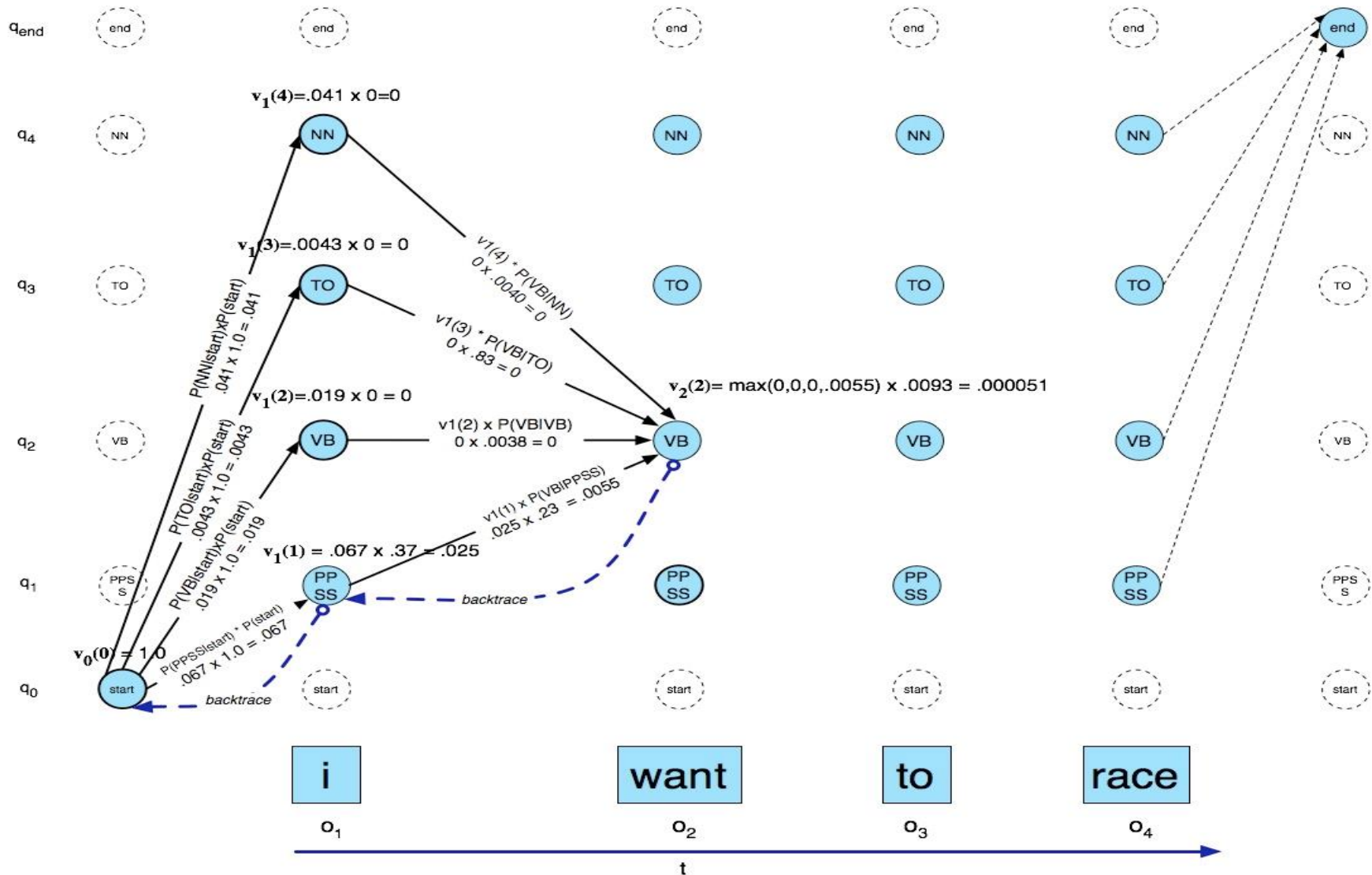


Main Idea

- We also have a matrix.
 - Each column– a time ' t ' (observation)
 - Each row – a state ' i '
 - For each cell $v_t[i]$, we compute the probability of the **best** path to the cell
- Viterbi path probability at time t for state i
 - there are $|Q|$ number of paths from $t - 1$ to $v_t[i]$
 - if we know the best path to each cell in $t - 1$ ($v_{t-1}[j]$)

$$\arg \max_j v_{t-1}[j] \times P(i|j) \times P(s_t|i)$$

Viterbi Example: Variable $v_t[i]$ the Viterbi path probability at time t for state i



Example continue from previous slide

- $v_2[NN] = \max(v_1[NN] * p(NN|NN), v_1[TO] * p(NN|TO), v_1[VB] * p(NN|VB), v_1[PPSS] * p(NN|PPSS)) * p(want|NN)$
- $= \max(0 * 0.087, 0 * 0.00047, 0 * 0.047, 0.025 * 0.0012) * 0.000054$

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

The Viterbi Algorithm

T : word N : tag

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

create a path probability matrix $viterbi[N+2, T]$

for each state s **from** 1 **to** N **do** ; initialization step

$viterbi[s, 1] \leftarrow a_{0,s} * b_s(o_1)$

$backpointer[s, 1] \leftarrow 0$

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

for each state s **from** 1 **to** N **do**

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

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

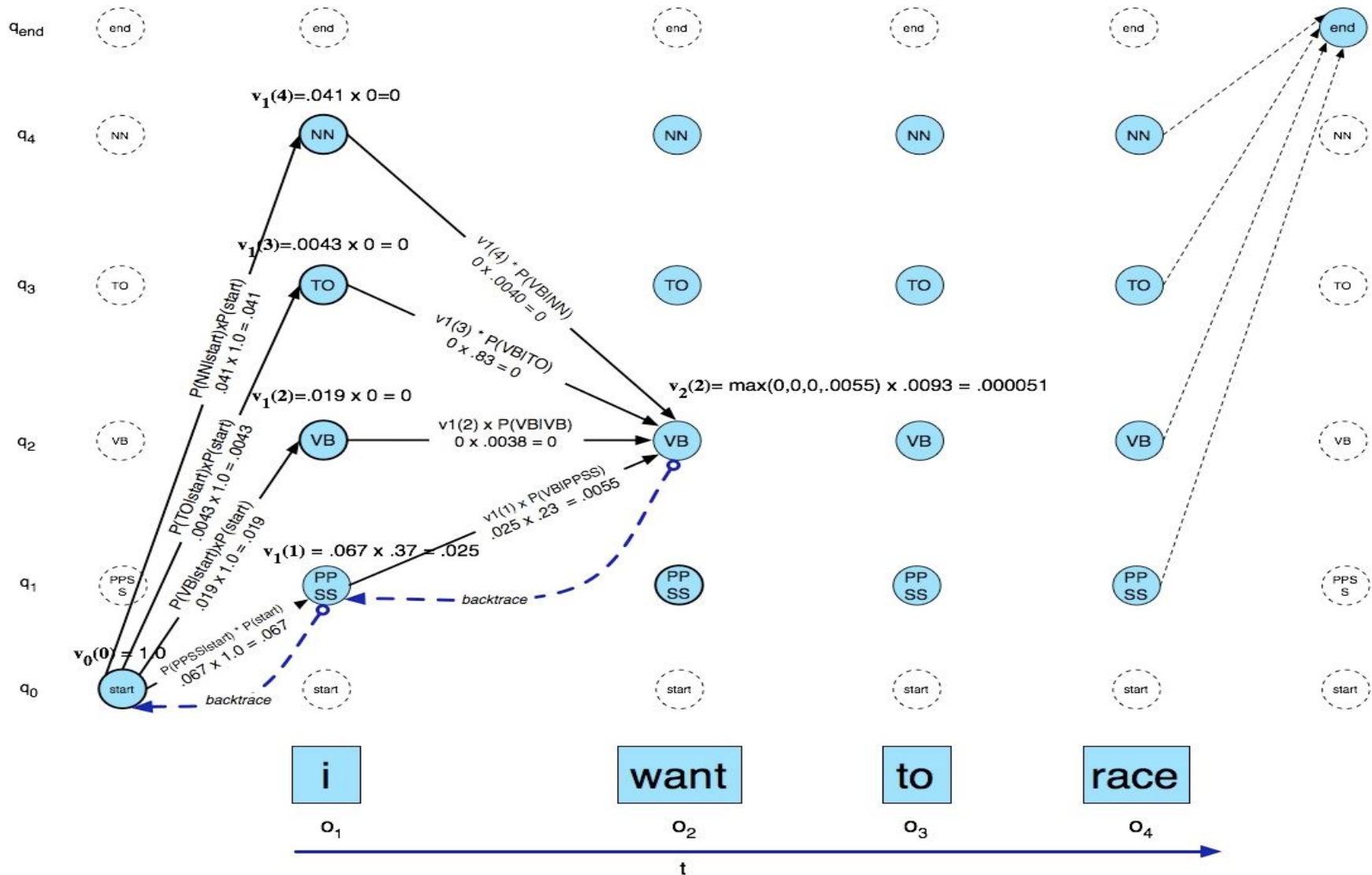
$viterbi[q_F, T] \leftarrow \max_{s=1}^N viterbi[s, T] * a_{s,q_F}$; termination step

$backpointer[q_F, T] \leftarrow \operatorname{argmax}_{s=1}^N viterbi[s, T] * a_{s,q_F}$; termination step

return the backtrace path by following backpointers to states back in time from $backpointer[q_F, T]$



Viterbi Example: Variable $v_t[i]$ the Viterbi path probability at time t for state i



Viterbi Summary

- Create a matrix (two-dimensional array)
 - With columns corresponding to inputs
 - Rows corresponding to possible states
- Sweep through the array in one pass filling the columns **left to right** using our transition probs and observations probs
- Dynamic programming key is that we need only store the **MAX** prob path to each cell, (not all paths).



Summary

- HMM
 - Transition Probabilities
 - Observation Likelihoods
- Decoding
 - Viterbi
- Next
 - Evaluation
 - Assigning probabilities to inputs
 - Forward
 - Finding optimal parameters for a model



Evaluation

- The result is compared with a manually coded “Gold Standard”
 - Typically accuracy reaches 96-97%
 - This may be compared with result for a baseline tagger (one that uses no context).
- Important: 100% is impossible even for human annotators.



HMM

- Given this framework there are 3 problems that we can pose to an HMM
 - Given an observation sequence and a model, what is the most likely state sequence?
 - Given an observation sequence, what is the probability of that sequence given a model?
 - Given an observation sequence, infer the best parameters for model



Problem

- Most probable state sequence given a model and an observation sequence

Decoding: Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, \dots, o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \dots q_T$.

- Typically used in tagging problems, where the tags correspond to hidden states
- Viterbi solves problem

Problem

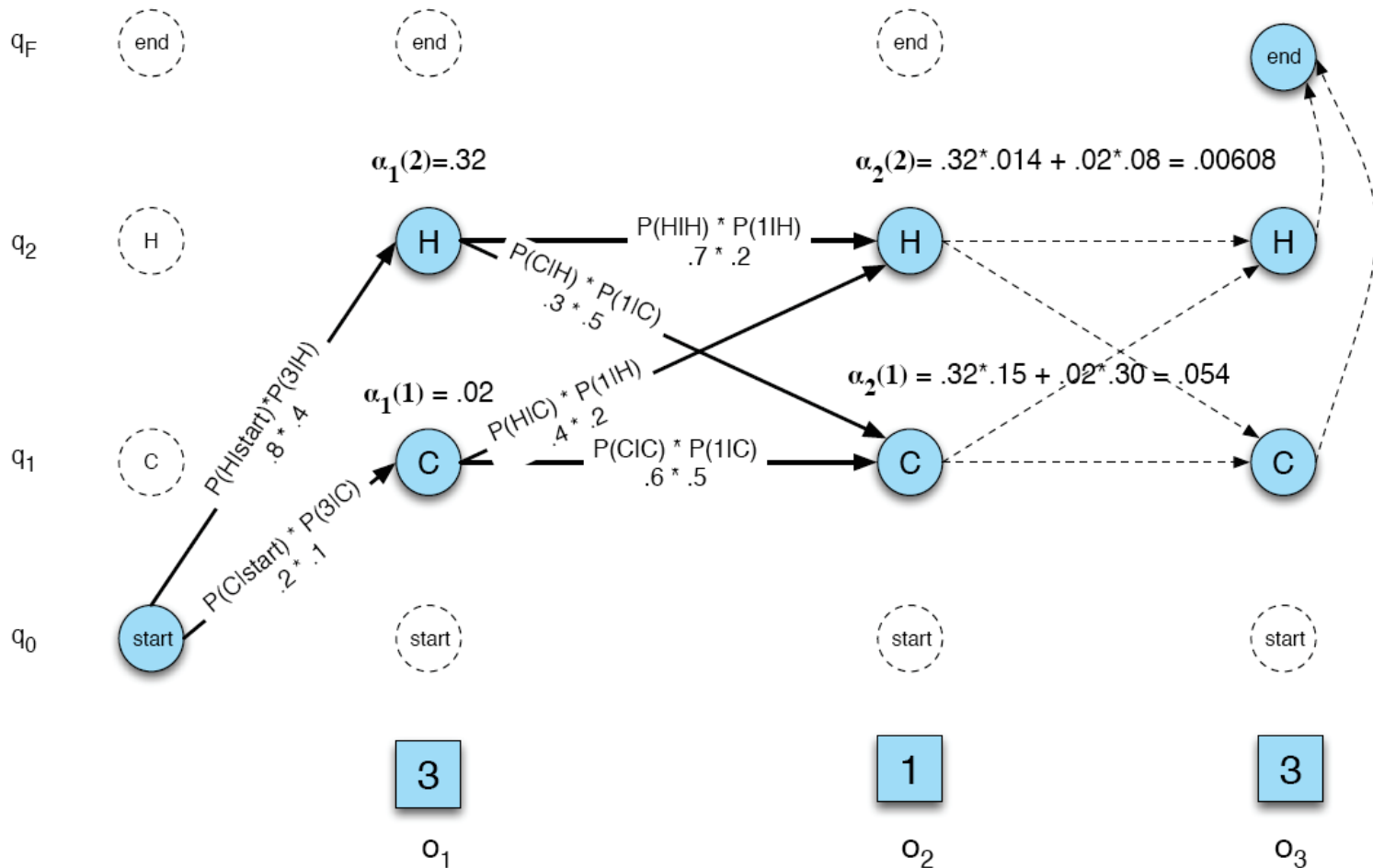
- The probability of a sequence given a model.. $P(seq|model)$.

Computing Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence O , determine the likelihood $P(O|\lambda)$.

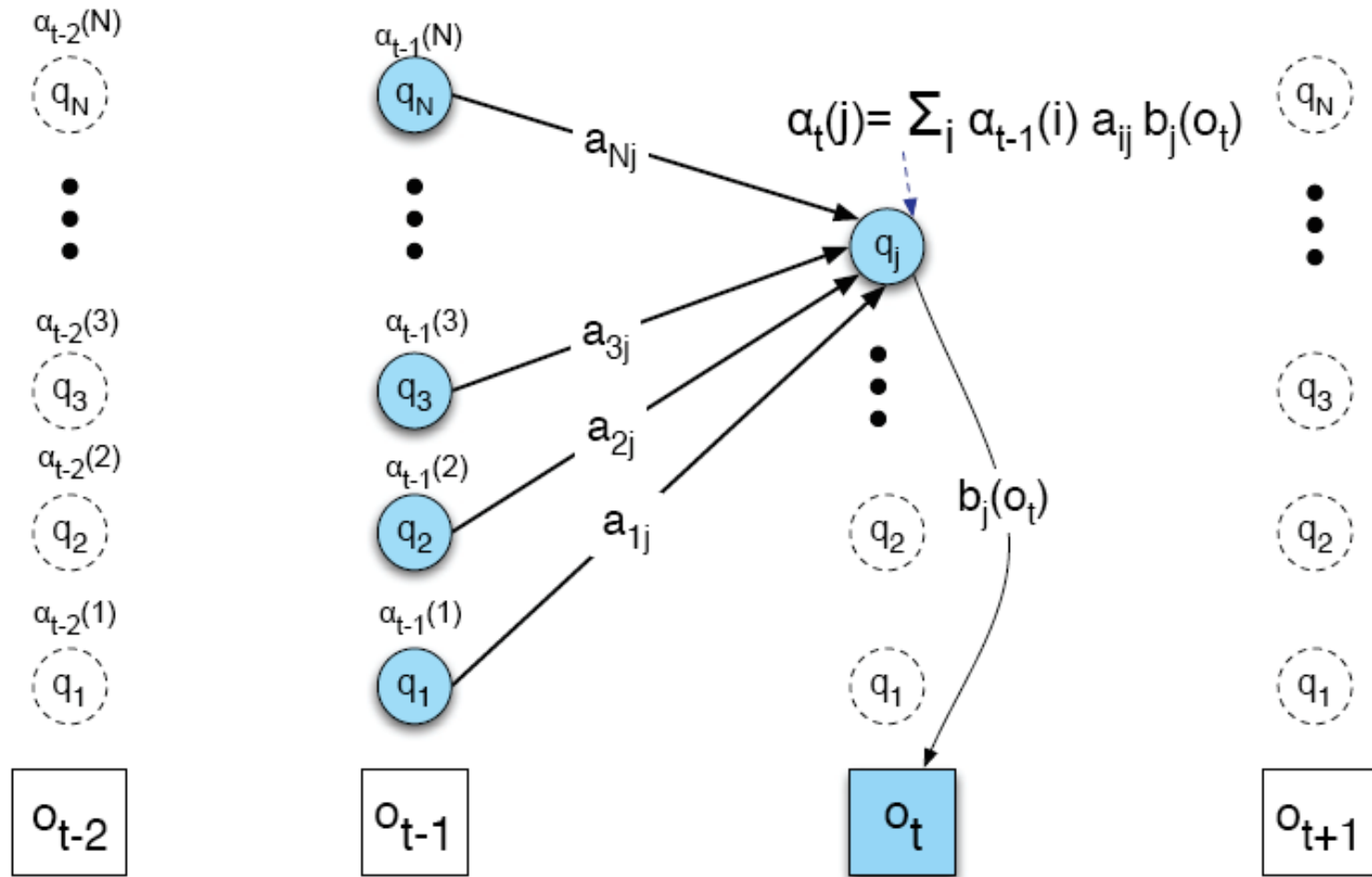
- Forward algorithm
 - Efficiently computes the probability of an observed sequence given a model
 - $P(sequence|model)$
 - Nearly identical to Viterbi: replace the MAX with a SUM

Ice Cream Example

Variable $a_t[i]$ the forward path probability at time t for state i



Forward algorithm: SUM



Forward

function FORWARD(*observations* of len T , *state-graph* of len N) **returns** *forward-prob*

create a probability matrix $forward[N+2, T]$

for each state s **from** 1 **to** N **do** ; initialization step

$$forward[s, 1] \leftarrow a_{0,s} * b_s(o_1)$$

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

for each state s **from** 1 **to** N **do**

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

$$forward[q_F, T] \leftarrow \sum_{s=1}^N forward[s, T] * a_{s,q_F} \quad ; \text{termination step}$$

return $forward[q_F, T]$

Summary

- HMM model- two probabilities
- Viterbi algorithm
- Evaluation
- Three problems in HMM model

