CSCI 5832
Natural Language Processing

Jim Martin

Lecture 9

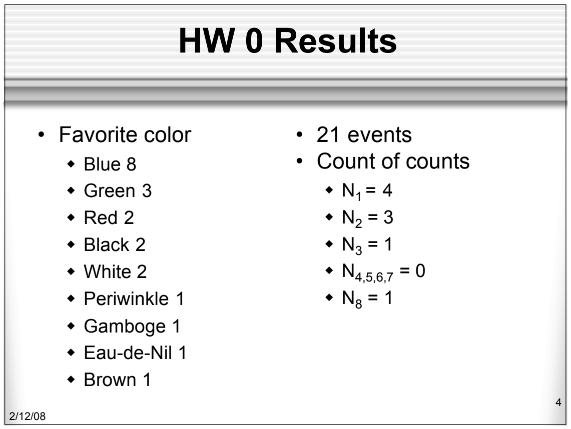
Today 2/12

- Review
  - GT example
- HMMs and Viterbi
  - POS tagging

Good-Turing Intuition

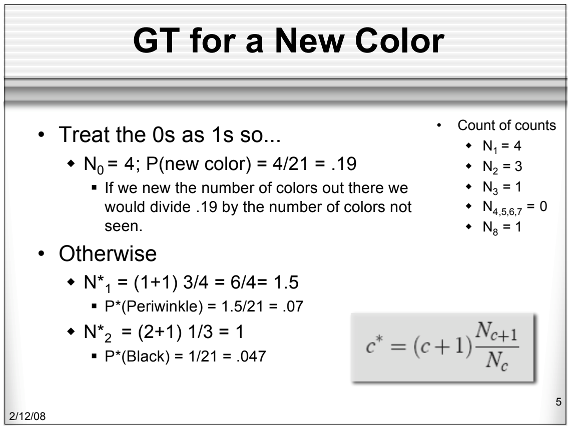
- Notation: N_x is the frequency-of-frequency-x
  - So N_{10}=1, N_1=3, etc
- To estimate counts/probs for unseen species
  - Use number of species (words) we've seen once
  - $c_0^* = c_1$ $p_0 = N_1/N$
- All other estimates are adjusted (down) to allow for increased probabilities for unseen

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$



HW 0 Results

- Favorite color
  - Blue 8
  - Green 3
  - Red 2
  - Black 2
  - White 2
  - Periwinkle 1
  - Gamboge 1
  - Eau-de-Nil 1
  - Brown 1
- 21 events
- Count of counts
  - $N_1 = 4$
  - $N_2 = 3$
  - $N_3 = 1$
  - $N_{4,5,6,7} = 0$
  - $N_8 = 1$

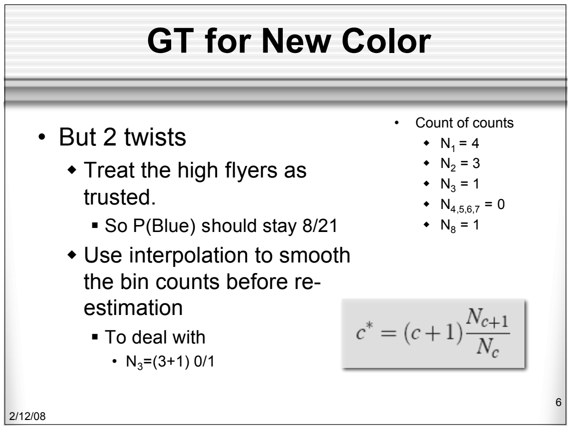


GT for a New Color

- Treat the 0s as 1s so...
  - $N_0 = 4$; P(new color) = 4/21 = .19
    - If we new the number of colors out there we would divide .19 by the number of colors not seen.
- Otherwise
  - N^*_1 = (1+1) 3/4 = 6/4= 1.5
    - P*(Periwinkle) = 1.5/21 = .07
  - N^*_2 = (2+1) 1/3 = 1
    - P*(Black) = 1/21 = .047

- Count of counts
  - $N_1 = 4$
  - $N_2 = 3$
  - $N_3 = 1$
  - $N_{4,5,6,7} = 0$
  - $N_8 = 1$

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

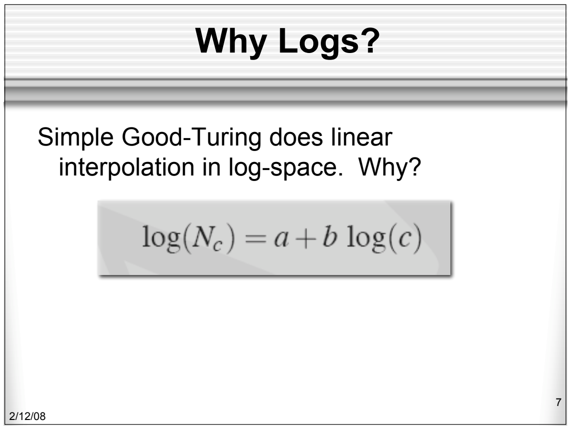


GT for New Color

- But 2 twists
  - Treat the high flyers as trusted.
    - So P(Blue) should stay 8/21
  - Use interpolation to smooth the bin counts before re-estimation
    - To deal with
      - N_3=(3+1) 0/1

- Count of counts
  - $N_1 = 4$
  - $N_2 = 3$
  - $N_3 = 1$
  - $N_{4,5,6,7} = 0$
  - $N_8 = 1$

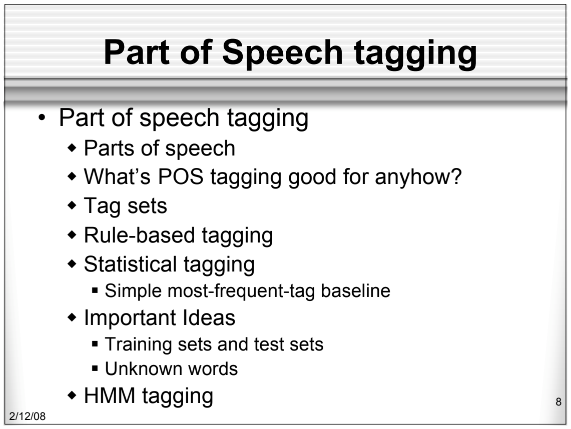
$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$



Why Logs?

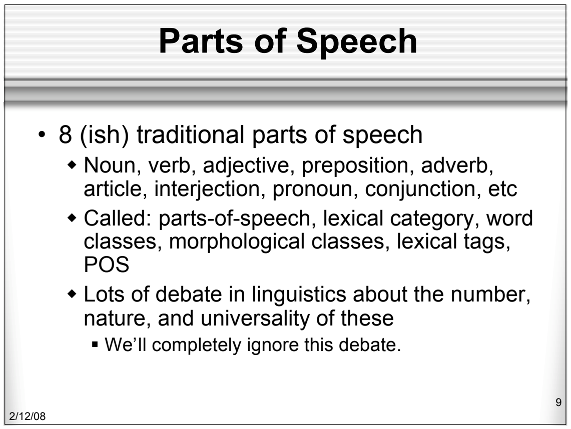
Simple Good-Turing does linear interpolation in log-space. Why?

$$\log(N_c) = a + b\ \log(c)$$



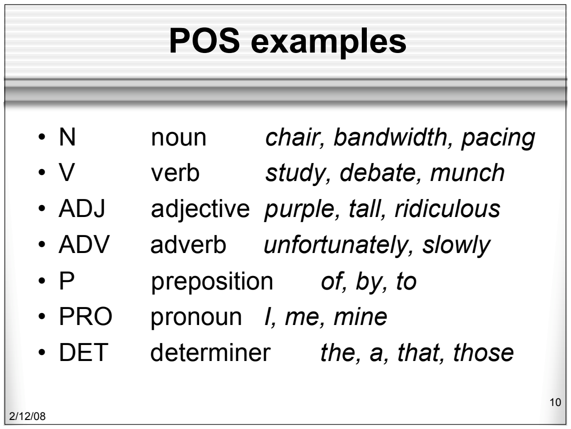
Part of Speech tagging

- Part of speech tagging
  - Parts of speech
  - What's POS tagging good for anyhow?
  - Tag sets
  - Rule-based tagging
  - Statistical tagging
    - Simple most-frequent-tag baseline
  - Important Ideas
    - Training sets and test sets
    - Unknown words
  - HMM tagging



Parts of Speech

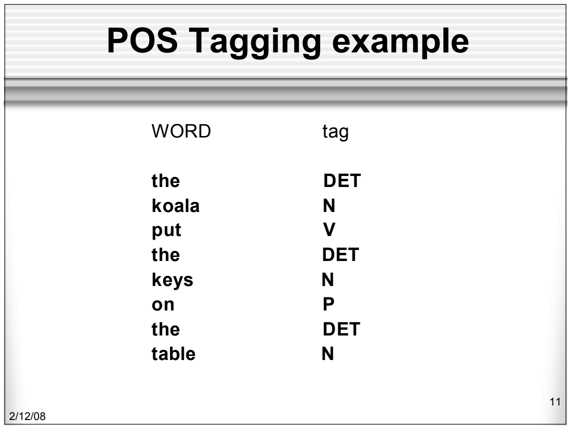
- 8 (ish) traditional parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc
  - Called: parts-of-speech, lexical category, word classes, morphological classes, lexical tags, POS
  - Lots of debate in linguistics about the number, nature, and 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*

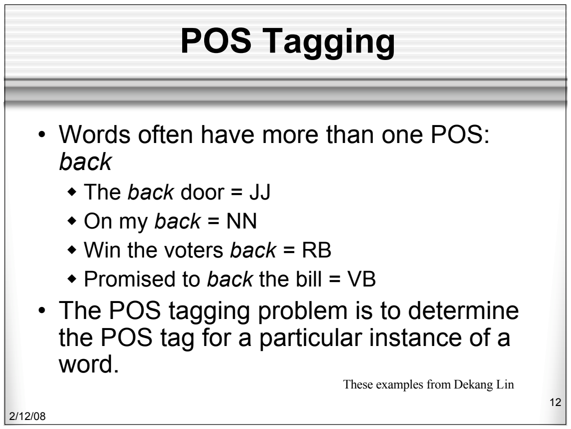
2/12/08 10



POS Tagging example

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

2/12/08 11

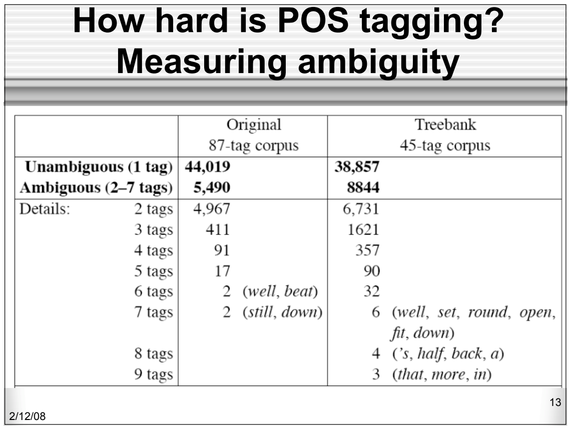


POS Tagging

- 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.

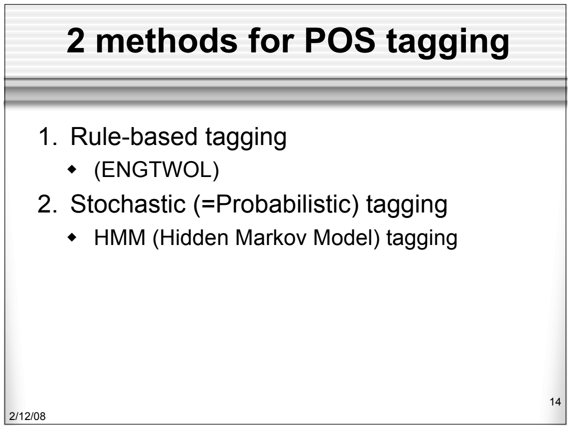
These examples from Dekang Lin

2/12/08 12



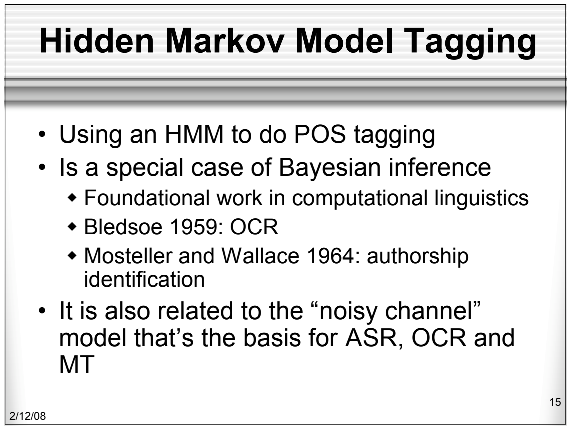
How hard is POS tagging? Measuring ambiguity

	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*)



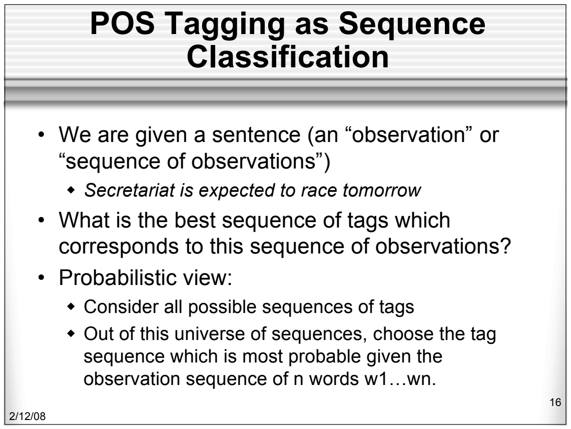
2 methods for POS tagging

1. Rule-based tagging
   - (ENGTWOL)
2. Stochastic (=Probabilistic) tagging
   - HMM (Hidden Markov Model) tagging



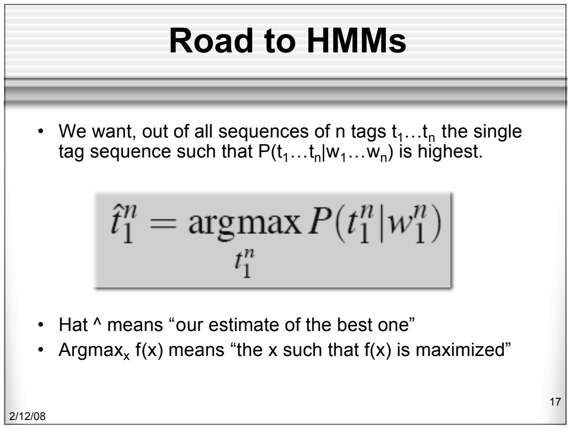
Hidden Markov Model Tagging

- Using an HMM to do POS tagging
- Is a special case of Bayesian inference
  - Foundational work in computational linguistics
  - Bledsoe 1959: OCR
  - Mosteller and Wallace 1964: authorship identification
- It is also related to the “noisy channel” model that’s the basis for ASR, OCR and MT



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.

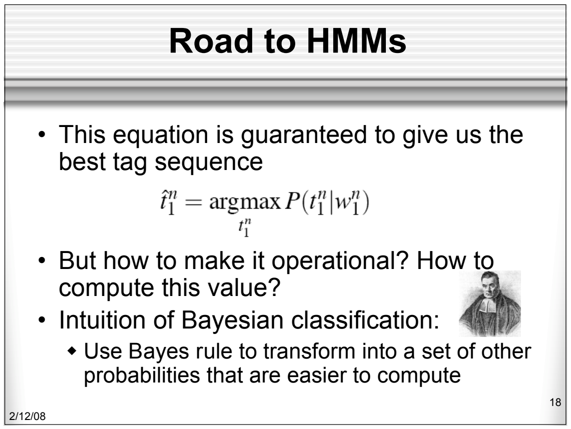


Road to HMMs

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

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

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

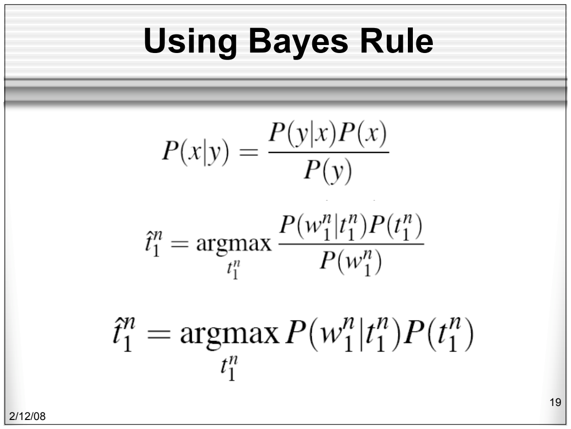


Road to HMMs

- 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

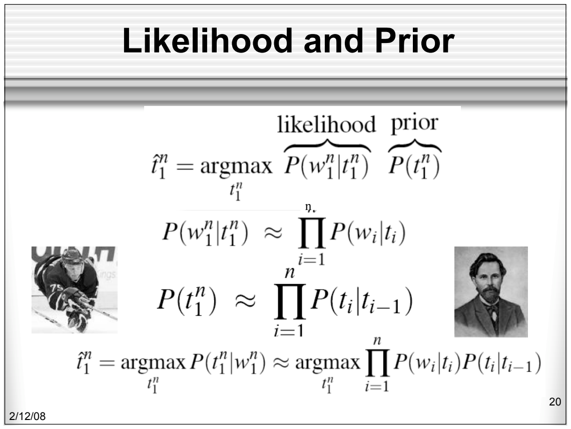


Using Bayes Rule

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

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

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



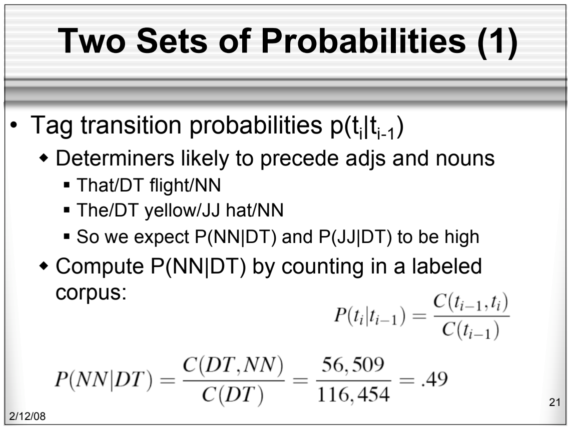
Likelihood and Prior

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

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

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

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



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)

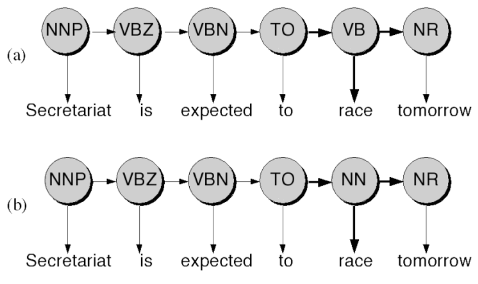
- Word likelihood probabilities $p(w_i|t_i)$
  - VBZ (3sg Pres verb) likely to be "is"
  - Compute P(is|VBZ) by counting in a labeled corpus: $P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$

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

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

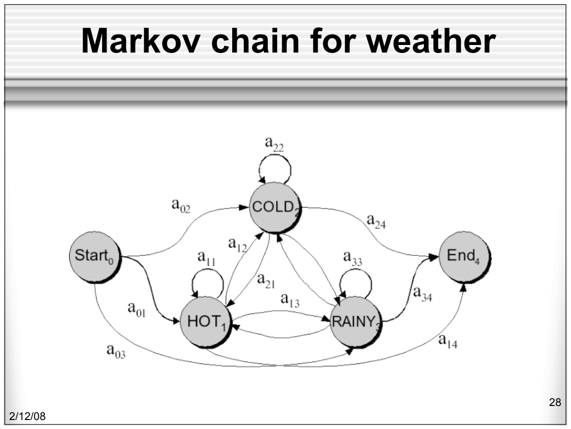
- P(NN|TO) = .00047
- P(VB|TO) = .83
- P(race|NN) = .00057
- P(race|VB) = .00012
- P(NR|VB) = .0027
- P(NR|NN) = .0012
- P(VB|TO)P(NR|VB)P(race|VB) = .00000027
- P(NN|TO)P(NR|NN)P(race|NN)=.00000000032
- So we (correctly) choose the verb reading,

Hidden Markov Models

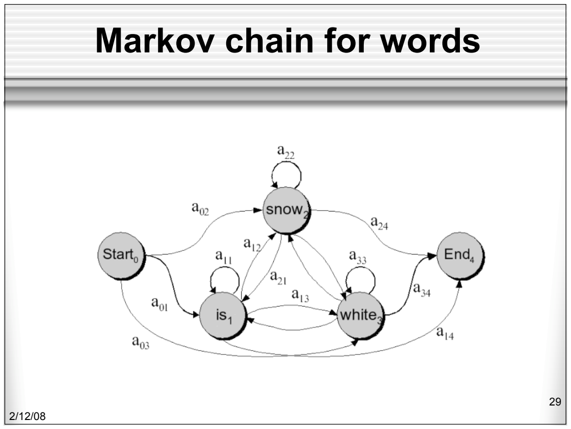
- What we've described with these two kinds of probabilities is a Hidden Markov Model
- Let's just spend a bit of time tying this into the model
- First some definitions.

Definitions

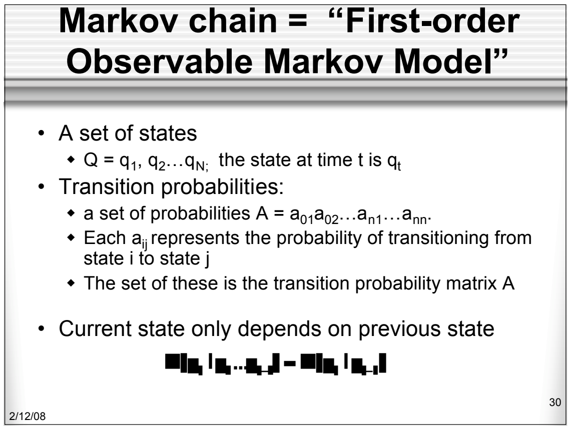
- A weighted finite-state automaton adds probabilities to the arcs
  - The sum of the probabilities leaving any arc must sum to one
- A Markov chain is a special case in which the input sequence uniquely determines which states the automaton will go through
- Markov chains can't represent inherently ambiguous problems
  - Useful for assigning probabilities to unambiguous sequences



Markov chain for weather



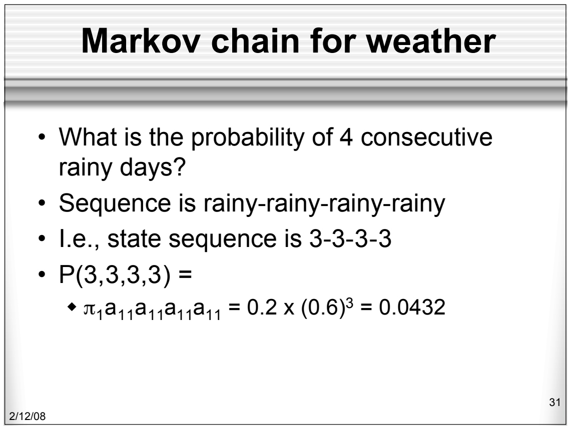
Markov chain for words



Markov chain = "First-order Observable Markov Model"

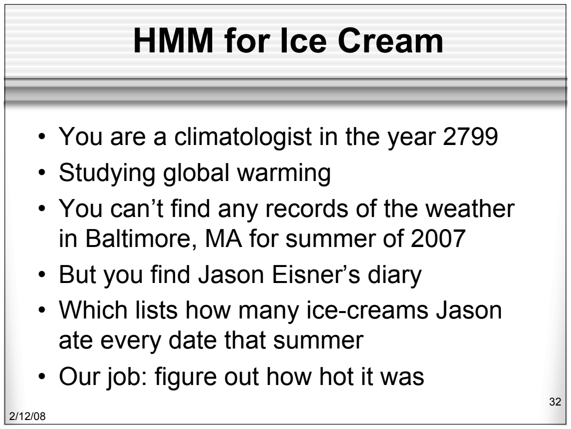
- A set of states
  - Q = $q_1, q_2 \ldots q_{N;}$ the state at time t is q_t
- Transition probabilities:
  - a set of probabilities A = $a_{01}a_{02} \ldots a_{n1} \ldots a_{nn}$.
  - Each a_{ij} represents the probability of transitioning from state i to state j
  - The set of these is the transition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

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



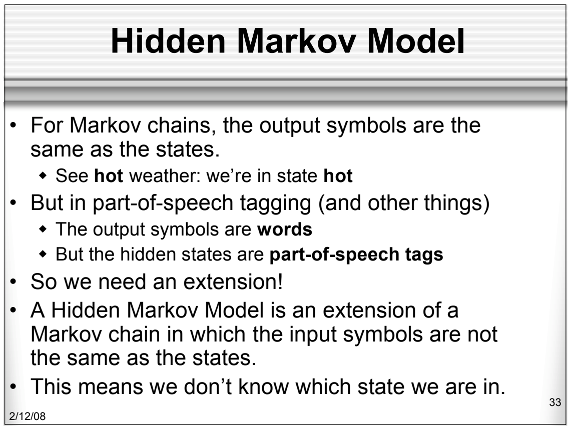
Markov chain for weather

- 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) =
  - $\pi_1 a_{11} a_{11} a_{11} a_{11}$ = 0.2 x $(0.6)^3$ = 0.0432



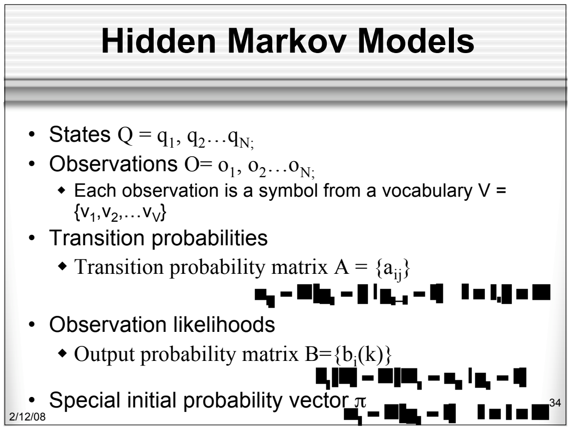
HMM for Ice Cream

- 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.



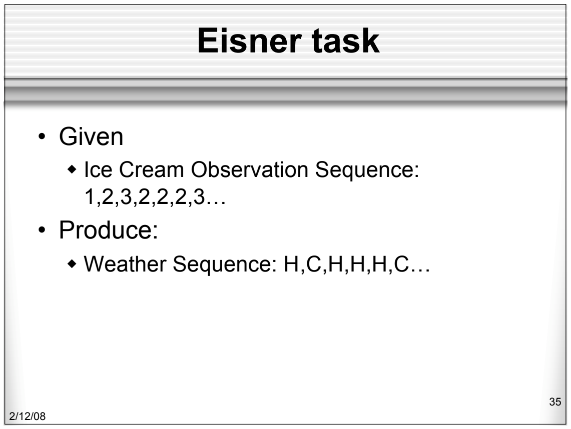
Hidden Markov Models

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

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

    [illegible]
- Special initial probability vector π [illegible]

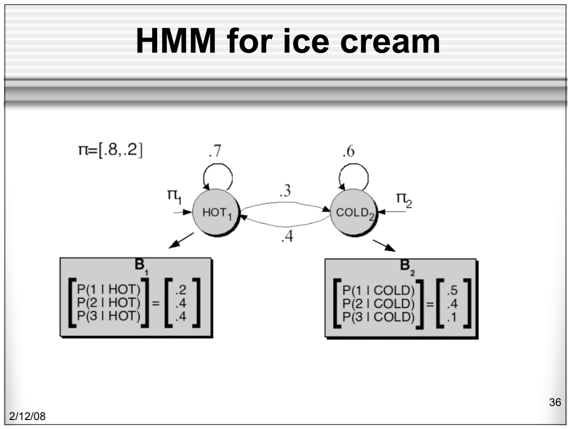
2/12/08 34



Eisner task

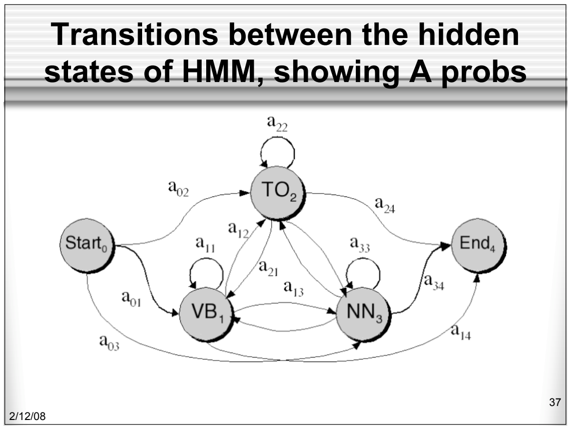
- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3…
- Produce:
  - Weather Sequence: H,C,H,H,H,C…

2/12/08 35



HMM for ice cream

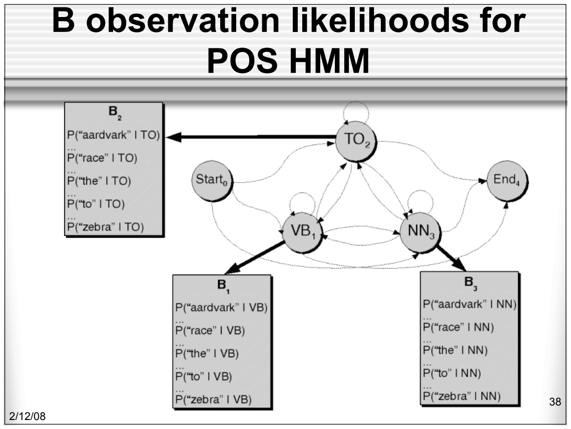
2/12/08 36



Transitions between the hidden states of HMM, showing A probs

2/12/08

37



B observation likelihoods for POS HMM

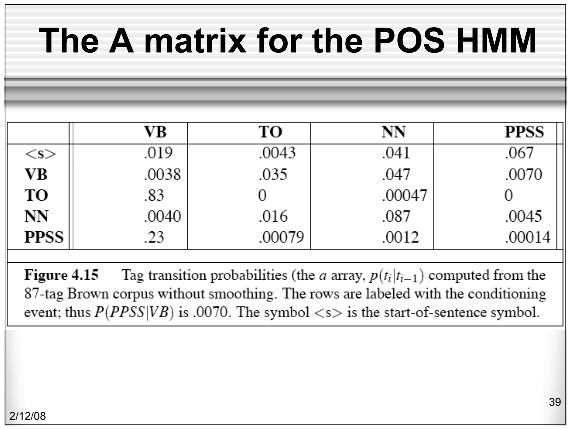
B_2
P("aardvark" | TO)
...
P("race" | TO)
...
P("the" | TO)
...
P("to" | TO)
...
P("zebra" | TO)

B_1
P("aardvark" | VB)
...
P("race" | VB)
...
P("the" | VB)
...
P("to" | VB)
...
P("zebra" | VB)

B_3
P("aardvark" | NN)
...
P("race" | NN)
...
P("the" | NN)
...
P("to" | NN)
...
P("zebra" | NN)

2/12/08

38



The A matrix for the POS HMM

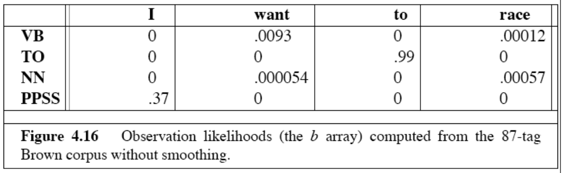
	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

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.

2/12/08

39

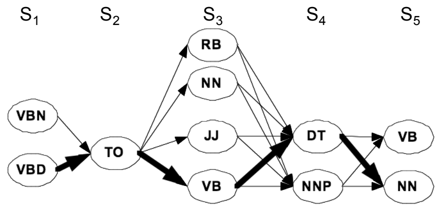
The B matrix for the POS HMM



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

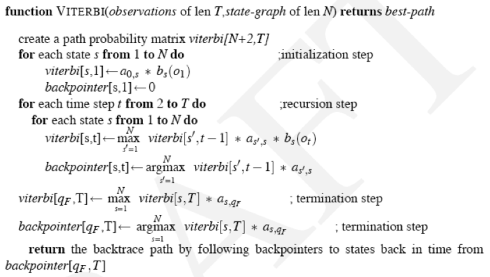
Figure 4.16 Observation likelihoods (the *b* array) computed from the 87-tag Brown corpus without smoothing.

Viterbi intuition: we are looking for the best 'path'



[redacted] [redacted] [redacted] [redacted] [redacted]

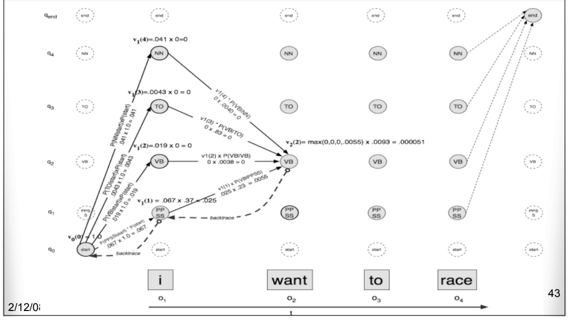
The Viterbi Algorithm



```
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 example



q_{end}
q_4
q_3
q_2
q_1
q_0
end
NN
TO
VB
PPSS
start
$v_1(4)$=.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
$v_0(0)$ = 1.0
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(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
v1(1) x P(VB|PPSS) .025 x .23 = .0055
$v_2(2)$= max(0,0,0,.0055) x .0093 = .000051
backtrace
backtrace
i
want
to
race
o_1
o_2
o_3
o_4
t

Error Analysis

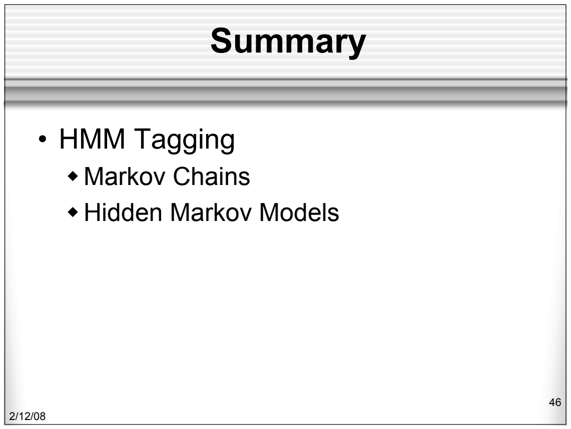
- Look at a confusion matrix

	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)

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.



Summary

- HMM Tagging
  - Markov Chains
  - Hidden Markov Models

2/12/08

46