Natural Language Processing

BITS Pilani
Pilani Campus

Dr. Chetana Gavankar, Ph.D,
IIT Bombay-Monash University Australia
Chetana.gavankar@pilani.bits-pilani.ac.in

Session 6-Part-of-Speech Tagging

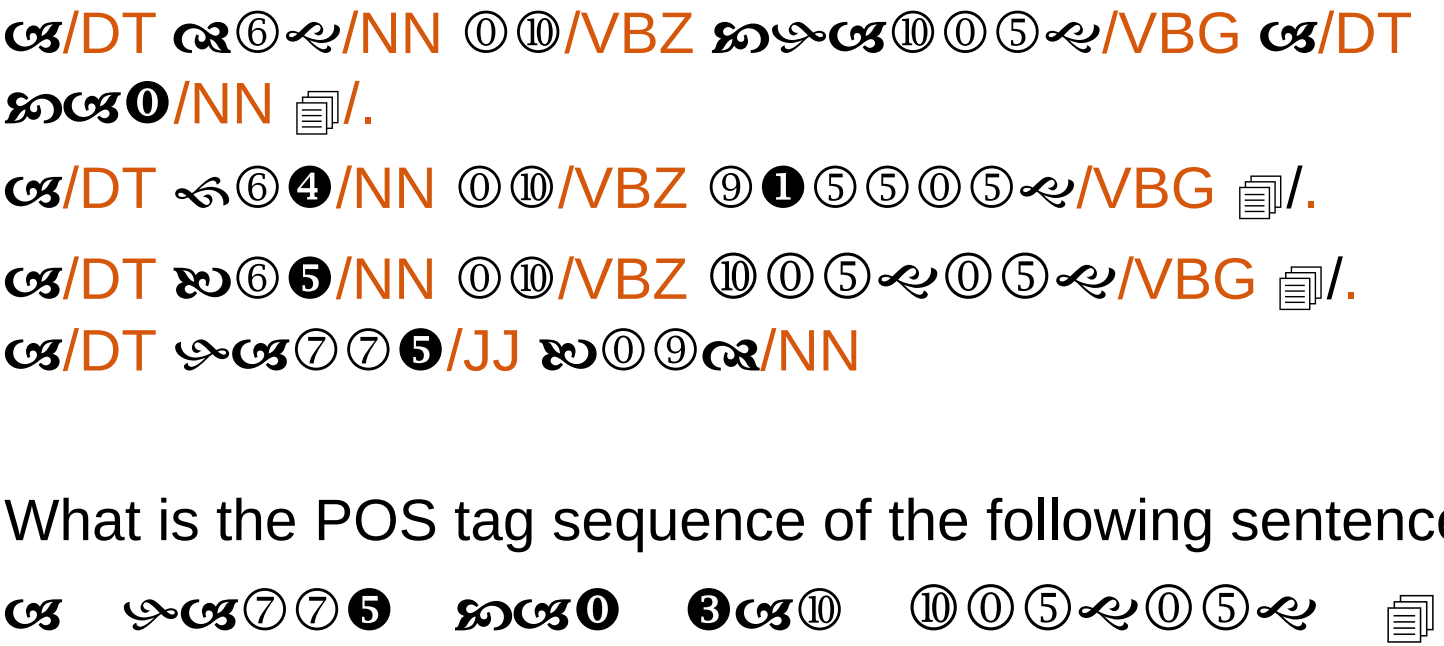
These slides are prepared by the instructor, with grateful acknowledgement of Jurafsky and Martin and many others who made their course materials freely available online.

Session Content

- Part-of-Speech Tagging
- POS tagging real world applications
- HMM Part-of-Speech Tagging
- Viterbi Algorithm
- Maximum Entropy Markov Model
- Bidirectionality

Let’s try

Don’t worry! There is no problem with your eyes or computer.



What is the POS tag sequence of the following sentence?

POS Tagging

- The process of assigning a part-of-speech or lexical class marker to each word in a sentence (and all sentences in a collection).

Input: **`the lead paint is unsafe`**

Output: **`the/Det lead/N paint/N is/V unsafe/Adj`**

POS tagging real world applications

- **Voice Assistants**
  Identify commands, actions (verbs), and targets (nouns)

- **Chatbots & Agentic AI**
  Ground user requests into API/tool actions safely

- **Search Engine Query Understanding**
  Distinguish between nouns vs. actions for relevant results
  Example: “play cricket rules” vs. “cricket playstation game”

- **Information Extraction**
  Find key entities, noun phrases in finance, legal, healthcare text

- **Grammar Checking & Writing Assistants**
  Detect incorrect verb forms, mismatched phrases

POS tagging real world applications

- **Machine Translation**
  Understand grammatical roles to select correct word order

- **Social Media Analytics**
  Better sentiment extraction by understanding adjective/noun structure

- **Document Structuring in Enterprises**
  Auto-cataloging: titles (nouns) vs. processes (verbs)

- **Text-to-Speech Systems**
  Correct intonation (e.g., questions vs. statements)

- **Biomedical NLP**
  Disambiguate drug names vs. actions (dosage instructions)

POS tagging in Agentic AI

User says:
"Set the timer for the oven to 10 minutes."

What the agent must extract:
•**Action** → "set", **Target device** → "oven", **Parameter** → "timer = 10 mins"

Risk without POS validation

LLM misinterprets: "**Set the oven for 10 minutes**" → might think **turn on oven** or **set temperature.** This could trigger a dangerous action.

With POS Tagging: Agent chooses the **Timer API** instead of Oven control API.

Word	POS Tag	Reason
Set	Verb	Action to execute
the timer	Noun Phrase	Actual object of verb
for the oven	Prepositional Phrase	Device context
10 minutes	Numeric time expression	Parameter

How POS Tagging helps

innovate achieve lead

Why POS tagging improves safety

- **More deterministic, fewer unsafe actions**
- **Better user trust + compliance in agentic systems**

Problem	POS Solution
Misinterpreting “oven” as the object of “set”	It’s attached via preposition (“for the oven”)
Ambiguity in parameters	Validates time-expression noun phrase
Hallucinated actions	Enforces correct verb-object relationship

Structured/explicit POS tagging remains useful in agents for speed, safety, and determinism
— especially statistical + neural models.

POS Tagging Approaches

Approach	Accuracy	Speed/ Compute	Explainability	Best Use Today
Rule-based	★★☆☆☆	★★★★★	★★★★★	Fixed-domain, edge devices
Statistical (HMM/CRF)	★★★★☆	★★★★☆	★★★☆☆	Hybrid agent pipelines
Neural (RNN/CNN)	★★★★★	★★★☆☆	★★☆☆☆	Legacy NLP production systems
Transformer-based	★★★★☆	★★☆☆☆	★★☆☆☆	High-accuracy enterprise systems
LLM-based implicit	★★★★☆	★☆☆☆☆	★★☆☆☆	Agentic AI with semantic flexibility

Cost of these approaches grows exponentially as we move from rule based to LLM based

Parts of Speech

- 8 (ish) traditional parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc
  - Called: parts-of-speech, lexical categories, word classes, morphological classes, lexical tags...
  - Lots of debate within linguistics about the number, nature, and universality of these
    - We'll completely ignore this debate.

POS Tagging

- The process of assigning a part-of-speech or lexical class marker to each word in a collection.

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

Open and Closed Classes

- Closed class: a small fixed membership
  - Prepositions: of, in, by, …
  - Auxiliaries: may, can, will had, been, …
  - Pronouns: I, you, she, mine, his, them, …
  - Usually function words (short common words which play a role in grammar)
- 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!

POS Tagging
Choosing a Tagset

- There are so many parts of speech, potential distinctions we can draw
- To do POS tagging, we need to choose a standard set of tags to work with
- Could pick very coarse tagsets
  - N, V, Adj, Adv.
- More commonly used set is finer grained, the “Penn TreeBank tagset”, 45 tags
  - PRP$, WRB, WP$, VBG
- Even more fine-grained tagsets exist

Penn TreeBank POS 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*			

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”.

POS Tagging

- 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 as Sequence Classification

- We are given a sentence (an "observation" or "sequence of observations")
  - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags that corresponds to this sequence of observations?
- 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 $w_1 \ldots w_n$.

Information Extraction

- Identify phrases in language that refer to specific types of entities and relations in text.
- Named entity recognition is task of identifying names of people, places, organizations, etc. in text.

  people organizations places

  – Michael Dell is the CEO of Dell Computer Corporation and lives in Austin Texas.
- Extract pieces of information relevant to a specific application, e.g. used car ads:

  make model year mileage price

  – For sale, 2002 Toyota Prius, 20,000 mi, $15K or best offer. Available starting July 30, 2006.

Semantic Role Labeling

- For each clause, determine the semantic role played by each noun phrase that is an argument to the verb.

  agent patient source destination instrument

  – John drove Mary from Austin to Dallas in his Toyota Prius.

  – The hammer broke the window.

- Also referred to a “case role analysis,” “thematic analysis,” and “shallow semantic parsing”

Bioinformatics

- Sequence labeling also valuable in labeling genetic sequences in genome analysis.

  extron intron

  – AGCTAACGTTCGATACGGATTACAGCCT

Problems with Sequence Labeling as Classification

- Not easy to integrate information from category of tokens on both sides.
- Difficult to propagate uncertainty between decisions and “collectively” determine the most likely joint assignment of categories to all of the tokens in a sequence.

Probabilistic Sequence Models

- Probabilistic sequence models allow integrating uncertainty over multiple, interdependent classifications and collectively determine the most likely global assignment.
- standard model
  - Hidden Markov Model (HMM)

Hidden Markov Models

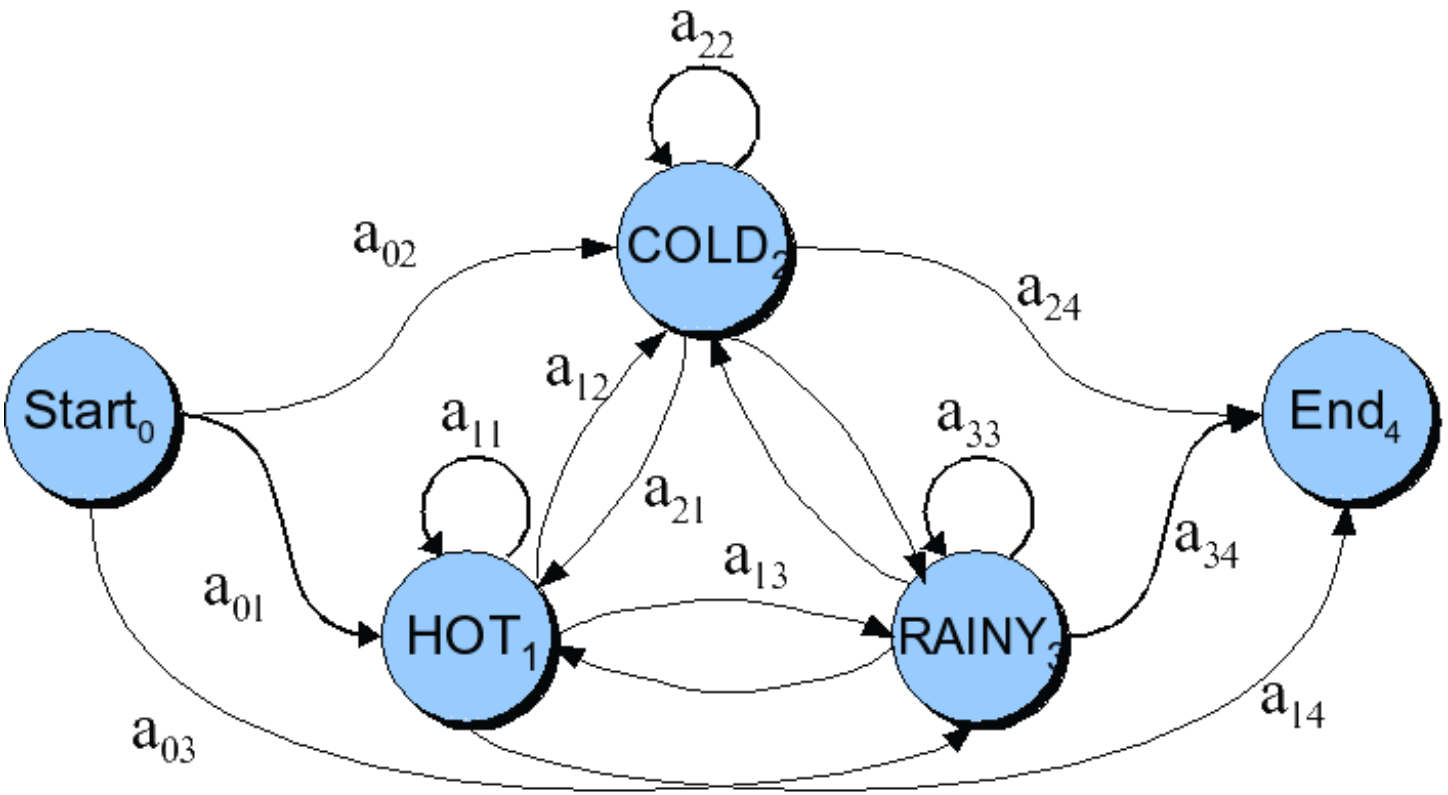
- It is a **sequence model**.
- Assigns a label or class to each unit in a sequence, thus mapping a **sequence of observations** to a **sequence of labels**.
- Probabilistic sequence model: given a sequence of units (e.g. words, letters, morphemes, sentences), compute a probability distribution over possible sequences of labels and choose the best label sequence.
- This is a kind of *generative* model.

Markov Chain: "First-order observable Markov Model"

- A set of states
  - Q = q_1, $q_2 \ldots q_{N;}$ the state at time t is q_t
- Transition probabilities:
  - a set of probabilities A = $a_{01}a_{02} \ldots a_{n1} \ldots a_{nn}$.
  - Each a_{ij} represents the probability of transitioning from state i to state j
  - The set of these is the transition probability matrix A
  - Special **initial** probability vector π
- Current state only depends on previous state

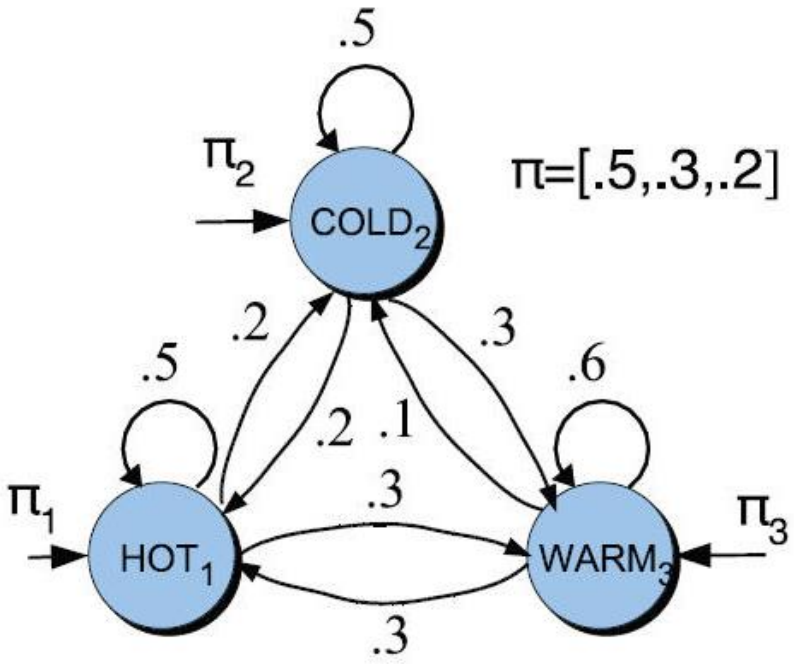
$$P(q_i \mid q_1 .. q_{i-1}) = P(q_i \mid q_{i-1})$$

Markov Chain for Weather



Markov Chain for Weather

- What is the probability of 4 consecutive warm days?



- Sequence is warm-warm-warm-warm
- And state sequence is 3-3-3-3
- P(3,3,3,3) =
  - $\pi_3 a_{33} a_{33} a_{33} = 0.2 \times (0.6)^3 = 0.0432$

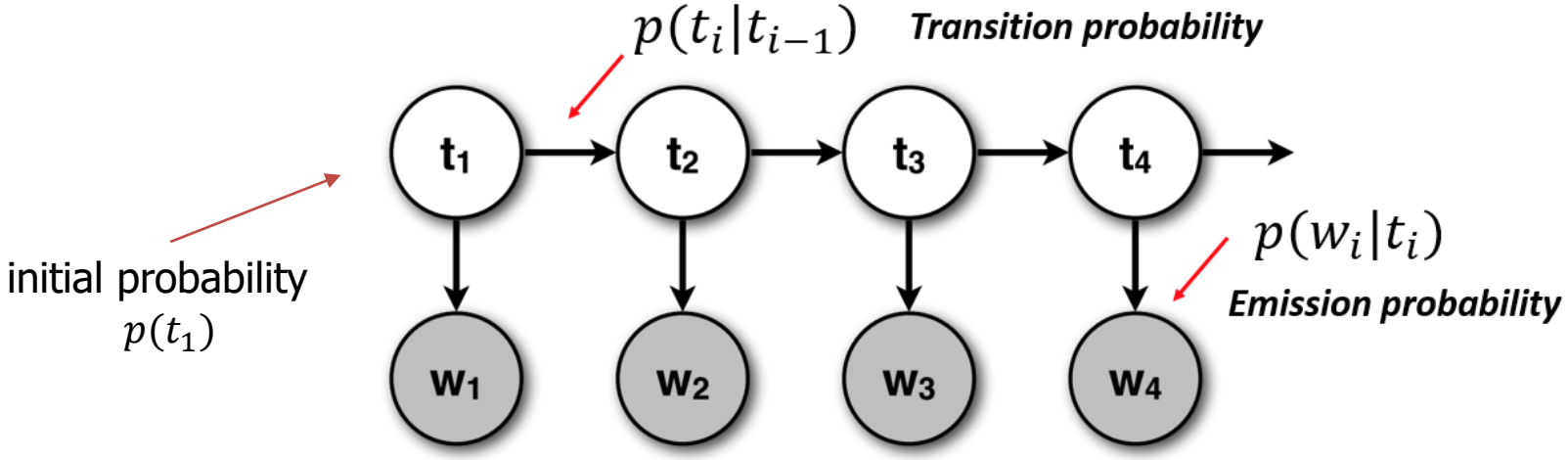
HMM to predict the tags

- Two types of information are useful
  - Relations between words and tags
  - Relations between tags and tags
    - DT NN, DT JJ NN…

P(DT JJ NN | a smart dog)
= P(DD JJ NN a smart dog) / P (a smart dog)
= P(DD JJ NN) P(a smart dog | DD JJ NN)

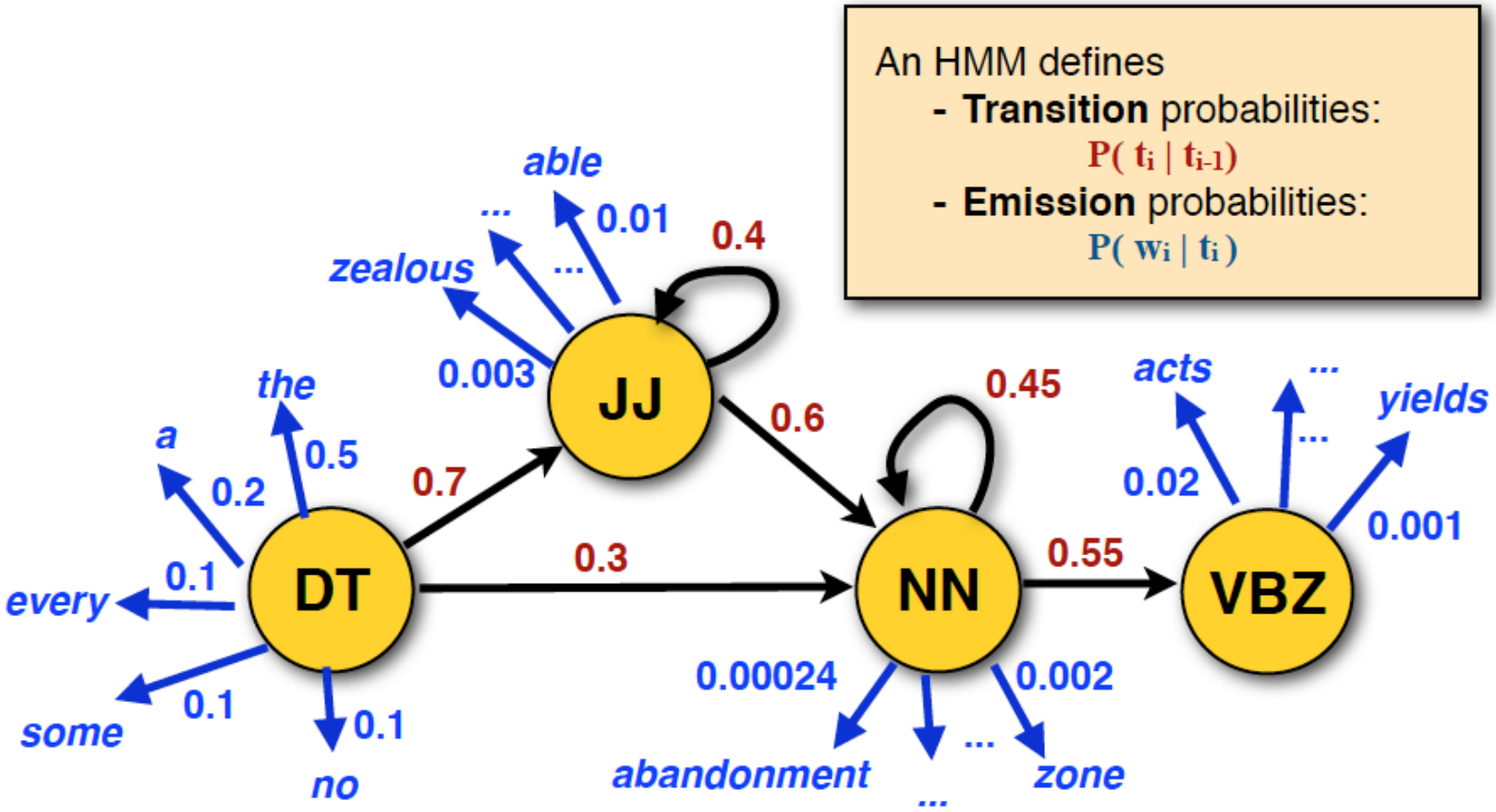
Prediction in generative model

- Inference: What is the most likely sequence of tags for the given sequence of words **w**



- What are the latent states that most likely generate the sequence of word **w**

HMMs as probabilistic FSA

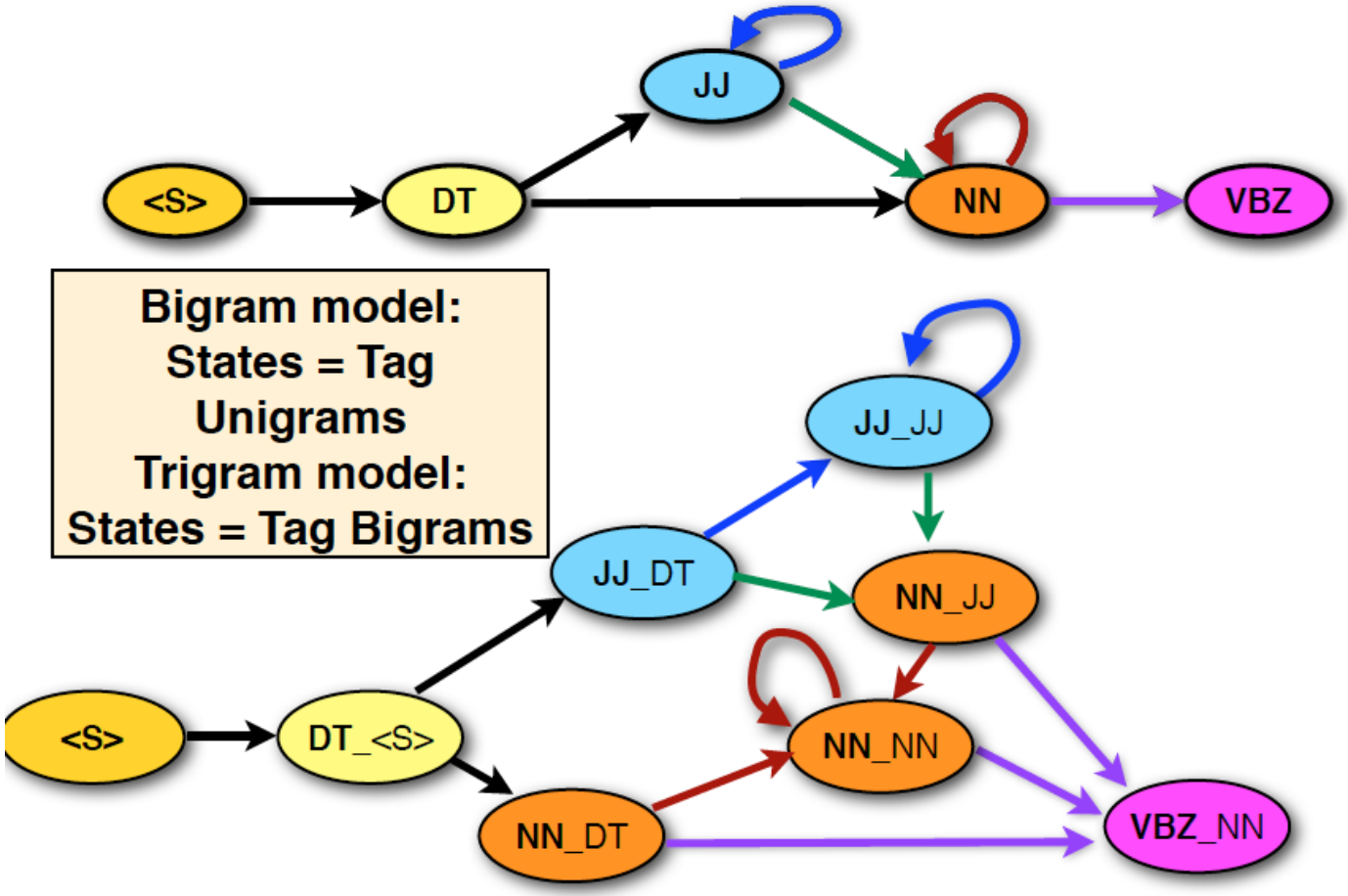


Julia Hockenmaier: Intro to NLP

How to build a second-order HMM?

- Second-order HMM
  - Current state only depends on previous 2 states
- Example
  - Trigram model over POS tags
  - $P(\boldsymbol{t}) = \Pi_{i=1}^{n} P(t_i \mid t_{i-1}, t_{i-2})$
  - $P(\boldsymbol{w}, \boldsymbol{t}) = \Pi_{i=1}^{n} P(t_i \mid t_{i-1}, t_{i-2}) P(w_i \mid t_i)$

Probabilistic FSA for second-order HMM



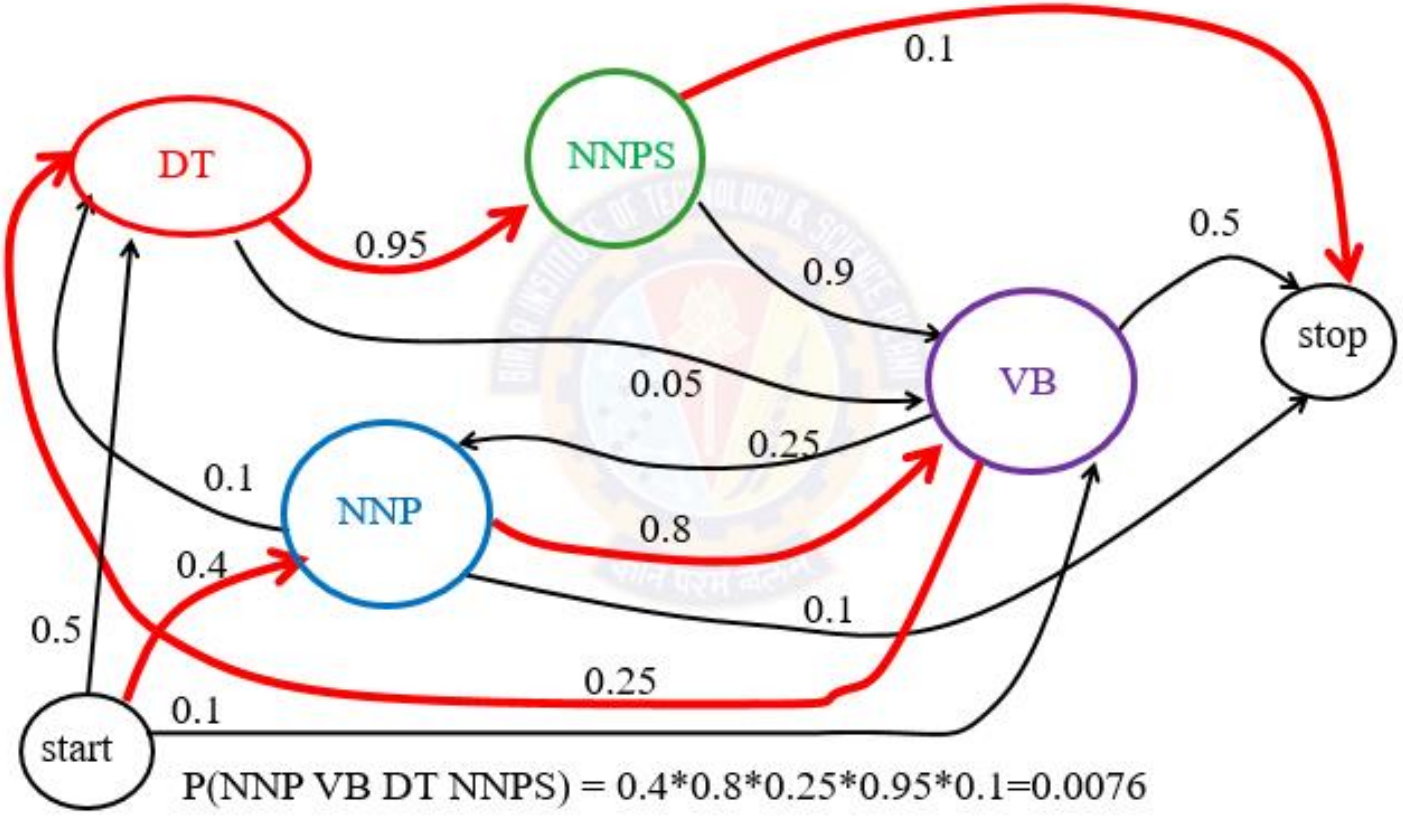
Julia Hockenmaier: Intro to NLP

Example

- Given a sentence of length 3, * * the dog barks STOP and the tag sequence * * DT NN VB * then

- P(w1 w2 w3,y1,y2,y3) = T(DT|*,*) X T(NN|*,DT) X T(VB|DT NN) X T(STOP|NN VB) X E(*|*) X E(*|*) E(the|DT) X E(dog|NN) X E(barks|VB)X E(STOP|*)

- We can also define y_{-1}=* and y_0=* as special symbols.


DT
NNPS
VB
stop
NNP
start
0.95
0.9
0.1
0.5
0.05
0.25
0.1
0.8
0.4
0.1
0.5
0.25
0.1
P(NNP VB DT NNPS) = 0.4*0.8*0.25*0.95*0.1=0.0076

Hidden Markov Models (POS Tagging)

- States $T = t_1, t_2 \ldots t_{N;}$
- Observations $W = w_1, w_2 \ldots w_{N;}$
  - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, \ldots v_V\}$
- Transition probabilities
  - Transition probability matrix $A = \{a_{ij}\}$

$$a_{ij} = P(t_i = j \mid t_{i-1} = i) \;\; 1 \leq i, j \leq N$$

- Observation likelihoods
  - Output probability matrix $B = \{b_i(k)\}$

$$b_i(k) = P(w_i = v_k \mid t_i = i)$$

- Special initial probability vector π

$$\pi_i = P(t_1 = i) \;\; 1 \leq i \leq N$$

Statistical POS Tagging

- We want, out of all sequences of n tags $t_1 \ldots t_n$ the single tag sequence such that

$$P(t_1 \ldots t_n | w_1 \ldots w_n) \text{ is highest.}$$

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

- Hat ^ means "our estimate of the best one"
- Argmax_x f(x) means "the x such that f(x) is maximized"

Statistical POS Tagging

- This equation should give us the best tag sequence

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

- But how to make it operational? How to compute this value?
- Intuition of Bayesian inference:
  - Use Bayes rule to transform this equation into a set of probabilities that are easier to compute (and give the right answer)

Using Bayes Rule

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

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

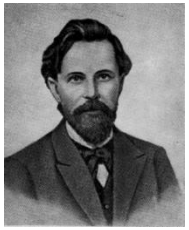
$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

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

Likelihood and Prior

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

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



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

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

Statistical POS tagging

- What is the most likely sequence of tags for the given sequence of words w

$$\begin{aligned}
\operatorname*{argmax}_{\mathbf{t}} P(\mathbf{t}|\mathbf{w}) &= \operatorname*{argmax}_{\mathbf{t}} \frac{P(\mathbf{t},\mathbf{w})}{P(\mathbf{w})} \\
&= \operatorname*{argmax}_{\mathbf{t}} P(\mathbf{t},\mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{t}} P(\mathbf{t})P(\mathbf{w}|\mathbf{t})
\end{aligned}$$

P(DT JJ NN | a smart dog)
= P(DD JJ NN a smart dog) / P (a smart dog)
= P(DD JJ NN) P(a smart dog | DD JJ NN)

Transition Probability

- Joint probability $P(\boldsymbol{t}, \boldsymbol{w}) = P(\boldsymbol{t})P(\boldsymbol{w}|\boldsymbol{t})$
- $P(\boldsymbol{t}) = P(t_1, t_2, \ldots t_n)$

$$= P(t_1)P(t_2 \mid t_1)P(t_3 \mid t_2, t_1) \ldots P(t_n \mid t_1 \ldots t_{n-1})$$

$$\sim \mathrm{P}(\mathrm{t}_1)\mathrm{P}(\mathrm{t}_2 \mid t_1)P(t_3 \mid t_2) \ldots P(t_n \mid t_{n-1})$$

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

Markov assumption

- Bigram model over POS tags!
  (similarly, we can define a n-gram model over POS tags, usually we called high-order HMM)

Emission Probability

- Joint probability $P(\boldsymbol{t}, \boldsymbol{w}) = P(\boldsymbol{t})P(\boldsymbol{w}|\boldsymbol{t})$
- Assume words only depend on their POS-tag
- $P(\boldsymbol{w}|\boldsymbol{t}) \sim P(w_1 \mid t_1)P(w_2 \mid t_2) \ldots P(w_n \mid t_n)$

Independent assumption

$$= \Pi_{i=1}^{n} P(w_i|t_i)$$

i.e., P(a smart dog | DD JJ NN)

= P(a | DD) P(smart | JJ) P(dog | NN)

Put them together

- Joint probability $P(\boldsymbol{t}, \boldsymbol{w}) = P(\boldsymbol{t})P(\boldsymbol{w}|\boldsymbol{t})$
- $P(\boldsymbol{t}, \boldsymbol{w})$

$= \mathrm{P}(\mathrm{t}_1)\mathrm{P}(\mathrm{t}_2 \mid t_1)P(t_3 \mid t_2) \ldots P(t_n \mid t_{n-1})$
$P(w_1 \mid t_1)P(w_2 \mid t_2) \ldots P(w_n \mid t_n)$

$= \Pi_{i=1}^{n} P(w_i|t_i)P(t_i \mid t_{i-1})$

e.g., P(a smart dog , DD JJ NN)

= P(a | DD) P(smart | JJ) P(dog | NN)

P(DD | start) P(JJ | DD) P(NN | JJ)

Two Kinds of Probabilities

1. **State transition probabilities** -- $p(t_i|t_{i-1})$
   - State-to-state transition probabilities

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

2. **Observation/Emission probabilities** -- $p(w_i|t_i)$
   - Probabilities of observing various values at a given state

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

Two Kinds of Probabilities

1. **Tag transition** probabilities -- $p(t_i|t_{i-1})$
   - Determiners likely to precede adjs 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
   - Compute P(NN|DT) by counting in a labeled corpus:

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

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

Two Kinds of Probabilities

2. **Word likelihood/emission** probabilities $p(w_i|t_i)$
   - VBZ (3sg Pres Verb) likely to be "is"
   - Compute P(is|VBZ) by counting in a labeled corpus:

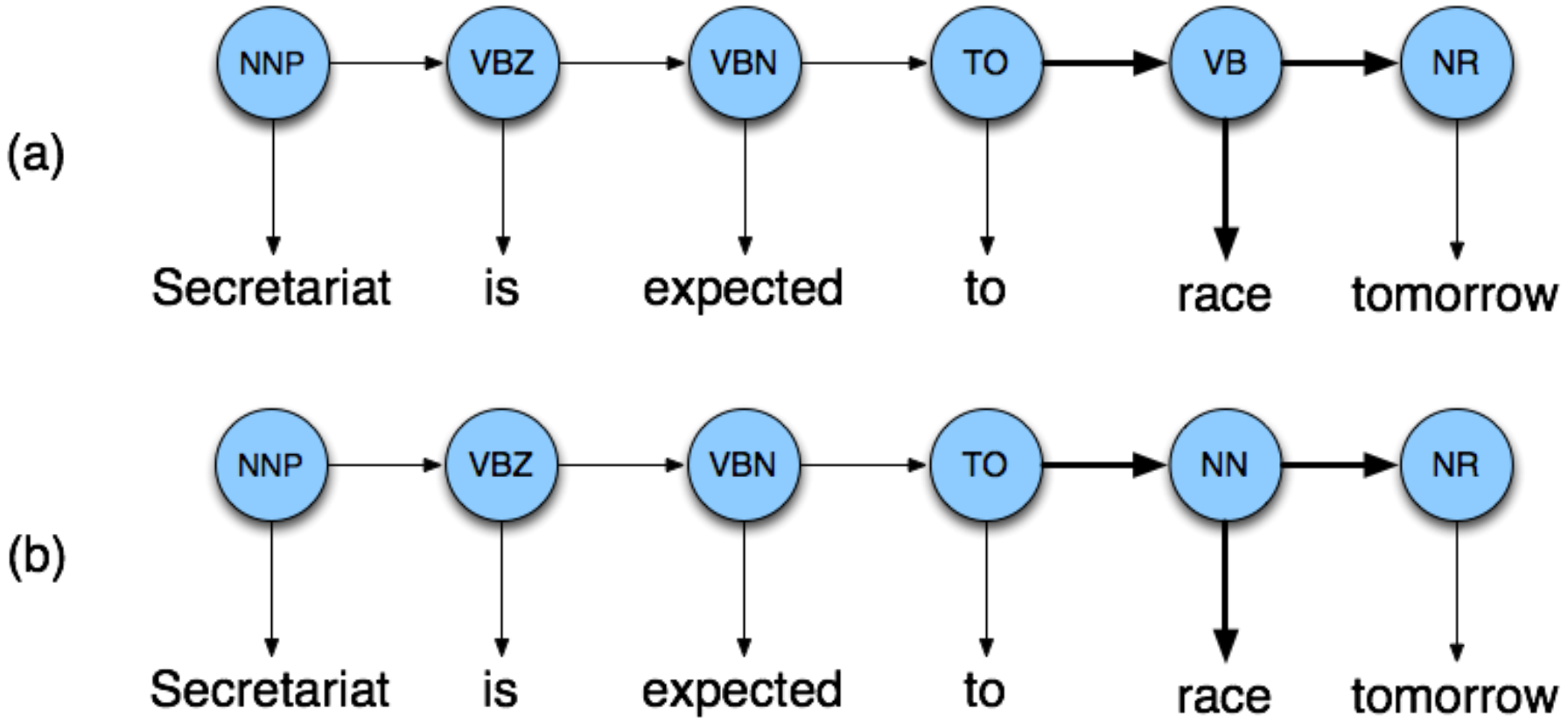
$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

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

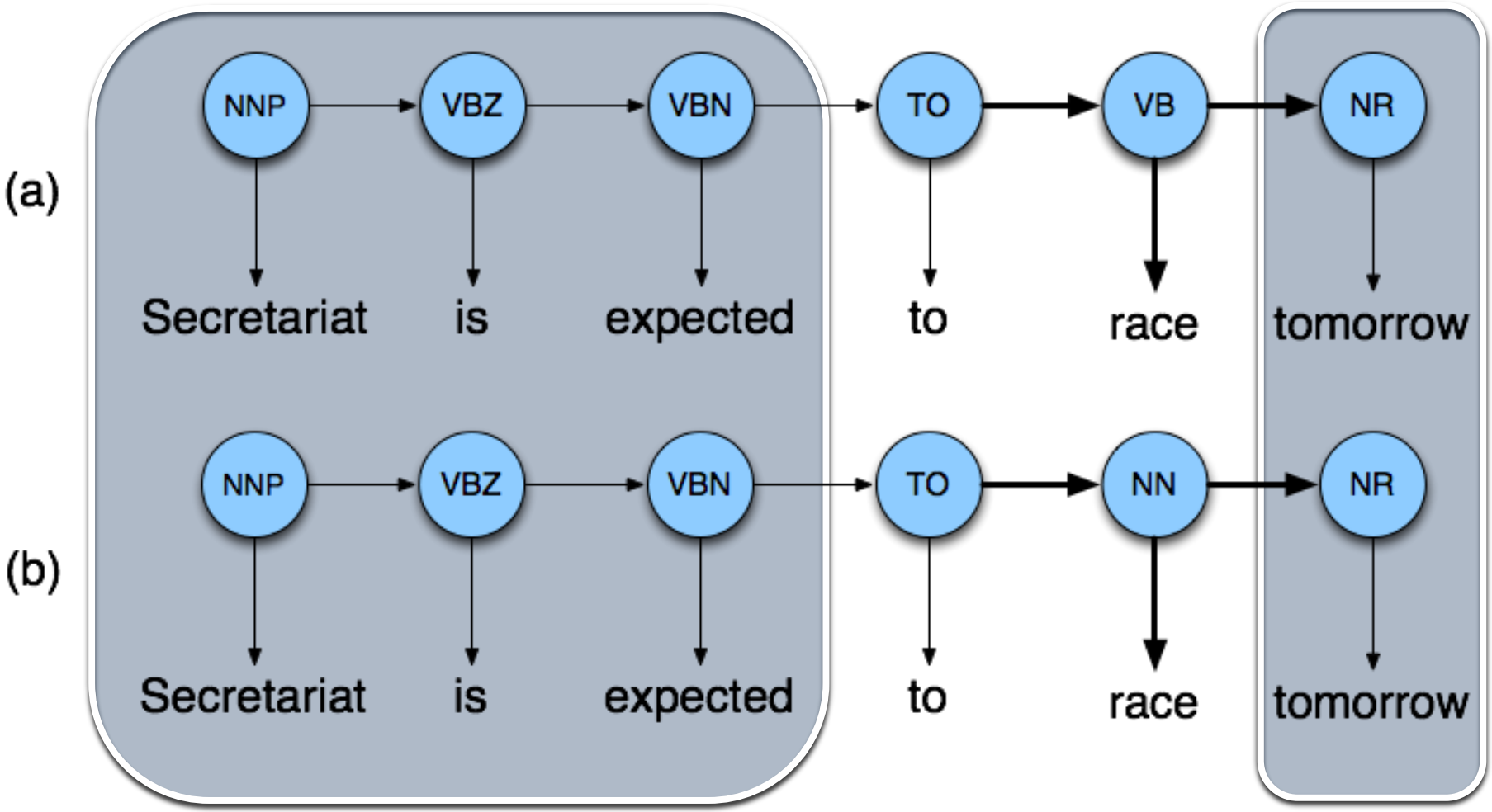
Example: The Verb "race"

- 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?

Disambiguating "race"



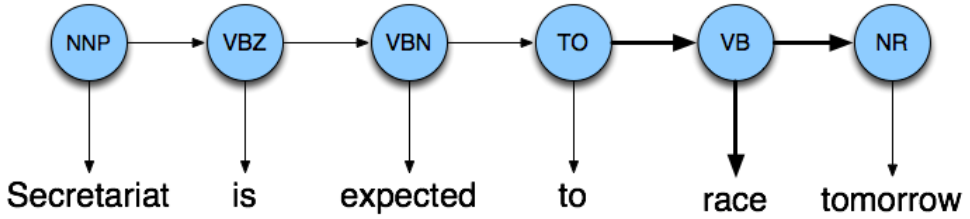
Disambiguating "race"



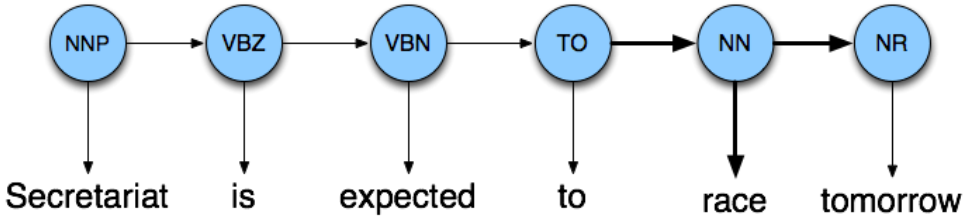
Example : NLTK POS tags

- P(NN|TO) = .00047
- P(VB|TO) = .83
- P(race|NN) = .00057
- P(race|VB) = .00012
- P(NR|VB) = .0027
- P(NR|NN) = .0012

(a)



(b)



NR-Adverbial Noun (tomorrow), TO-proposition

- P(VB|TO)P(NR|VB)P(race|VB) = .00000027
- P(NN|TO)P(NR|NN)P(race|NN)=.00000000032
- So we (correctly) choose the verb tag for "race"

https://www.guru99.com/pos-tagging-chunking-nltk.html

Example

Emission Matrix

	*	DT	NNS	VB	NN	IN	STOP
*	1						
the		3/4					
employees			3/4				
pass				2/4			
an		1/4					
exam					1		
wait				1/4			
for						1	
employers			1/4				
fire				1/4			
.							1

Tag Translation Matrix

SECOND TAG

FIRST TAG

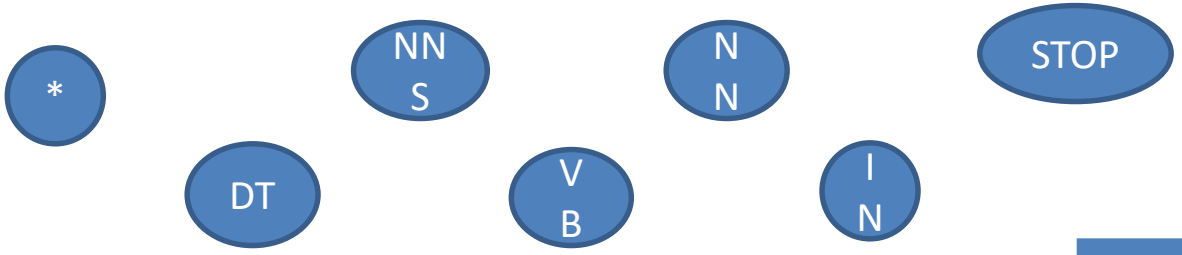
	*	DT	NNS	VB	NN	IN	STOP
*		2/3	1/3				
DT			2/4	1/4	1/4		
NNS				3/4			1/4
VB		1/4	1/4			1/4	1/4
NN							1
IN		1					
STOP							

S1: the Employees pass an exam .
T1: DT NNS VB DT NN STOP

S2: the employees wait for the pass .
T2: DT NNS VB IN DT VB STOP

S3: employers fire employees .
T3: NNS VB NNS STOP

Transition diagram



Tag Translation Matrix

SECOND TAG

FIRST TAG	*	DT	NNS	VB	NN	IN	STOP
*		2/3	1/3				
DT			2/4	1/4	1/4		
NNS				3/4			1/4
VB		1/4	1/4			1/4	1/4
NN							1
IN		1					
STOP							

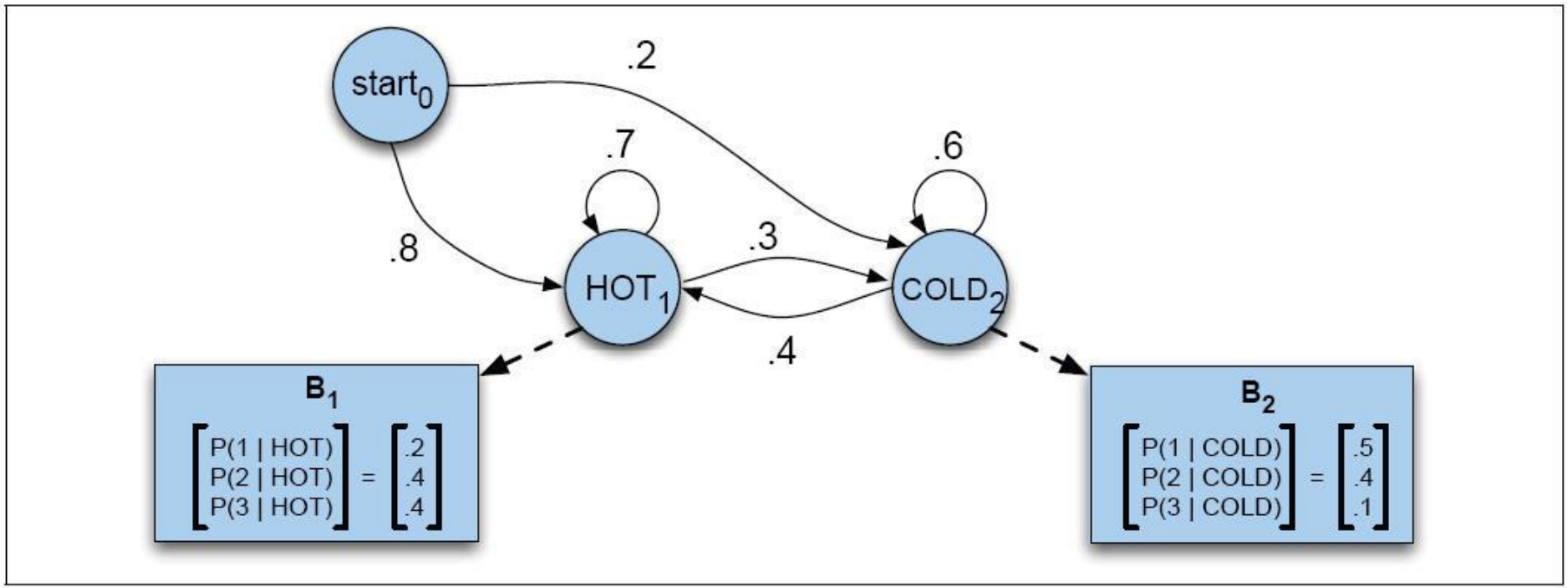
HMMs for Ice Cream

- You are a climatologist in the year 2799 studying global warming
- You can't find any records of the weather in Baltimore for summer of 2007
- But you find Jason Eisner's diary which lists how many ice-creams Jason ate every day that summer
- Your job: figure out how hot it was each day

Eisner Task

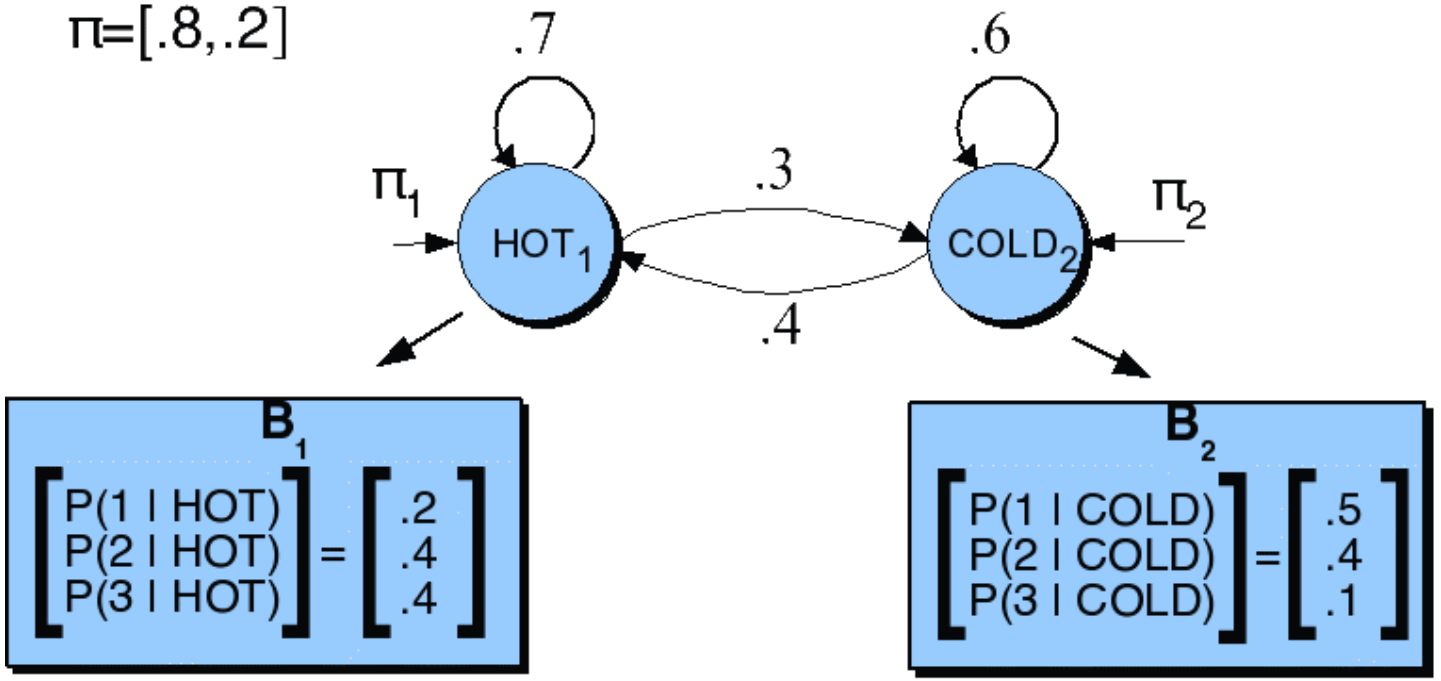
- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3…
- Produce:
  - Hidden Weather Sequence:

    H,C,H,H,H,C, C…



What's the state sequence for the observed sequence "1 3 1"?

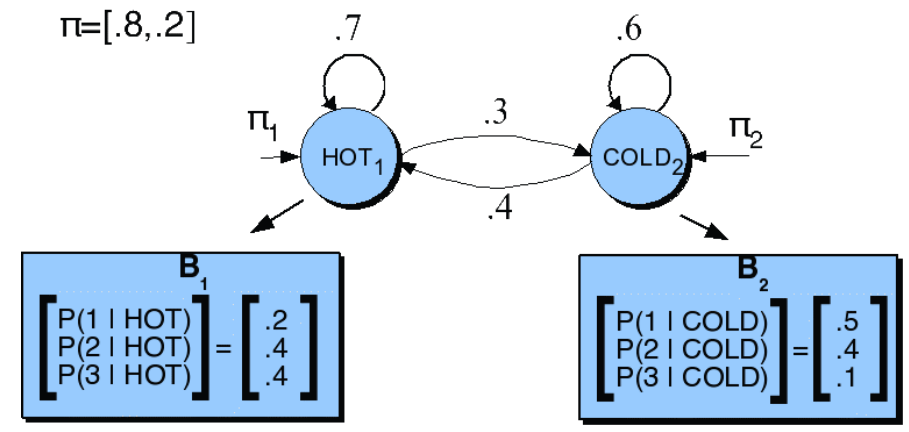
HMM for Ice Cream



Ice Cream HMM

- Let's just do **131** as the sequence
  - How many underlying state (hot/cold) sequences are there?

HHH
HHC
HCH
HCC
CCC
CCH
CHC
CHH



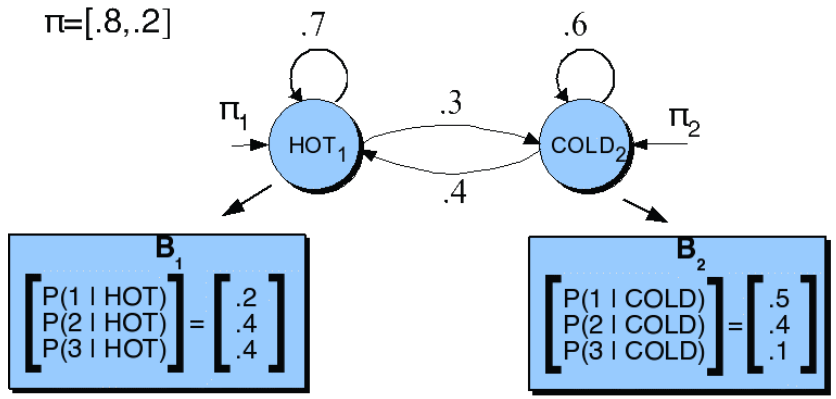
Argmax P(sequence | 1 3 1)

  - How do you pick the right one?

Ice Cream HMM

Let's just do 1 sequence: CHC

Cold as the initial state P(Cold|Start)	.2
Observing a 1 on a cold day P(1 | Cold)	.5
Hot as the next state P(Hot | Cold)	.4
Observing a 3 on a hot day P(3 | Hot)	.4
Cold as the next state P(Cold|Hot)	.3
Observing a 1 on a cold day P(1 | Cold)	.5



P(C H C)
= 0.2*0.5*0.4*0.4*0.3*0.5
= 0.0024

Viterbi Example 2: Ice Cream

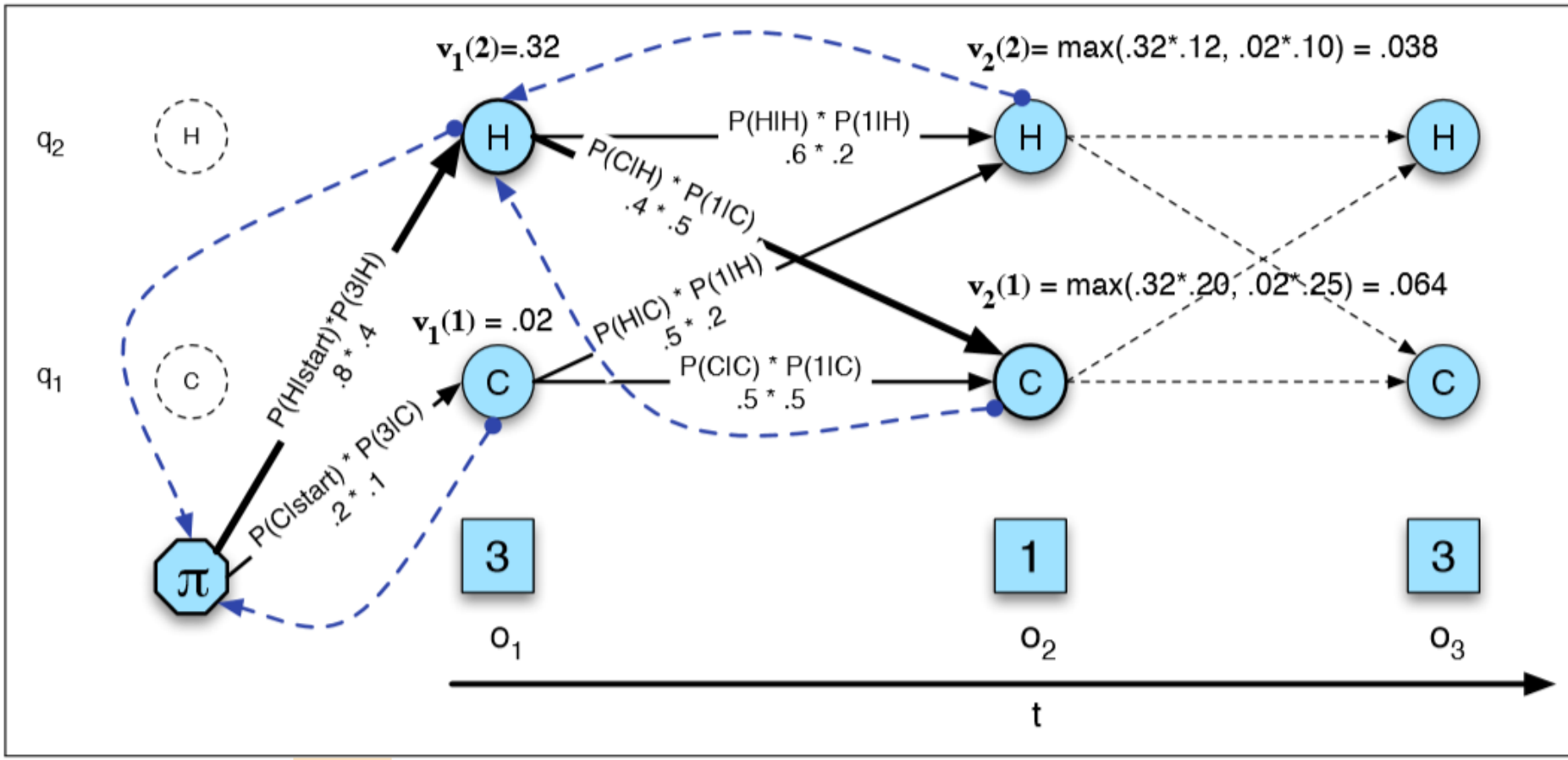
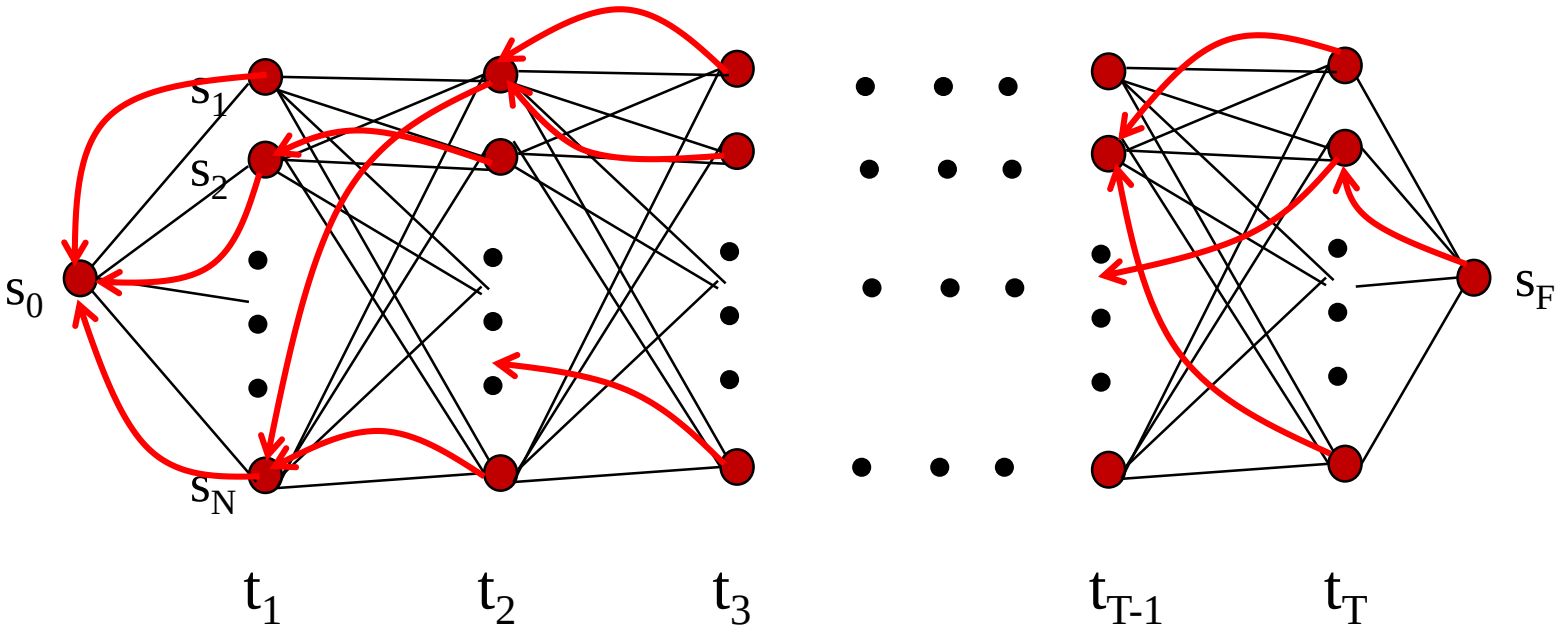
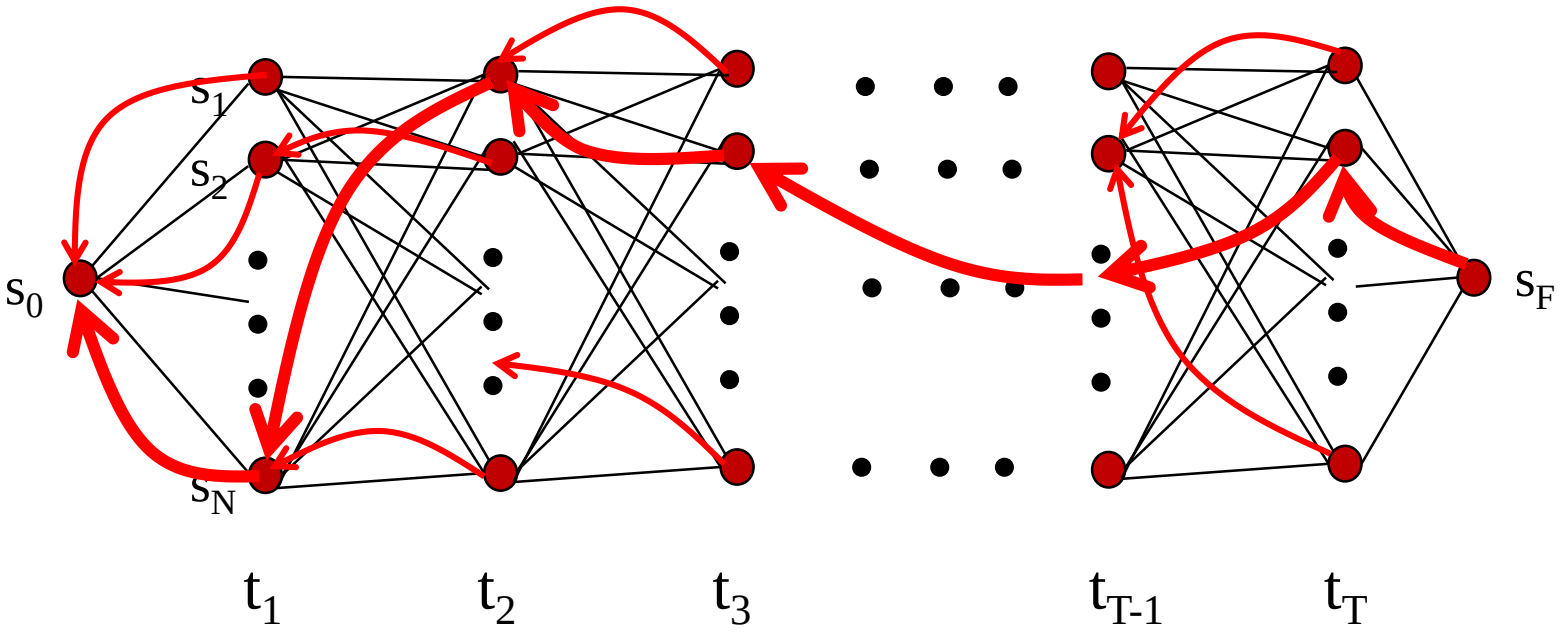


Figure A.10 The Viterbi backtrace. As we extend each path to a new state account for the next observation we keep a backpointer (shown with broken lines) to the best path that led us to this state.

Viterbi Backpointers



Viterbi Backtrace



Most likely Sequence: $s_0\ s_N\ s_1\ s_2\ \ldots s_2\ s_F$

The 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 $backpointer[q_F,T]$

Viterbi algorithm

- Create a table V with N+2 rows and T columns:
  - N – the number of states/tags
  - T – the length of the sequence/sentence
- Initialize the first column
- For each tag t in the tagset compute:

$$V[t,1] = P(t|start)P(w_1|t)$$

- For each column j = 2 to T in the table V:
  - For each tag t in the tagset compute:

$$V[t,j] = \max_{t'} V[t',j-1]P(t|t')P(w_j|t)$$

Viterbi Example 1: POS Tagging

Example: I want to race.

	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

Figure 5.15 Tag transition probabilities (the a array, $p(t_i|t_{i-1})$) computed from the 87-tag Brown corpus without smoothing. The rows are labeled with the conditioning event; thus $P(PPSS|VB)$ is .0070. The symbol <s> is the start-of-sentence symbol.

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

Figure 5.16 Observation likelihoods (the b array) computed from the 87-tag Brown corpus without smoothing.

Viterbi Algorithm Example

Algorithm first creates N or four state columns.

- First column corresponds to the observation of the first word "**I**",
- the second to the second word "**want**",
- the third to the third word "**to**", and
- the fourth to the fourth word "**race**"

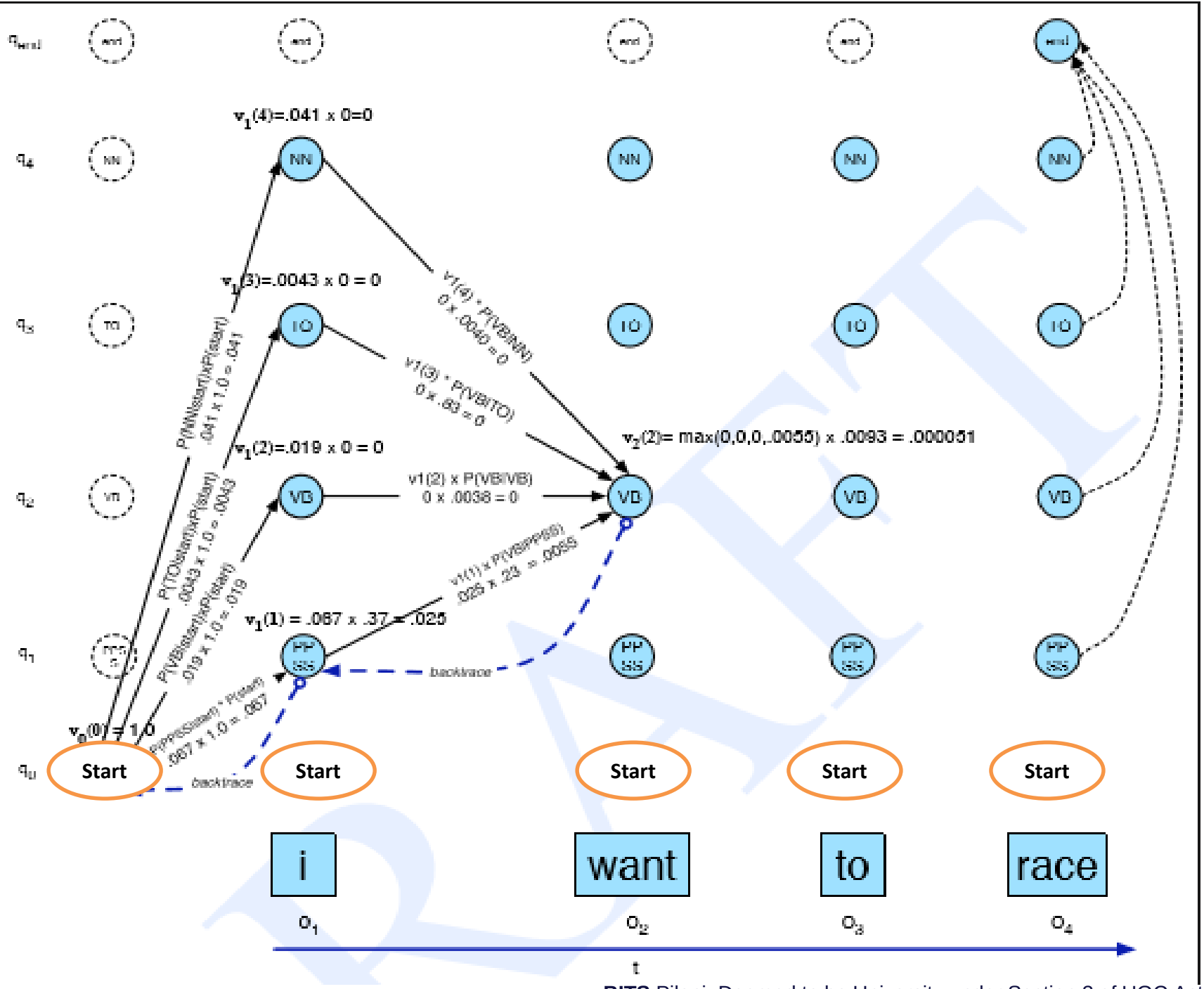
Begin by setting the Viterbi value in each cell to the product of the transition probability (into it from the state state) and the observation probability (of the first word)

Viterbi Algorithm Example

- For each state qj at time t, the value Viterbi [s,t] is computed by taking the maximum over the extensions of all the paths that lead to the current cell, following the following equation:

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

$v_{t-1}(i)$	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


$v_1(4)$=.041 x 0=0
$v_1(3)$=.0043 x 0 = 0
$v_1(2)$=.019 x 0 = 0
$v_1(1)$ = .067 x .37 = .025
$v_2(2)$= max(0,0,0,.0055) x .0093 = .000051
v1(4) * P(VB|NN)
0 x .0040 = 0
v1(3) * P(VB|TO)
0 x .83 = 0
v1(2) x P(VB|VB)
0 x .0038 = 0
P(NN|start)xP(start)
.041 x 1.0 = .041
P(TO|start)xP(start)
.0043 x 1.0 = .0043
P(VB|start)xP(start)
.019 x 1.0 = .019
P(PPSS|start) * P(start)
.067 x 1.0 = .067
v1(1) x P(VB|PPSS)
.025 x .23 = .0055
backtrace
backtrace
$v_0(0)$ = 1.0
NN
TO
VB
PP SS
Start
Start
Start
Start
Start
i
want
to
race
o_1
o_2
o_3
o_4
t

Example 3 : POS Tagging

Transition matrix: $P(t_i|t_{i-1})$

	NOUN	Verb	Det	Prep	ADV	STOP
<S>	.3	.1	.3	.2	.1	0
Noun	.2	.4	.01	.3	.04	.05
Verb	.3	.05	.3	.2	.1	.05
Det	.9	.01	.01	.01	.07	0
Prep	.4	.05	.4	.1	.05	0
Adv	.1	.5	.1	.1	.1	.1

	w1-=the	w2=doctor	w3=is	w4=in	STOP
Noun	0				
Verb	0				
Det	.21				
Prep	0				
Adv	0				

Emission matrix: $P(w_i|t_i)$

	a	cat	doctor	in	is	the	very
Noun	0	.5	.4	0	0.1	0	0
Verb	0	0	.1	0	.9	0	0
Det	.3	0	0	0	0	.7	0
Prep	0	0	0	1.0	0	0	0
Adv	0	0	0	.1	0	0	.9

$$V[t,1] = P(t|start)P(w_1|t)$$

V(Noun,the) = P(Noun|<S>)P(the|Noun) = .3 X 0=0
V(Verb,the) = P(Verb|<S>)P(the|Verb) = .1 X 0=0
V(Det,the) = P(Det|<S>)P(the|Det) = .3 X .7=.21
V(Prep,the) = P(Prep|<S>)P(the|Prep) = .2 X .0=0
V(Adv,the) = P(Adv|<S>)P(the|Adv) = .2 X .0=0

Example (Contd..)

innovate achieve lead

$V(\text{Noun, doctor}) = \max_{t'} V(t',\text{the})XP(\text{Noun}|t')X\ P(\text{doctor}|\text{Noun})$

$= \max \{0, 0, .21\ (.9\ X.4), 0, 0\} = .0756$

$V(\text{Verb, doctor}) = \max_{t'} V(t',\text{the})XP(\text{Verb}|t')X\ P(\text{doctor}|\text{Verb})$

$= \max \{0, 0, .21(.01X.1), 0, 0\} = .00021$

	w1-=the	w2=doctor	w3=is	w4=in	STOP
Noun	0	.0756			
Verb	0	.00021			
Det	.21	0			
Prep	0	0			
Adv	0	0			

Backtracking the Viterbi Matrix

innovate achieve lead

	w1-=the	w2=doctor	w3=is	w4=in	STOP
Noun	0	.0756	.001512	0	
Verb	0	.00021	.027216	0	
Det	.21	0	0	0	.005443
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	

Det Noun Verb Prep

Bidirectionality

- One problem with the HMM models as presented is that they are exclusively run left-to-right.
- Viterbi algorithm still allows present decisions to be influenced indirectly by future decisions

Bidirectionality

- Any sequence model can be turned into a bidirectional model by using multiple passes.
- For example, the first pass would use only part-of-speech features from already-disambiguated words on the left. In the second pass, tags for all words, including those on the right, can be used.
- Alternately, the tagger can be run twice, once left-to-right and once right-to-left.
- In Viterbi decoding, the classifier chooses the higher scoring of the two sequences (left-to-right or right-to-left).
- Modern taggers are generally run bidirectionally.

Issues with HMM

- Unknown Words
  - We do not have the probabilities
    - Use smoothing
- Limited Context
  - Use trigrams/ 4 grams etc.
  - Sparsity

Maximum Entropy Markov Model

- Identify heterogeneous set of features
  - tag given, the previous tag or the word given, the tag you can use, any other features which may contribute to the choice of part of speech tags.
- Turn logistic regression into a discriminative sequence model simply by running it on successive words, using the class assigned to the prior word as a feature in the classification of the next word.
- When we apply logistic regression in this way, it's called maximum entropy Markov model or MEMM

Maximum Entropy Markov Model

- Let the sequence of words be W = w_1^n and the sequence of tags T = t_1^n
- In an HMM to compute the best tag sequence that maximizes P(T|W) we rely on Bayes' rule and the likelihood P(W|T):

$$\begin{aligned}\hat{T} &= \underset{T}{\operatorname{argmax}} P(T|W) \\ &= \underset{T}{\operatorname{argmax}} P(W|T)P(T) \\ &= \underset{T}{\operatorname{argmax}} \prod_i P(word_i|tag_i) \prod_i P(tag_i|tag_{i-1})\end{aligned}$$

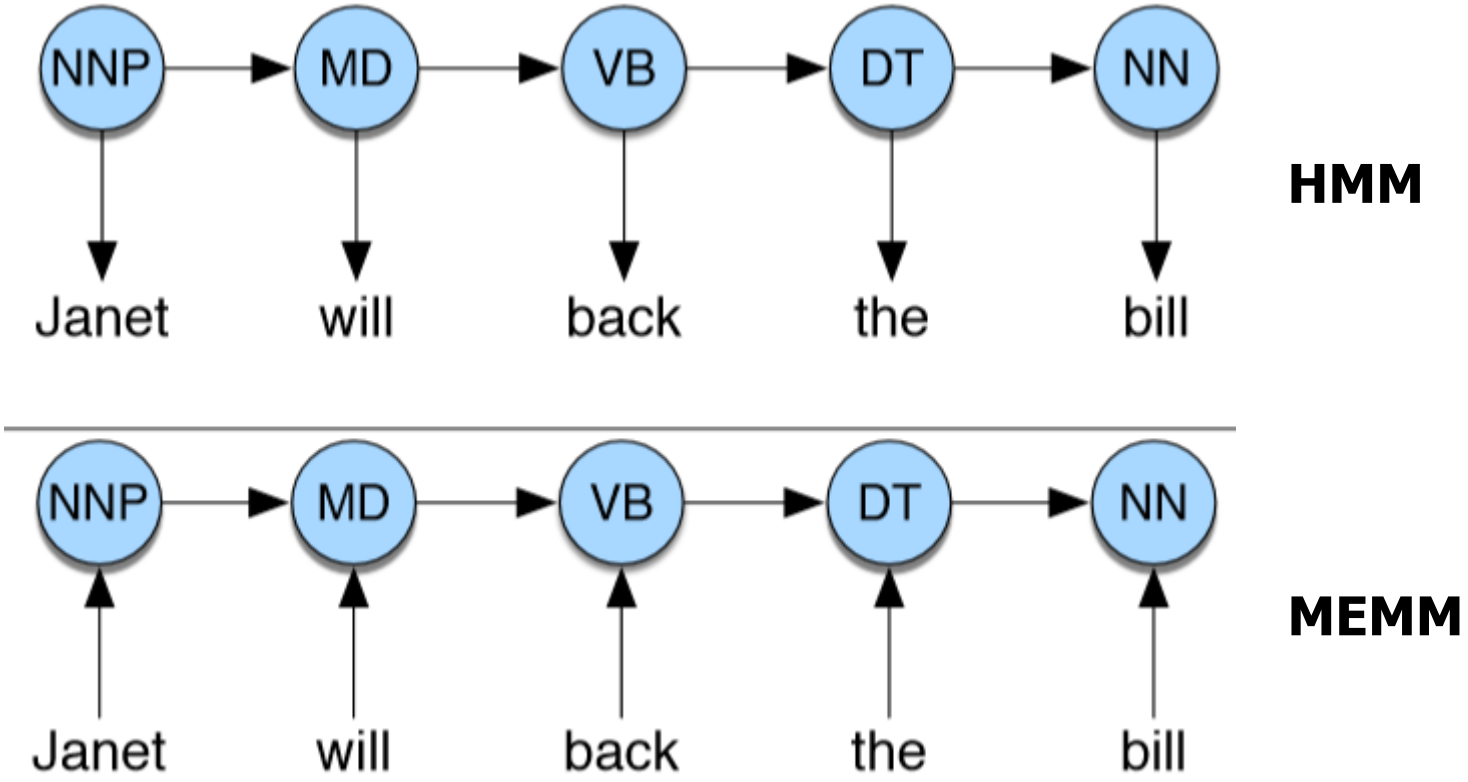
Maximum Entropy Markov Model

- In an MEMM, by contrast, we compute the posterior P(T|W) directly, training it to discriminate among the possible tag sequences:

$$\begin{aligned}\hat{T} &= \operatorname*{argmax}_T P(T|W) \\ &= \operatorname*{argmax}_T \prod_i P(t_i|w_i, t_{i-1})\end{aligned}$$

Maximum Entropy Markov Model

- Consider tagging just one word. A multinomial logistic regression classifier could compute the single probability P(ti|wi,ti−1) in a different way than an HMM
- HMMs compute likelihood (observation word conditioned on tags) but MEMMs compute posterior (tags conditioned on observation words).



Maximum Entropy Markov Model

- Reason to use a discriminative sequence model is that it's easier to incorporate a lot of features

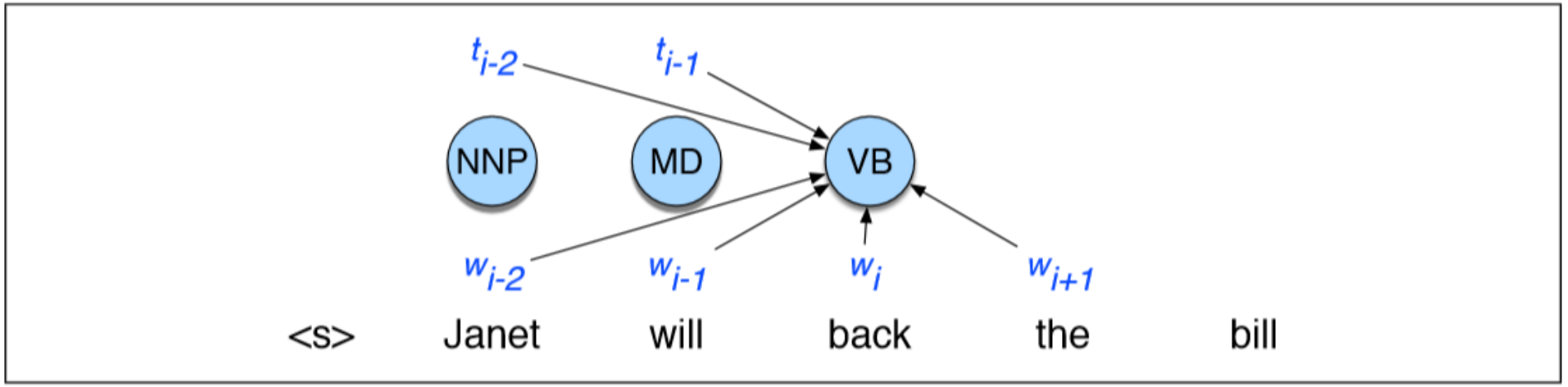


Figure 8.13 An MEMM for part-of-speech tagging showing the ability to condition on more features.

MEMM

- Janet/NNP will/MD back/VB the/DT bill/NN, when wi is the word back, would generate the following features

$$
\begin{aligned}
&t_i = \text{VB and } w_{i-2} = \text{Janet}\\
&t_i = \text{VB and } w_{i-1} = \text{will}\\
&t_i = \text{VB and } w_i = \text{back}\\
&t_i = \text{VB and } w_{i+1} = \text{the}\\
&t_i = \text{VB and } w_{i+2} = \text{bill}\\
&t_i = \text{VB and } t_{i-1} = \text{MD}\\
&t_i = \text{VB and } t_{i-1} = \text{MD and } t_{i-2} = \text{NNP}\\
&t_i = \text{VB and } w_i = \text{back and } w_{i+1} = \text{the}
\end{aligned}
$$

Decoding and Training MEMMs

- The most likely sequence of tags is then computed by combining these features of the input word w_i, its neighbors within l words w^{i+l}_{i-l}, and the previous k tags t^{i-1}_{i-k} as follows (using θ to refer to feature weights instead of w to avoid the confusion with w meaning words):

$$\begin{aligned}
\hat{T} &= \operatorname*{argmax}_T P(T|W) \\
&= \operatorname*{argmax}_T \prod_i P(t_i|w_{i-l}^{i+l}, t_{i-k}^{i-1}) \\
&= \operatorname*{argmax}_T \prod_i \frac{\exp\left(\sum_j \theta_j f_j(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}{\sum_{t' \in \text{tagset}} \exp\left(\sum_j \theta_j f_j(t', w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}
\end{aligned}$$

How to decode to find this optimal tag sequence ˆT?

- Simplest way to turn logistic regression into a sequence model is to build a local classifier that classifies each word left to right, making a hard classification on the first word in the sentence, then a hard decision on the second word, and so on.
- This is called a greedy decoding algorithm

function GREEDY SEQUENCE DECODING(words W, model P) **returns** tag sequence T

for $i = 1$ **to** *length(W)*

$$\hat{t}_i = \underset{t' \in T}{\text{argmax}}\ P(t' \mid w_{i-l}^{i+l}, t_{i-k}^{i-1})$$

Figure 8.14 In greedy decoding we simply run the classifier on each token, left to right, each time making a hard decision about which is the best tag.

Issue with greedy algorithm

- The problem with the greedy algorithm is that by making a hard decision on each word before moving on to the next word, the classifier can't use evidence from future decisions.
- Although the greedy algorithm is very fast, and occasionally has sufficient accuracy to be useful, in general the hard decision causes too great a drop in performance, and we don't use it.
- MEMM with the Viterbi algorithm just as with the HMM, Viterbi finding the sequence of part-of-speech tags that is optimal for the whole sentence

MEMM with Viterbi algorithm

- Finding the sequence of part-of-speech tags that is optimal for the whole sentence. Viterbi value of time t for state j

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

- In HMM

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, P(s_j|s_i)\, P(o_t|s_j) \quad 1 \le j \le N, 1 < t \le T$$

- In MEMM

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, P(s_j|s_i, o_t) \quad 1 \le j \le N, 1 < t \le T$$

Learning MEMM

- Learning in MEMMs relies on the same supervised learning algorithms we presented for logistic regression.
- Given a sequence of observations, feature functions, and corresponding hidden states, we use gradient descent to train the weights to maximize the log-likelihood of the training corpus.

References

NLTK :: nltk.tag package

NLTK POS Tagging – Python Examples - Python Examples

https://www.coursera.org/lecture/probabilistic-models-in-nlp/part-of-speech-tagging-VbnrA

https://www.coursera.org/lecture/text-mining-analytics/3-5-how-to-do-ner-and-pos-tagging-yQEYw

https://github.com/rajesh-iiith/POS-Tagging-and-CYK-Parsing-for-Indian-Languages/tree/master/data

https://marcossilva.github.io/en/2020/09/07/coursera-nlp-module-2-week-2.html

https://www.intechopen.com/online-first/88505

Dr. Chetana Gavankar has over 26 years of Teaching, Research and Industry experience. She has published papers in peer reviewed international conferences and journals. She is also reviewer for multiple conferences and journals. She has worked on different projects with multiple industries and received awards for her research work . Her areas of research interests include Natural Language Processing, Information Retrieval, Web Mining and Semantic Web, Ontology, Big Data Analytics, Machine learning, Deep learning and Artificial Intelligence.

Thank you!!