自然语言处理

week-3

凌震华

2024年3月14日

Part of speech tagging (词类标注)

- Parts of speech (词类)
- Tag sets (词类标记集)
- Rule-based tagging (self-study)
- Statistical tagging
  - Simple most-frequent-tag baseline
- HMM (隐马尔科夫模型) tagging

Parts of Speech

- ~8 traditional parts of speech
  - Noun 名词, verb 动词, adjective 形容词, adverb 副词, preposition 介词, pronoun 代词, conjunction 连接词, determiner 限定词, etc
  - Called: parts-of-speech, lexical category, word classes, morphological classes, lexical tags, POS
  - Lots of debate in linguistics about the number, nature, and universality of these
    - We'll completely ignore this debate.

POS examples

• 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 example

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

Example Tags

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	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, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	*#*
PDT	predeterminer	*all, both*	“	left quote	‘ or “
POS	possessive ending	*'s*	”	right quote	’ or ”
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	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*			

Figure 10.1 Penn Treebank part-of-speech tags (including punctuation).

POS Tagging

- Words often have more than one POS: *back* *(see previous slide for tags)*
  - The *back* door = JJ (adj)
  - On my *back* = NN
  - Win the voters *back* = RB (adverb)
  - Promised to *back* the bill = VB (base verb)

- The POS tagging problem is to determine the POS tag for a particular instance of a word.

These examples from Dekang Lin

How hard is POS tagging? Measuring ambiguity

	Original 87-tag corpus	Treebank 45-tag corpus
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 (*well*, *beat*)	32
7 tags	2 (*still*, *down*)	6 (*well*, *set*, *round*, *open*, *fit*, *down*)
8 tags		4 (*'s*, *half*, *back*, *a*)
9 tags		3 (*that*, *more*, *in*)

Two methods for POS tagging

1. Rule-based tagging

2. Stochastic (=Probabilistic) tagging
   – HMM (Hidden Markov Model 隐马尔科夫模型) tagging

Simple baselines

- Default tagger (based on most frequent)
- Lookup tagger (e.g., for 100 most frequent words, used in conjunction with default tagger)
- Regular expression tagger (RE applied to word)

Hidden Markov Model Tagging

- Using an HMM to do POS tagging
- Is a special case of Bayesian inference (贝叶斯推理)
  - Foundational work in computational linguistics
  - Bledsoe 1959: OCR (光学字符识别)
  - Mosteller and Wallace 1964: authorship identification
- It is also related to the "noisy channel" model that's the basis for ASR (自动语音识别), OCR and MT (机器翻译)

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 which 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 w1…wn.

Road to HMMs

- We want, out of all sequences of n tags $t_1 \ldots t_n$ the single tag sequence such that $P(t_1 \ldots t_n | w_1 \ldots w_n)$ 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"
- $\text{argmax}_x f(x)$ means "the x such that f(x) is maximized"

Road to HMMs

- This equation is guaranteed to 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 classification:
  – Use Bayes rule to transform into a set of other probabilities that are easier to compute

Using Bayes Rule

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

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

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

Likelihood and Prior

$$\hat{t}_1^n = \operatorname*{argmax}_{t_1^n} \overbrace{P(w_1^n|t_1^n)}^{\substack{\text{似然度} \\ \text{likelihood}}} \ \overbrace{P(t_1^n)}^{\substack{\text{先验概率} \\ \text{prior}}}$$

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

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

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

Two Sets 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 Sets of Probabilities (2)

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

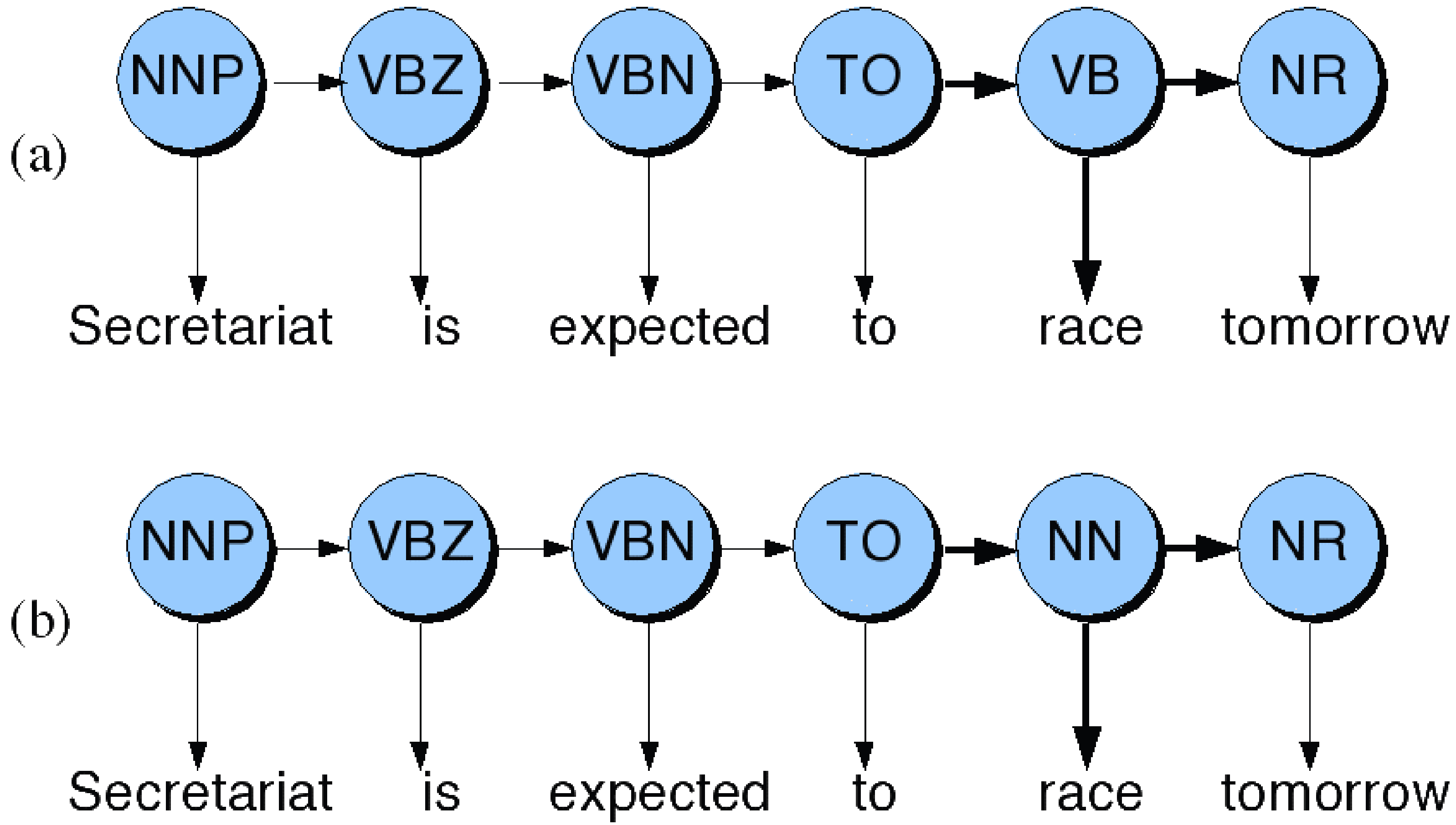
$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

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

An 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”



Example

- $P(NN|TO) = .00047$
- $P(VB|TO) = .83$
- $P(race|NN) = .00057$
- $P(race|VB) = .00012$
- $P(NR|VB) = .0027$
- $P(NR|NN) = .0012$
- $P(VB|TO)P(NR|VB)P(race|VB) = .00000027$
- $P(NN|TO)P(NR|NN)P(race|NN)=.00000000032$
- So we (correctly) choose the verb reading

Hidden Markov Models

- What we’ve described with these two kinds of probabilities is a Hidden Markov Model

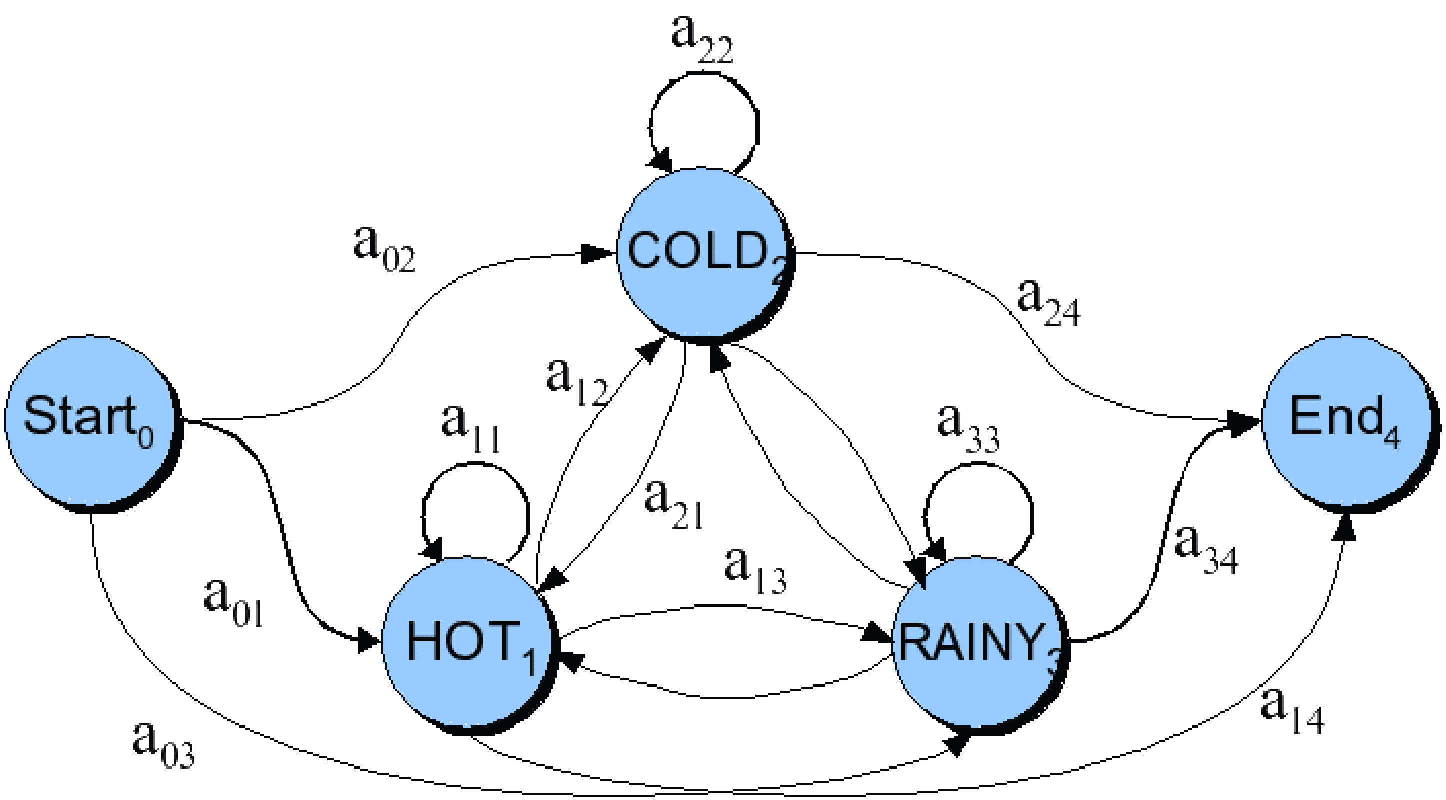
- Let’s just spend a bit of time tying this into the model

- First some definitions.

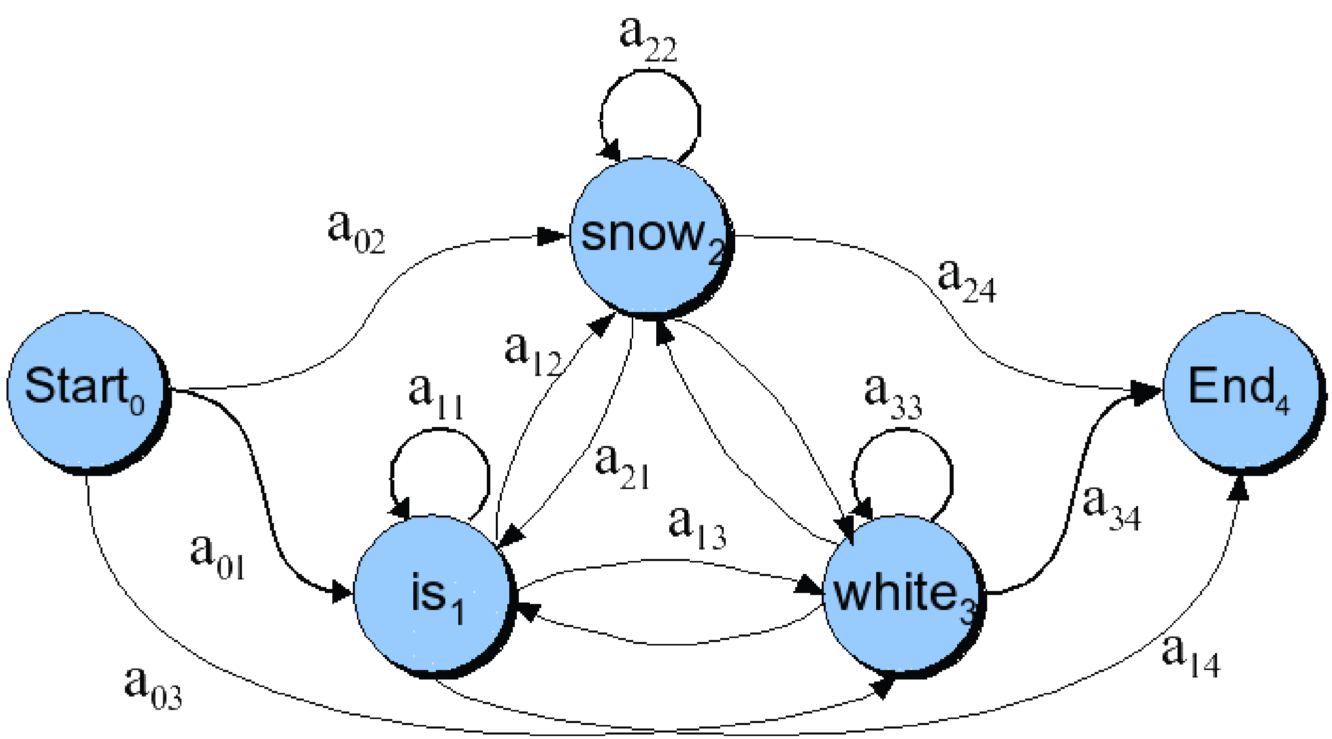
Definitions

- A weighted finite-state automaton (加权有限状态自动机) adds probabilities to the arcs
  - The sum of the probabilities leaving any state must sum to one
- A Markov chain is a special case in which the input sequence uniquely determines which states the automaton will go through
- Markov chains can’ t represent inherently ambiguous problems
  - Useful for assigning probabilities to unambiguous sequences

Markov chain for weather



Markov chain for words



Markov chain = "First-order Observable Markov Model"

- A set of states (状态)
  - $Q = q_1, q_2 \ldots q_{N;}$ the state at time t is q_t
- Transition probabilities (转移概率)
  - 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

- Current state only depends on previous state

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

Markov chain for weather

- What is the probability of 4 consecutive rainy days?
- Sequence is rainy-rainy-rainy-rainy
- i.e., 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 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, MA for summer of 2007
- But you find Jason Eisner's diary
- Which lists how many ice-creams Jason ate every date that summer
- Our job: figure out how hot it was

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states.
  - See **hot** weather: we' re in state **hot**
- But in part-of-speech tagging (and other things)
  - The output symbols are **words**
  - But the hidden states are **part-of-speech tags**
- So we need an extension!
- A Hidden Markov Model is an extension of a Markov chain in which the input symbols are not the same as the states.
- This means we don' t know which state we are in.

Hidden Markov Models

- States $Q = q_1, q_2 \ldots q_{N;}$ 状态
- Observations $O= o_1, o_2 \ldots o_{N;}$ 观测值
  – Each observation is a symbol from a vocabulary $V = \{v_1,v_2,\ldots v_V\}$
- Transition probabilities 转移概率
  – Transition probability matrix $A = \{a_{ij}\}$

$$a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \le i, j \le N$$

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

$$b_i(k) = P(X_t = o_k \mid q_t = i)$$

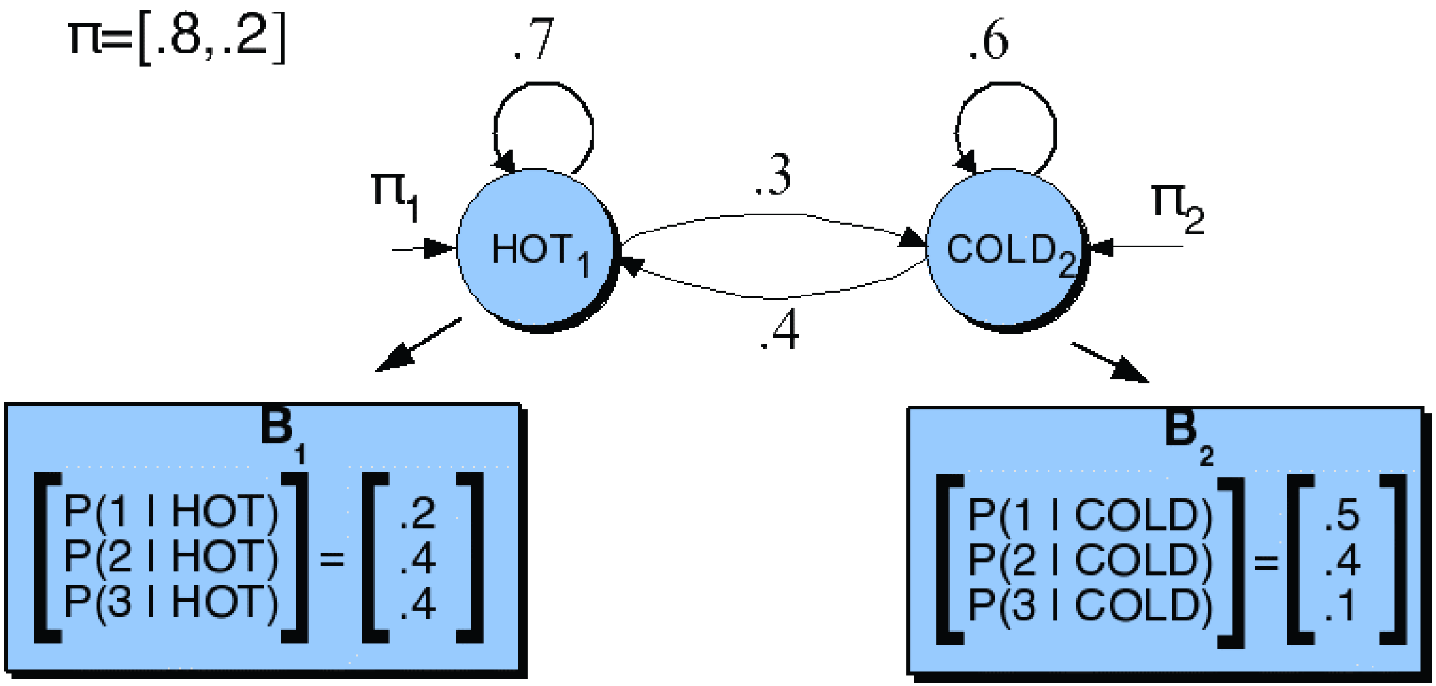
- Special initial probability vector π 初始概率

$$\pi_i = P(q_1 = i) \quad 1 \le i \le N$$

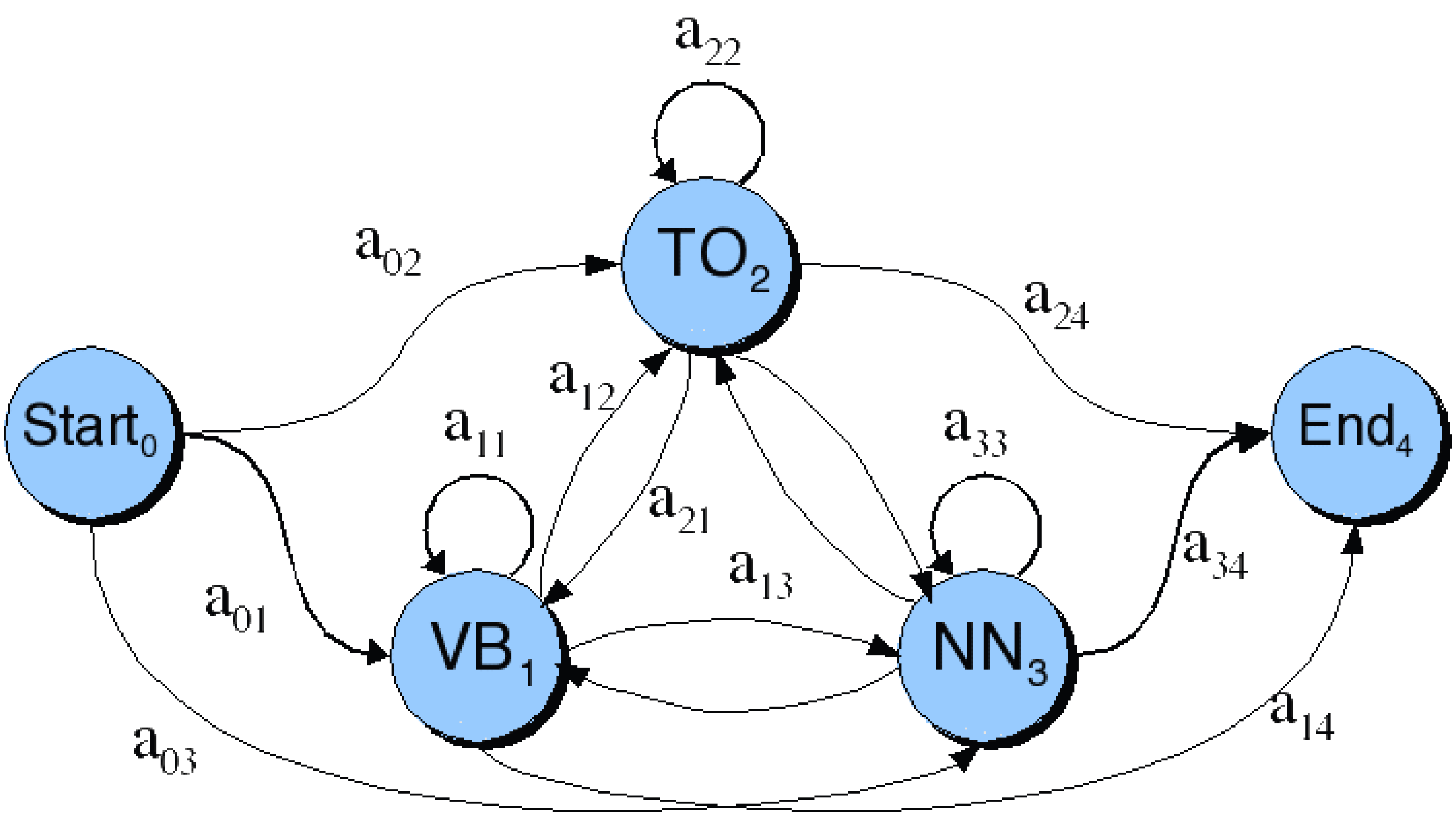
Eisner task

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

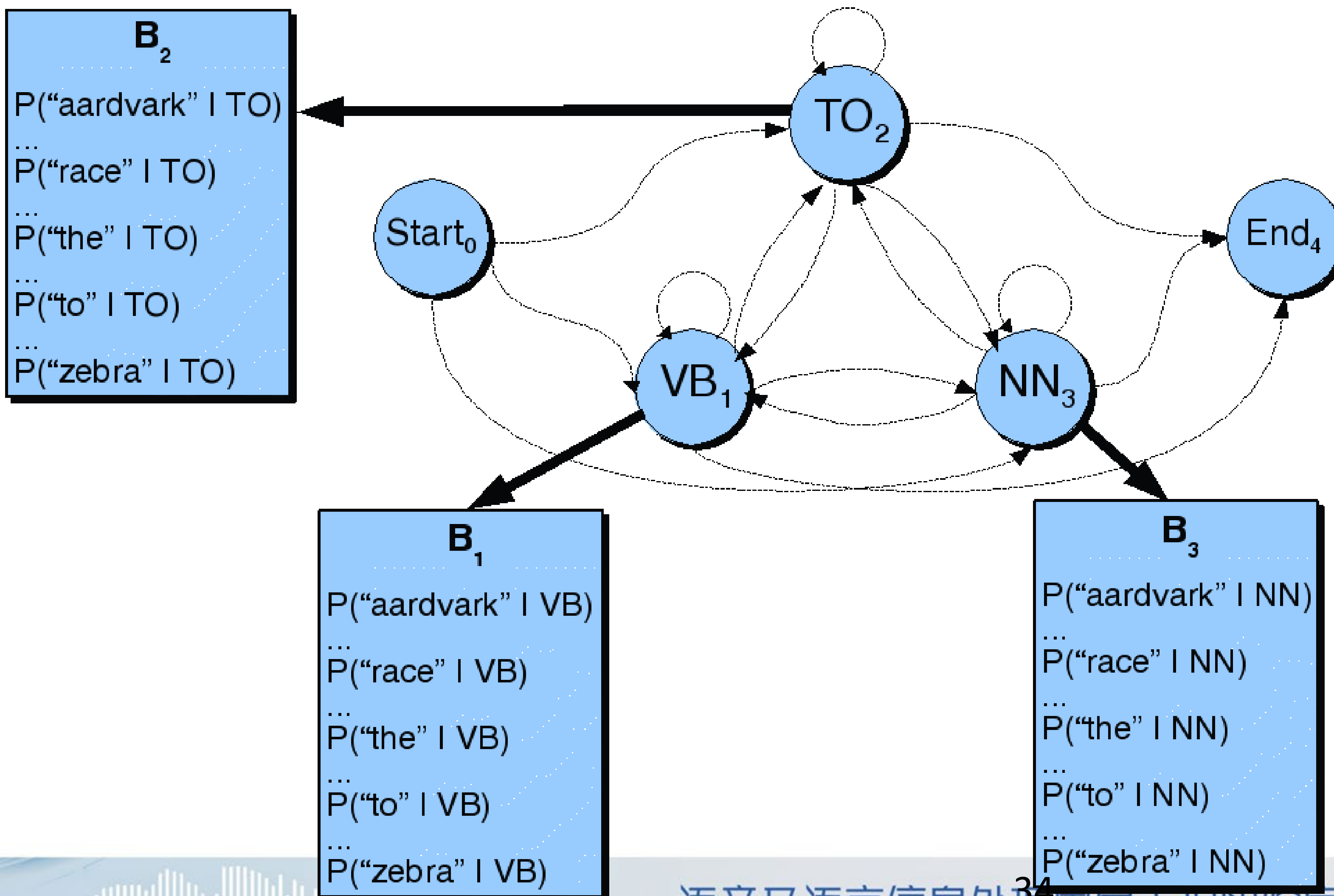
HMM for ice cream



Transitions between the hidden states of HMM, showing A probs



B observation likelihoods for POS HMM



The A matrix for the POS HMM

	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 4.15 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.

The B matrix for the POS HMM

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

Urns and Balls

- A genie has two urns filled with red and blue balls. The genie selects an urn and then draws a ball from it (and replaces it). The genie then selects either the same urn or the other one and then selects another ball…
  - The urns are hidden
  - The balls are observed

Urns and Balls

- Based on the results of a long series of draws…
  - Figure out the distribution of colors of balls in each urn
  - Figure out the genie’s preferences in going from one urn to the next

Urns and Balls

- Pi: Urn 1: 0.9; Urn 2: 0.1
- A

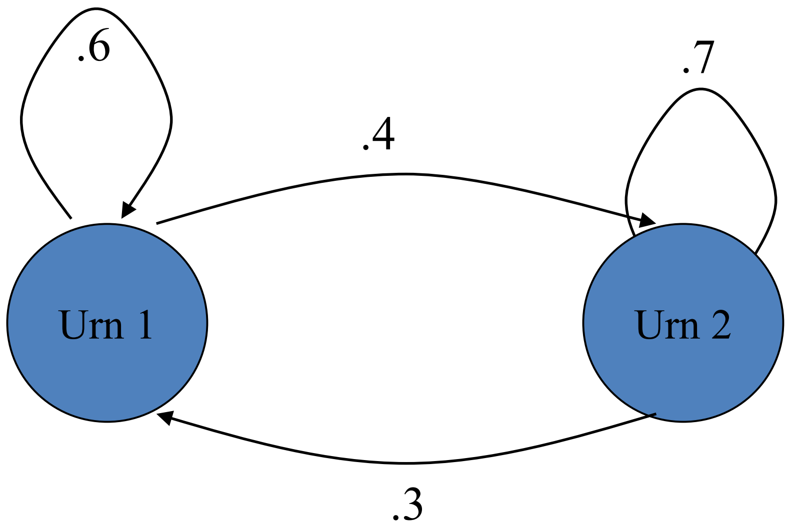
	Urn 1	Urn 2
Urn 1	0.6	0.4
Urn 2	0.3	0.7

- B

	Urn 1	Urn 2
Red	0.7	0.4
Blue	0.3	0.6

Urns and Balls

- Let’ s assume the input (observables) is Blue Blue Red (BBR)
- Since both urns contain
  red and blue balls
  any path through
  this machine
  could produce this output



Urns and Balls

Blue Blue Red

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

Viterbi: Says 111 is the most likely state sequence

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181
2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

Forward: P(BBR| model) = .0792

Σ

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

- Model training
  - What if I told you I lied about the numbers in the model (Priors,A,B). I just made them up.
  - Can I get better numbers just from the input sequence?

Urns and Balls

- Just count up and prorate (分派) the number of times a given transition is traversed while processing the observations inputs.

- Then use that count to re-estimate the transition probability for that transition

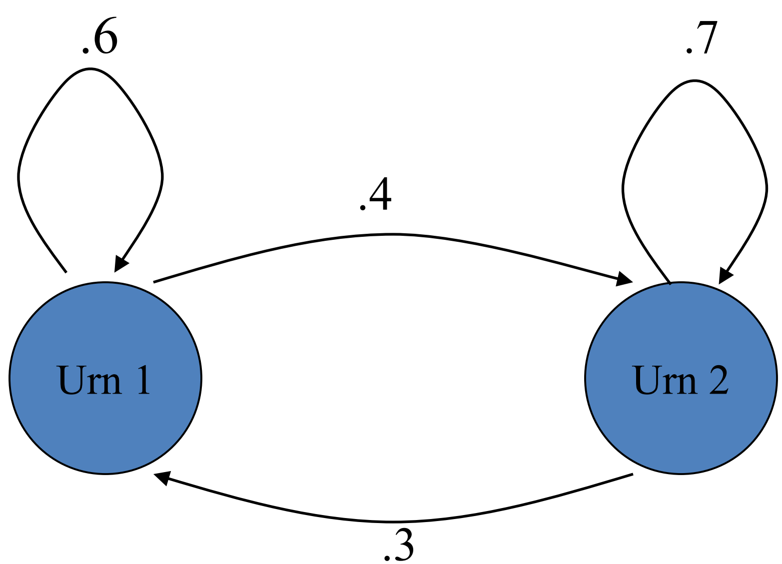
- See Page 50 (Equations 6.31 and 6.39 in textbook)

Urns and Balls

- But… we just saw that don’ t know the actual path the input took, its hidden!
  - So prorate the counts from all the possible paths based on the path probabilities the model gives you

- But you said the numbers were wrong
  - Doesn’ t matter; use the original numbers then replace the old ones with the new ones.

Urn Example



Let's re-estimate the Urn1->Urn2 transition and the Urn1->Urn1 transition (using Blue Blue Red as training data).

Urns and Balls

Blue Blue Red

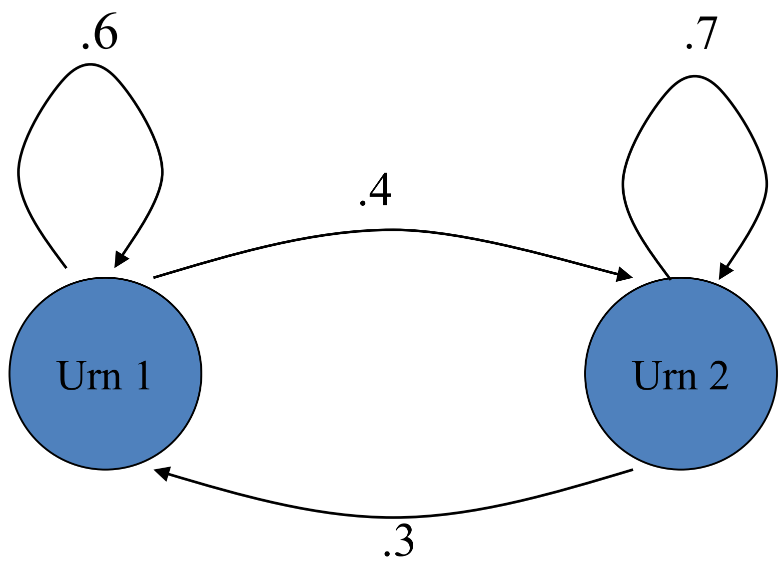
1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

- That’ s
  (.0077*1)+(.0136*1)+(.0181*1)+(.0020*1)
  = .0414
- Of course, that’ s not a probability, it needs to be divided by the probability of leaving Urn 1 total.
- There’ s only one other way out of Urn 1 (going back to urn1)
  – So let’ s reestimate Urn1-> Urn1

Urn Example



Let's re-estimate the Urn1->Urn1 transition

Urns and Balls

Blue Blue Red

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

- That’ s just
  (2*.0204)+(1*.0077)+(1*.0052) = .0537

- Again not what we need but we’ re closer… we just need to normalize using those two numbers.

Urns and Balls

- The 1->2 transition probability is
  .0414/(.0414+.0537) = 0.435

- The 1->1 transition probability is
  .0537/(.0414+.0537) = 0.565

- So in re-estimation the 1->2 transition went from .4 to .435 and the 1->1 transition went from .6 to .565

Maximum Entropy Markov Models

- MaxEnt (最大熵)，Multinomial logistic regression (多元逻辑回归)

$$p(c|x) = \frac{1}{Z}\exp(\sum_i w_i f_i)$$

- Discriminative (区分式) models based on features（特征)
- Features extracted locally and globally
  - eg: sentence length > 5, date in sentence, quotation in sentence
- Features usually Boolean
- MEMM: MaxEnt algorithm with Viterbi used

Sample features

w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i contains a number
w_i contains an upper-case letter
w_i contains a hyphen
w_i is all upper case
w_i's word shape
w_i's short word shape
w_i is upper case and has a digit and a dash (like *CFC-12*)
w_i is upper case and followed within 3 words by Co., Inc., etc.

HMM vs MEMM

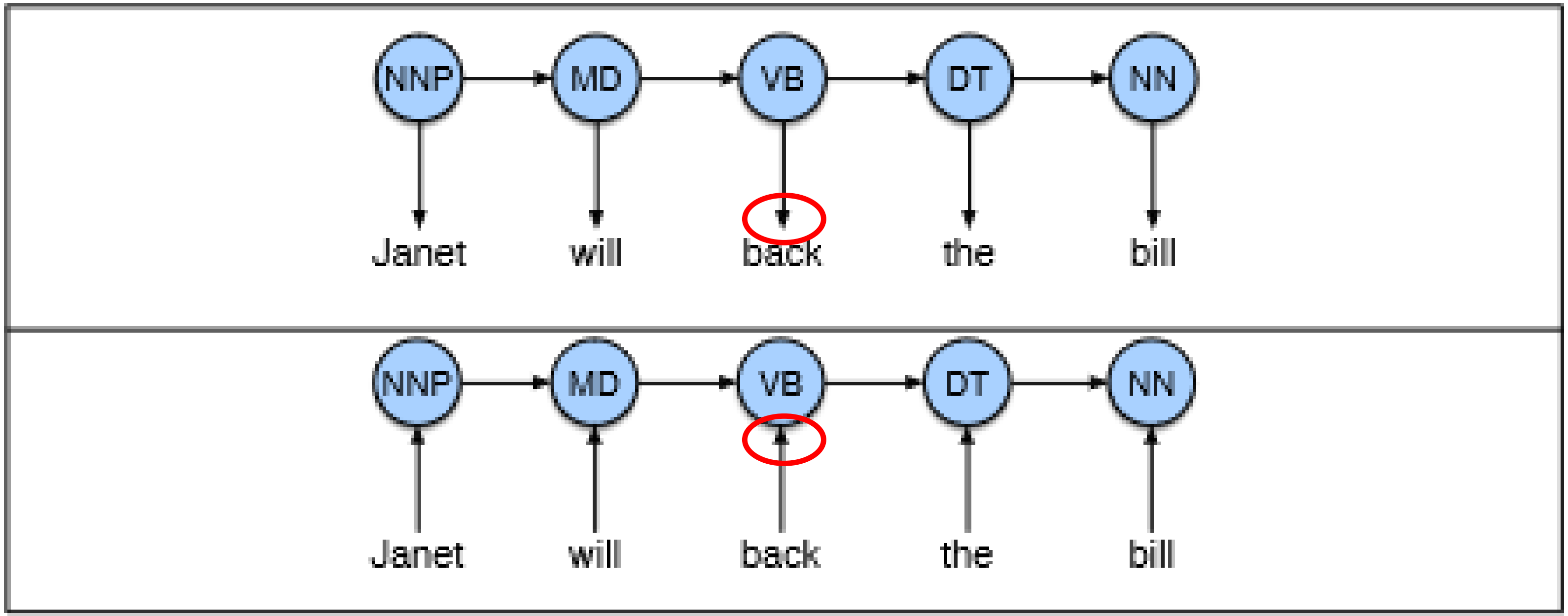


Figure 10.11 A schematic view of the HMM (top) and MEMM (bottom) representation of the probability computation for the correct sequence of tags for the *back* sentence. The HMM computes the likelihood of the observation given the hidden state, while the MEMM computes the posterior of each state, conditioned on the previous state and current observation.

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.

Error Analysis

- Look at a confusion matrix (混淆矩阵)

	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**	-

- See what errors are causing problems
  – Noun (NN) vs ProperNoun (NNP) vs Adj (JJ)
  – Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)

POS tags in Chinese

- More ambiguity (compared to English)
- Problems with unknown words
  - More common nouns/verbs (English tends to have proper nouns)
  - Radicals of characters (字根) used as features to disambiguate
- Neural models give the best performance

Reading Materials

- L.R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition”, Proceedings of the IEEE, vol. 77, no. 2, pp. 257-286, 1989