CS 4248
Natural Language Processing

Professor NG Hwee Tou

Department of Computer Science
School of Computing
National University of Singapore
nght@comp.nus.edu.sg

Chapter 5: Part-of-Speech Tagging

- Part-of-speech (POS), word class, morphological class, lexical tag
- Examples: noun, verb, adjective, adverb, determiner, preposition, etc.

Utility of POS

- Give info about a word and its neighbors
  - A possessive pronoun (my, his) likely to be followed by a noun
  - A personal pronoun (I, he) likely to be followed by a verb
- The same word in different POS may be pronounced differently
  - lives (noun) vs. lives (verb)
  - lead (noun) vs. lead (verb)

Utility of POS

- POS can help to decide the correct morphological affixes
  - chairs (chair +N +PL)
  - chairs (chair +V +3SG)
- POS tags provide useful info to other NLP components (parsing, word sense disambiguation)
- POS can help to determine the structure of a sentence (e.g., noun phrase boundary)

English POS

- Definition of POS
  - Morphological property: words in the same POS have similar affixes
  - Distributional property: words in the same POS occur with similar surrounding words
- Classes of POS
  - Open class: noun, verb, adjective, adverb
  - Closed class: the rest (determiner, preposition, conjunction, etc.); function words

Tagsets for English

- Penn Treebank (45 POS tags)
  - Used for tagging Brown corpus and Wall Street Journal corpus
- Brown corpus (87 POS tags)
  - Used in the original Brown corpus
- C5 tagset (61 POS tags)
  - Used for tagging British National Corpus (BNC)
- C7 tagset (146 POS tags)

Penn Treebank POS Tags

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*			

Penn Treebank POS Tags

Example:

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

Penn Treebank POS Tags

there with different POS tags:

There/**EX** are 70 children there/**RB**

VBN (event/verb) vs. JJ (property):

They were married/**VBN** by the Justice of the Peace.

At the time, she was already married/**JJ** .

POS Tagging

- Input:
  - A sentence *S* (and a POS tagset)
  - E.g. “The grand jury commented on a number of other topics .”
- Output:
  - One single best POS tag for each word in *S*
  - E.g., “The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.”

POS Tagging

- Resolve ambiguity
  - Verb: *book* that flight
  - Noun: hand me that *book*
  - Determiner: Does *that* flight serve dinner
  - Subordinating conjunction: I thought *that* your flight was earlier

POS Tagging

- Most words in English are unambiguous, but many of the most common words are ambiguous
- DeRose (1988):
  - 11.5% of word types in Brown corpus are ambiguous
  - Over 40% of word tokens are ambiguous

POS Tagging

- Approaches:
  1. Rule-based tagging
  2. Stochastic HMM (hidden Markov model) tagging

Rule-based Tagging

- Two stages:
  1. Use a dictionary to assign each word a list of potential POS
  2. Use a large list of hand-written disambiguation rules to narrow down to a single best POS for each word
     - E.g., an ambiguous word is a noun rather than a verb if it follows a determiner

EngCG Tagger

- Lexicon
  - 56,000 entries for English word stems
  - Each entry has morphological/syntactic features
- 3,744 rules/constraints
  - Rule out incorrect POS

Stochastic POS Tagging

$$
\begin{aligned}
\hat{T} &= \arg\max_{T} P(T \mid W) \\
&= \arg\max_{T} \frac{P(T, W)}{P(W)} \\
&= \arg\max_{T} P(T, W)
\end{aligned}
$$

Stochastic POS Tagging

$$\begin{aligned} P(T,W) &= P(<s>, t_1, w_1, t_2, w_2, \ldots, t_T, w_T, </s>) \\ &= P(<s>) \cdot P(t_1 \mid <s>) \cdot P(w_1 \mid <s>, t_1) \cdot \\ &\quad P(t_2 \mid <s>, t_1, w_1) \cdot P(w_2 \mid <s>, t_1, w_1, t_2) \cdot \ldots \cdot \\ &\quad P(t_T \mid <s>, t_1, w_1, \ldots, t_{T-1}, w_{T-1}) \cdot \\ &\quad P(w_T \mid <s>, t_1, w_1, \ldots, t_{T-1}, w_{T-1}, t_T) \cdot \\ &\quad P(</s> \mid <s>, t_1, w_1, \ldots, t_T, w_T) \end{aligned}$$

$$P(<s>) = 1$$

$$P(t_i \mid <s>, t_1, w_1, \ldots, t_{i-1}, w_{i-1}) \approx P(t_i \mid t_{i-1})$$

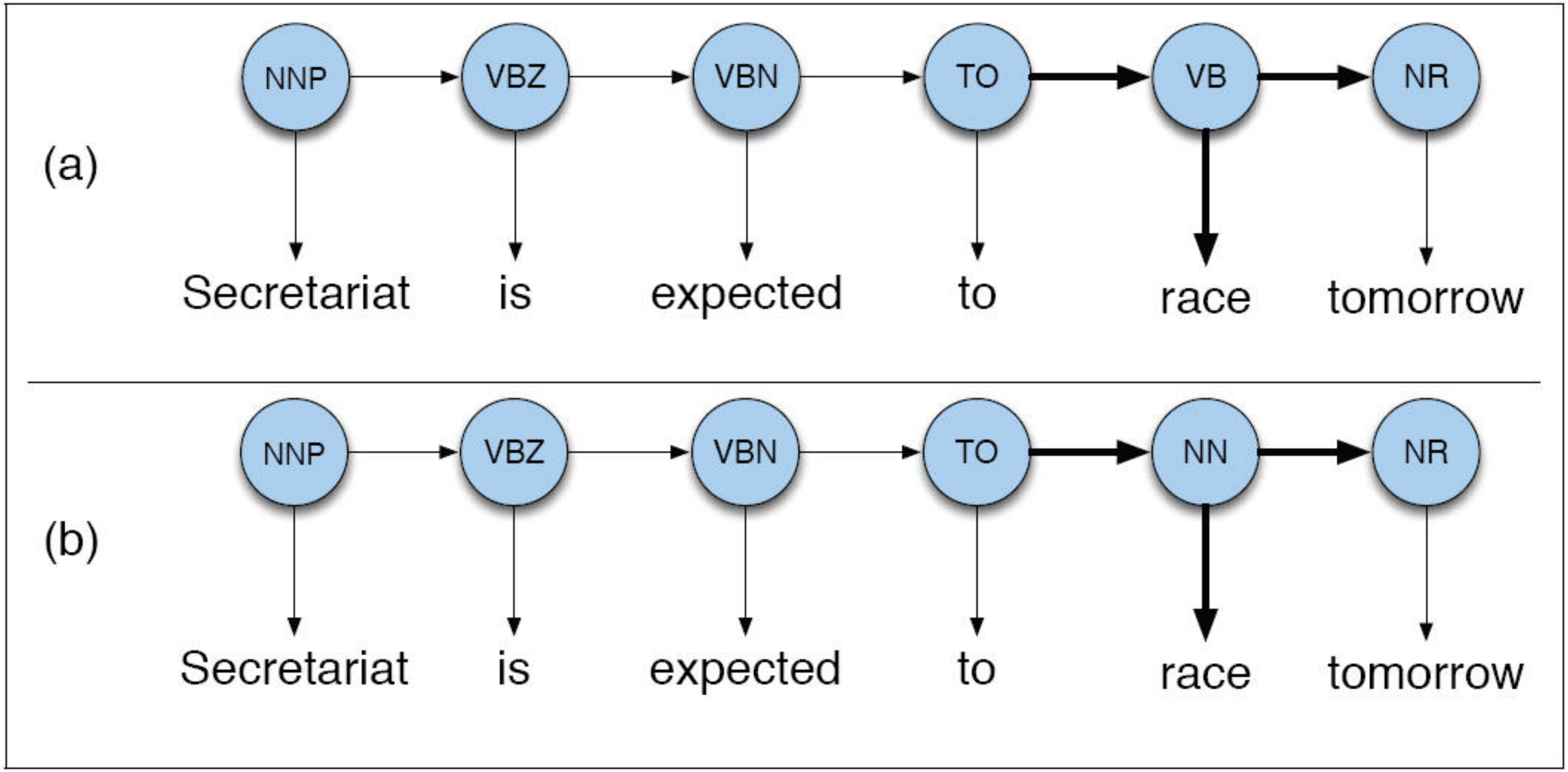
$$P(w_i \mid <s>, t_1, w_1, \ldots, t_{i-1}, w_{i-1}, t_i) \approx P(w_i \mid t_i)$$

$$P(</s> \mid <s>, t_1, w_1, \ldots, t_T, w_T) \approx P(</s> \mid t_T)$$

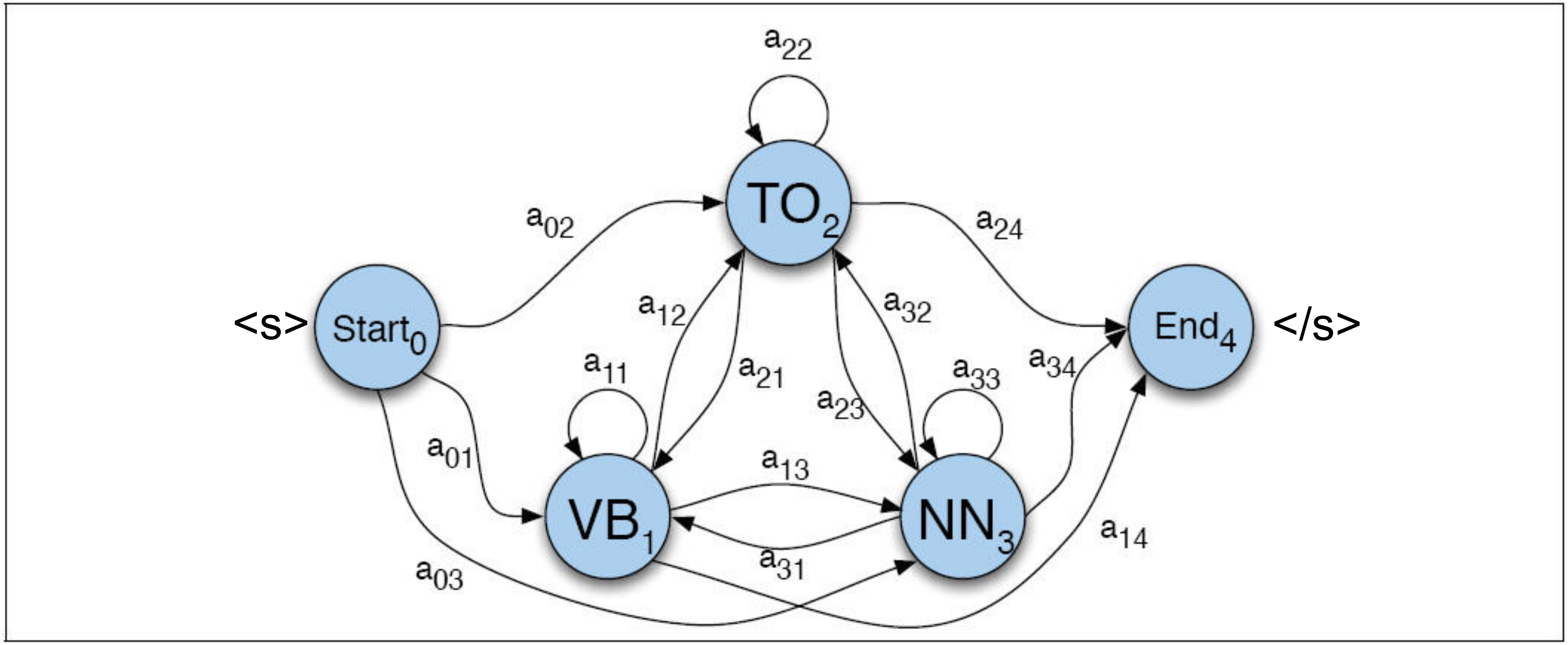
$$P(T,W) = \left(\prod_{i=1}^{T} P(t_i \mid t_{i-1}) \cdot P(w_i \mid t_i)\right) \cdot P(</s> \mid t_T)$$

$$\hat{T} = \arg\max_{t_1,\ldots,t_T} \left\{ \left(\prod_{i=1}^{T} P(t_i \mid t_{i-1}) \cdot P(w_i \mid t_i)\right) \cdot P(</s> \mid t_T) \right\}$$

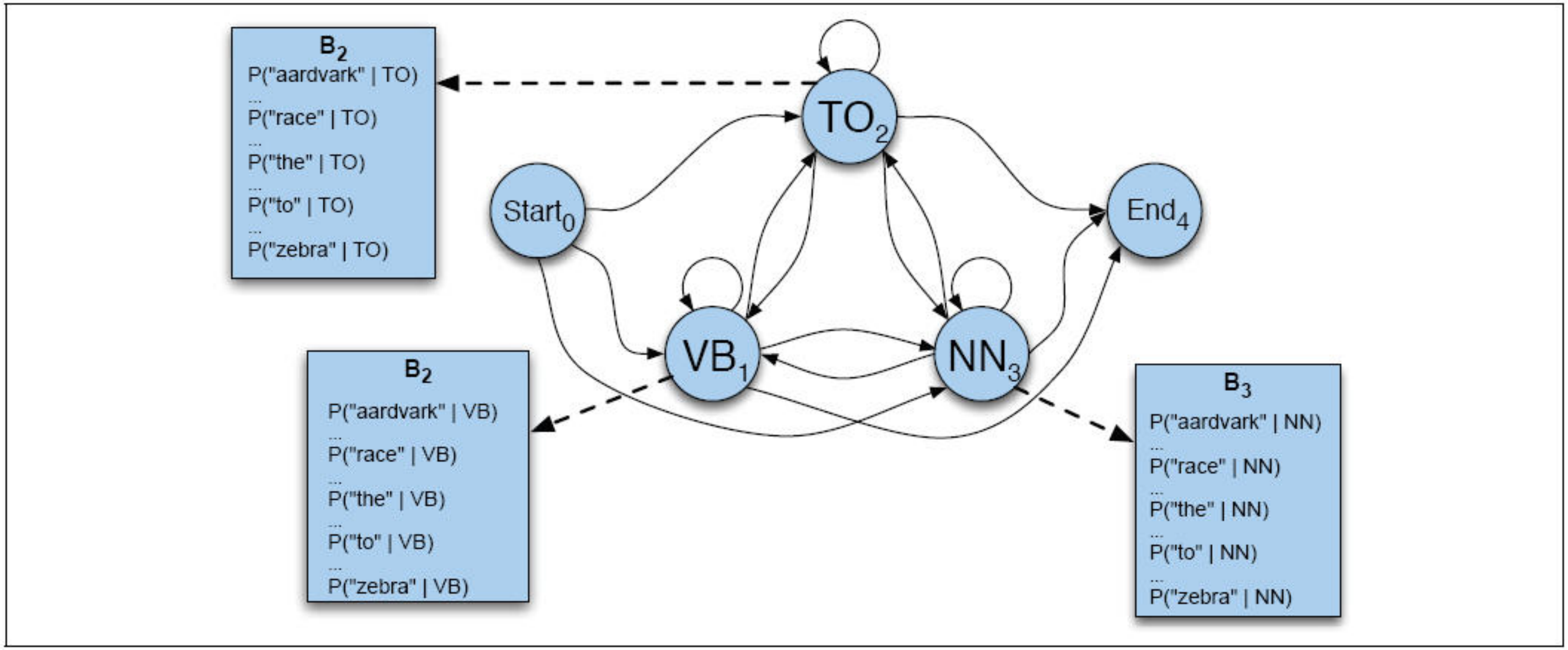
Stochastic POS Tagging



Hidden Markov Model (HMM)



Hidden Markov Model (HMM)

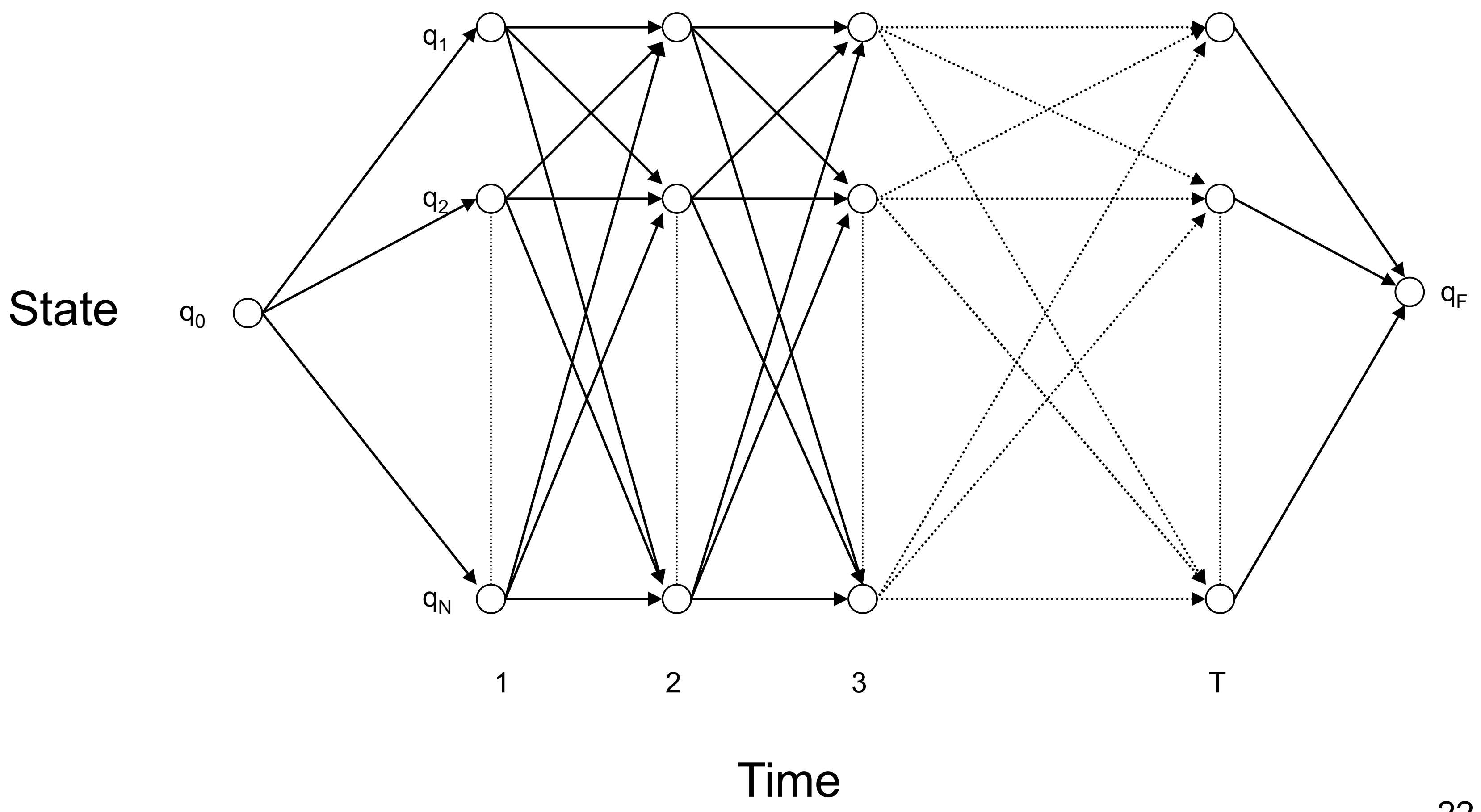


Hidden Markov Model (HMM)

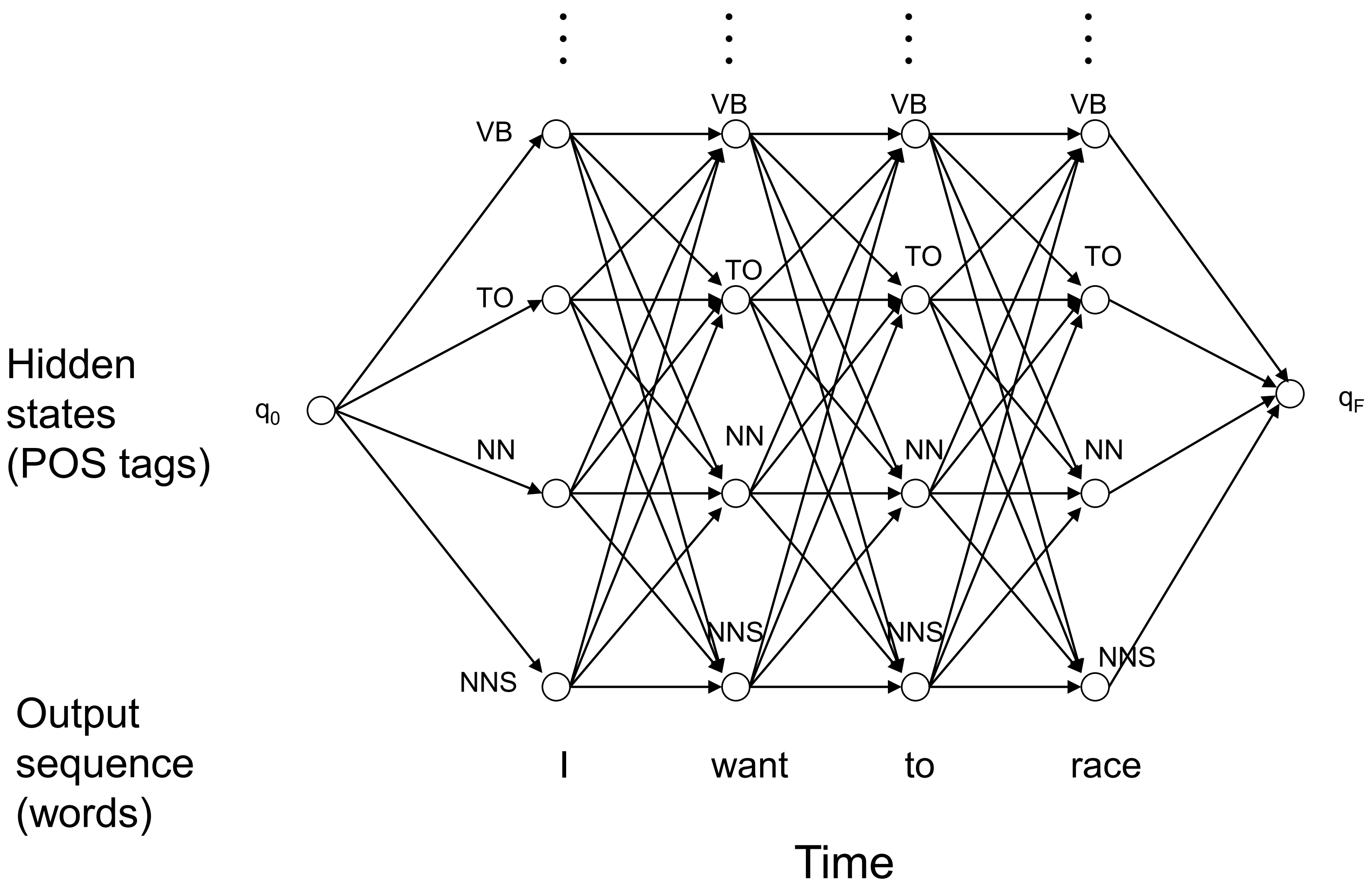
$Q = q_1 q_2 \ldots q_N$	a set of N **states**.
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$.
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$.
$B = b_i(o_t)$	A sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i.
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state.

$$\forall i \quad \sum_{o_t \in V} b_i(o_t) = 1$$

Hidden Markov Model (HMM)



Illustrating Example



Hidden Markov Model (HMM)

- Determine the most probable state sequence
  - Direct evaluation: $O(T \cdot N^T)$
  - *T* = no. of words
  - *N* = no. of POS tags

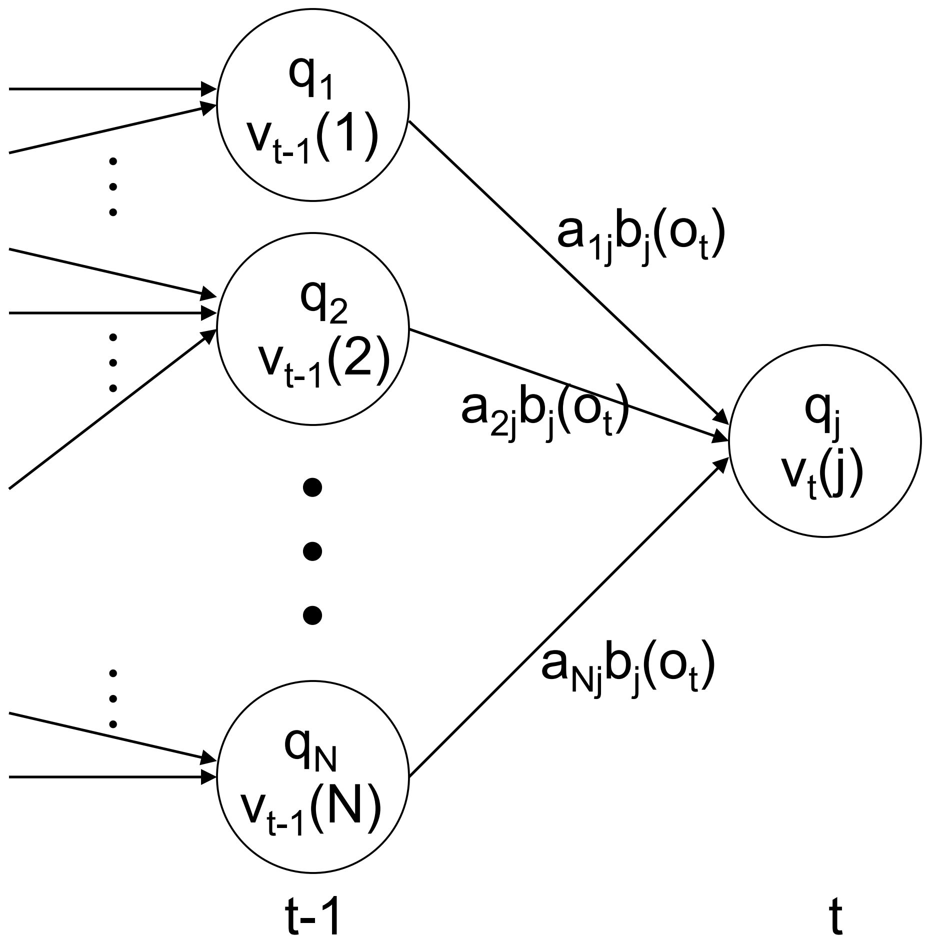
$$\hat{T} = \arg\max_{t_1,\ldots,t_T} \left\{ \left(\prod_{i=1}^{T} P(t_i \mid t_{i-1}) \cdot P(w_i \mid t_i) \right) \cdot P(</s> \mid t_T) \right\}$$

Viterbi Algorithm

- Dynamic programming algorithm: $O(T \cdot N^2)$

$v_t(j)$: max probability of all paths ending in state q_j at time t

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



Viterbi Algorithm

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 $backpointer[q_F, T]$

Illustrating Example

$P(t_i \mid t_{i-1})$

t_i

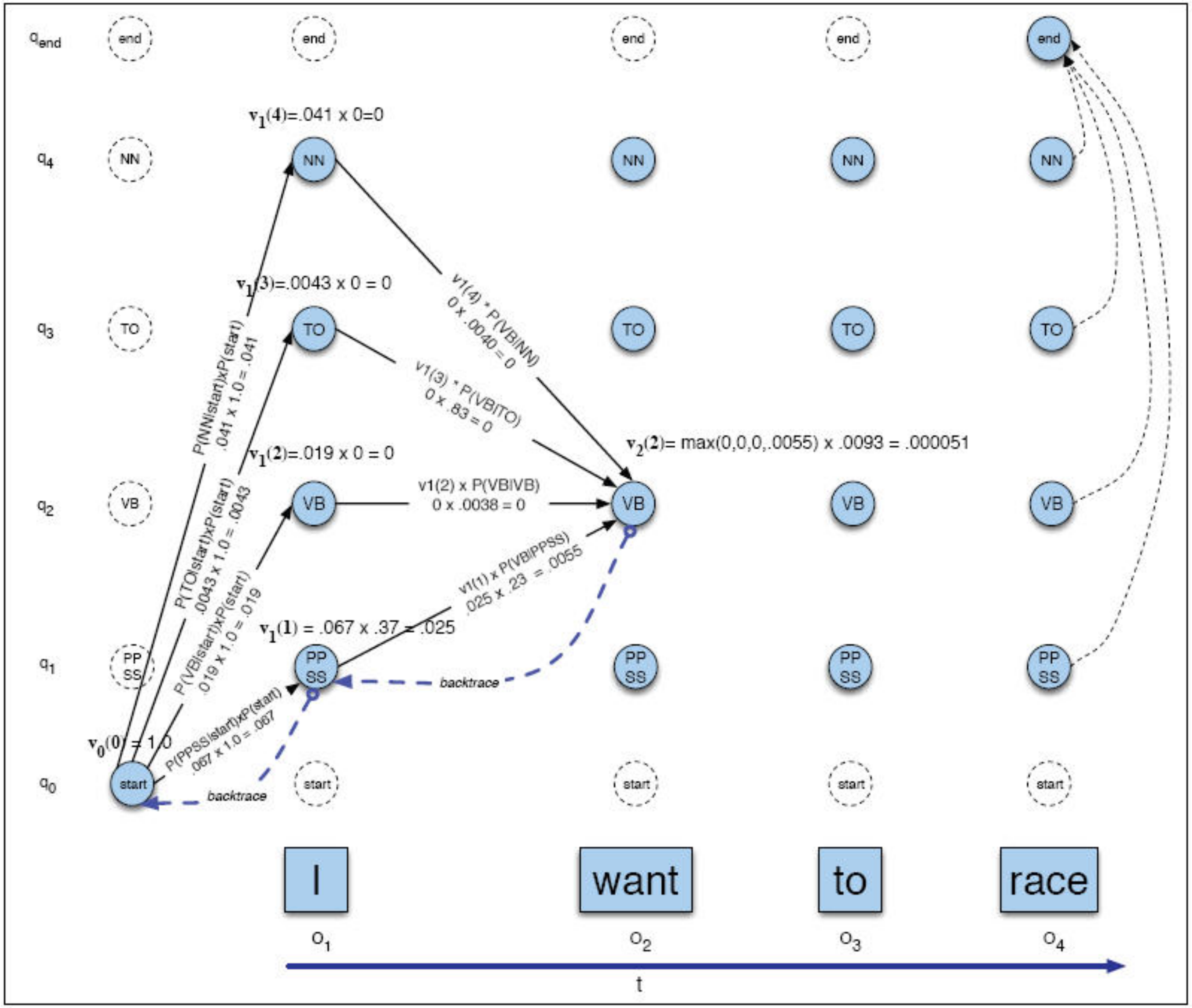
t_{i-1}	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

$P(w_i \mid t_i)$

w_i

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

Illustrating Example



Stochastic POS Tagging

- Unknown word model
  - Unknown words similar to words that occurred only once in training data
- Depend on factors:
  - Ratio of unknown words
  - Capitalization (capitalized initial, capitalized non-initial)
  - Morphological suffixes (-s -ed -ing -ion -al -ive etc.)
- If w_i is an unknown word:

$$P(w_i \mid t_i) = P(\text{unknownword} \mid t_i) \cdot P(\text{capital} \mid t_i) \cdot P(\text{endings/hyph} \mid t_i)$$

Evaluating POS Taggers

- Baseline (lower bound):
  - Unigram most probable tag
    - Assign each word to the POS tag it occurs in most often in the training set
    - 90%
- State-of-the-art POS taggers:
  - 97.9% (Penn Treebank tagset)

Evaluation

- N-fold cross-validation
  - Randomly split the data set into N equal parts (D_1, …, D_N)
  - For i = 1, …, N:
    - let D_i be the test set and the union of the remaining D_j ($j \neq i$) be the training set
    - Train a tagger on the training set and evaluate on the test set. Let the accuracy be A_i
  - Average A_i to obtain the average accuracy

Error Analysis

Confusion matrix (Contingency table):

Predicted class

True class

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	—	.2			.7		
JJ	.2	—	3.3	2.1	1.7	.2	2.7
NN		8.7	—				.2
NNP	.2	3.3	4.1	—	.2		
RB	2.2	2.0	.5		—		
VBD		.3	.5			—	4.4
VBN		2.8				2.6	—