

**6.864 (Fall 2006): Lecture 7**  
**Tagging and History-Based Models**

1

**Overview**

- The Tagging Problem
- Hidden Markov Model (HMM) taggers
- Log-linear taggers
- Log-linear models for parsing and other problems

2

**Tagging Problems**

- Mapping strings to **Tagged Sequences**

a b e e a f h j  $\Rightarrow$  a/C b/D e/C e/C a/D f/C h/D j/C

3

**Part-of-Speech Tagging**

**INPUT:**

Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

**OUTPUT:**

Profits/N soared/V at/P Boeing/N Co./N ./, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ./, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./.

N = Noun  
V = Verb  
P = Preposition  
Adv = Adverb  
Adj = Adjective  
...

4

## Information Extraction

### Named Entity Recognition

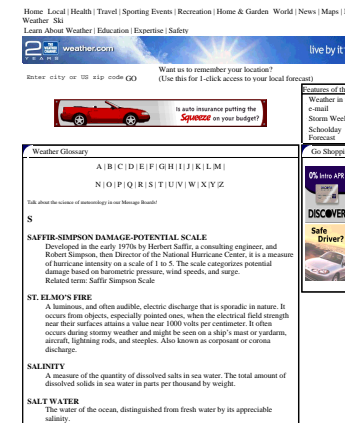
**INPUT:** Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

**OUTPUT:** Profits soared at [Company Boeing Co.], easily topping forecasts on [Location Wall Street], as their CEO [Person Alan Mulally] announced first quarter results.

5

## Extracting Glossary Entries from the Web

**Input:**



The screenshot shows a weather website interface. At the top, there are navigation links for Home, Local, Health, Travel, Sporting Events, Recreation, Home & Garden, World, News, Maps, and Weather. Below this is a search bar with the text 'weather.com' and a 'live by it!' logo. A search prompt asks 'Want us to remember your location?' with a subtext '(Use this for 1-click access to your local forecast)'. There are input fields for 'Enter city or zip code' and a 'GO' button. A red car icon is visible with the text 'is auto insurance putting the SPIN on your budget?'. On the right side, there are links for 'Features of the Weather in y', 'e-mail', 'Storm Week', 'SchoolDay', and 'Forecast'. Below the search bar, there is a 'Weather Glossary' section with an alphabetical index (A-Z) and a list of entries. The entry for 'ST. ELMO'S FIRE' is highlighted, showing its definition: 'A luminous, and often audible, electric discharge that is sporadic in nature. It occurs from objects, especially pointed ones, when the electrical field strength near their surfaces attains a value near 100 volts per centimeter. It often occurs during stormy weather and might be seen on a ship's mast or yardarm, aircraft, lightning rods, and steeples. Also known as coronas or corona discharge.'

**Output:** St. Elmo's Fire: A luminous, and often audible, electric discharge that is sporadic in nature. It occurs from objects, especially pointed ones, when the electrical field strength near their surfaces attains a value near 100 volts per centimeter...

7

## Named Entity Extraction as Tagging

**INPUT:**

Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

**OUTPUT:**

Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

NA = No entity  
SC = Start Company  
CC = Continue Company  
SL = Start Location  
CL = Continue Location  
...

6

## Our Goal

**Training set:**

- 1 Pierre/NNP Vinken/NNP ./, 61/CD years/NNS old/JJ ./, will/MD join/VB the/DT board/NN as/IN a/DT nonexecutive/JJ director/NN Nov./NNP 29/CD ./.
- 2 Mr./NNP Vinken/NNP is/VBZ chairman/NN of/IN Elsevier/NNP N.V./NNP ./, the/DT Dutch/NNP publishing/VBG group/NN ./.
- 3 Rudolph/NNP Agnew/NNP ./, 55/CD years/NNS old/JJ and/CC chairman/NN of/IN Consolidated/NNP Gold/NNP Fields/NNP PLC/NNP ./, was/VBD named/VBN a/DT nonexecutive/JJ director/NN of/IN this/DT British/JJ industrial/JJ conglomerate/NN ./.
- ...
- 38,219 It/PRP is/VBZ also/RB pulling/VBG 20/CD people/NNS out/IN of/IN Puerto/NNP Rico/NNP ./, who/WP were/VBD helping/VBG Hurricane/NNP Hugo/NNP victims/NNS ./, and/CC sending/VBG them/PRP to/TO San/NNP Francisco/NNP instead/RB ./.

- From the training set, induce a function/algorithm that maps new sentences to their tag sequences.

8

## Our Goal (continued)

- **A test data sentence:**

Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new savings-and-loan bailout agency can raise capital , creating another potential obstacle to the government 's sale of sick thrifts .

- **Should be mapped to underlying tags:**

Influential/JJ members/NNS of/IN the/DT House/NNP Ways/NNP and/CC Means/NNP Committee/NNP introduced/VBD legislation/NN that/WDT would/MD restrict/VB how/WRB the/DT new/JJ savings-and-loan/NN bailout/NN agency/NN can/MD raise/VB capital/NN ./, creating/VBG another/DT potential/JJ obstacle/NN to/TO the/DT government/NN 's/POS sale/NN of/IN sick/JJ thrifts/NNS ./.

- Our goal is to minimize the number of tagging errors on sentences not seen in the training set

9

## A Naive Approach

- Use a machine learning method to build a “classifier” that maps each word individually to its tag
- A problem: does not take contextual constraints into account

11

## Two Types of Constraints

Influential/JJ members/NNS of/IN the/DT House/NNP Ways/NNP and/CC Means/NNP Committee/NNP introduced/VBD legislation/NN that/WDT would/MD restrict/VB how/WRB the/DT new/JJ savings-and-loan/NN bailout/NN agency/NN can/MD raise/VB capital/NN ./.

- “Local”: e.g., *can* is more likely to be a modal verb **MD** rather than a noun **NN**
- “Contextual”: e.g., a noun is much more likely than a verb to follow a determiner
- Sometimes these preferences are in conflict:

The trash can is in the garage

10

## Hidden Markov Models

- We have an input sentence  $S = w_1, w_2, \dots, w_n$  ( $w_i$  is the  $i$ 'th word in the sentence)
- We have a tag sequence  $T = t_1, t_2, \dots, t_n$  ( $t_i$  is the  $i$ 'th tag in the sentence)

- We'll use an HMM to define

$$P(t_1, t_2, \dots, t_n, w_1, w_2, \dots, w_n)$$

for any sentence  $S$  and tag sequence  $T$  of the same length.

- Then the most likely tag sequence for  $S$  is

$$T^* = \operatorname{argmax}_T P(T, S)$$

12

## How to model $P(T, S)$ ?

### A Trigram HMM Tagger:

$$P(T, S) = P(\text{END} | t_1 \dots t_n, w_1 \dots w_n) \times \prod_{j=1}^n [P(t_j | w_1 \dots w_{j-1}, t_1 \dots t_{j-1}) \times P(w_j | w_1 \dots w_{j-1}, t_1 \dots t_j)] \quad \text{Chain rule}$$

$$= P(\text{END} | t_{n-1}, t_n) \times \prod_{j=1}^n [P(t_j | t_{j-2}, t_{j-1}) \times P(w_j | t_j)] \quad \text{Independence assumptions}$$

- END is a special tag that terminates the sequence
- We take  $t_0 = t_{-1} = \text{START}$
- 1st assumption: each tag only depends on previous two tags  
 $P(t_j | t_{j-1}, t_{j-2})$
- 2nd assumption: each word only depends on underlying tag  
 $P(w_j | t_j)$

13

## Why the Name?

$$P(T, S) = \underbrace{P(\text{END} | t_{n-1}, t_n) \prod_{j=1}^n P(t_j | t_{j-2}, t_{j-1})}_{\text{Hidden Markov Chain}} \times \underbrace{\prod_{j=1}^n P(w_j | t_j)}_{w_j \text{'s are observed}}$$

15

## An Example

- $S =$  the boy laughed
- $T =$  DT NN VBD

$$P(T, S) = P(\text{END} | \text{NN}, \text{VBD}) \times P(\text{DT} | \text{START}, \text{START}) \times P(\text{NN} | \text{START}, \text{DT}) \times P(\text{VBD} | \text{DT}, \text{NN}) \times P(\text{the} | \text{DT}) \times P(\text{boy} | \text{NN}) \times P(\text{laughed} | \text{VBD})$$

14

## How to model $P(T, S)$ ?

Hispaniola/**NNP** quickly/**RB** became/**VB** an/**DT** important/**JJ**  
base/**Vt** from which Spain expanded its empire into the rest of the  
Western Hemisphere .

“Score” for tag  $\text{Vt}$ :

$$P(\text{Vt} | \text{DT}, \text{JJ}) \times P(\text{base} | \text{Vt})$$

16

## Smoothed Estimation

$$P(Vt | DT, JJ) = \lambda_1 \times \frac{Count(Dt, JJ, Vt)}{Count(Dt, JJ)} \\ + \lambda_2 \times \frac{Count(JJ, Vt)}{Count(JJ)} \\ + \lambda_3 \times \frac{Count(Vt)}{Count()}$$

$$P(base | Vt) = \frac{Count(Vt, base)}{Count(Vt)}$$

17

## Dealing with Low-Frequency Words: An Example

[Bikel et. al 1999] An Algorithm that Learns What's in a Name

Word class	Example	Intuition
twoDigitNum	90	Two digit year
fourDigitNum	1990	Four digit year
containsDigitAndAlpha	A8956-67	Product code
containsDigitAndDash	09-96	Date
containsDigitAndSlash	11/9/89	Date
containsDigitAndComma	23,000.00	Monetary amount
containsDigitAndPeriod	1.00	Monetary amount,percentage
othernum	456789	Other number
allCaps	BBN	Organization
capPeriod	M.	Person name initial
fi rstWord	fi rst word of sentence	no useful capitalization information
initCap	Sally	Capitalized word
lowercase	can	Uncapitalized word
other	,	Punctuation marks, all other words

19

## Dealing with Low-Frequency Words

- **Step 1:** Split vocabulary into two sets
  - Frequent words** = words occurring  $\geq 5$  times in training
  - Low frequency words** = all other words
- **Step 2:** Map low frequency words into a small, finite set, depending on prefixes, suffixes etc.

18

## Dealing with Low-Frequency Words: An Example

Profi ts/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA fi rst/NA quarter/NA results/NA ./NA

↓

firstword/NA soared/NA at/NA initCap/SC Co./CC ,/NA easily/NA lowercase/NA forecasts/NA on/NA initCap/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP initCap/CP announced/NA fi rst/NA quarter/NA results/NA ./NA

NA = No entity  
 SC = Start Company  
 CC = Continue Company  
 SL = Start Location  
 CL = Continue Location

...

20

## The Viterbi Algorithm

- Question: how do we calculate the following?:

$$T^* = \operatorname{argmax}_T \log P(T, S)$$

- Define  $n$  to be the length of the sentence
- Define a dynamic programming table

$$\pi[i, t_{-2}, t_{-1}] = \text{maximum log probability of a tag sequence ending in tags } t_{-2}, t_{-1} \text{ at position } i$$

- Our goal is to calculate  $\max_{t_{-2}, t_{-1} \in \mathcal{T}} \pi[n, t_{-2}, t_{-1}]$

21

## The Viterbi Algorithm: Running Time

- $O(n|\mathcal{T}|^3)$  time to calculate  $Score(S, i, t, t_{-2}, t_{-1})$  for all  $i, t, t_{-2}, t_{-1}$ .
- $n|\mathcal{T}|^2$  entries in  $\pi$  to be filled in.
- $O(\mathcal{T})$  time to fill in one entry (assuming  $O(1)$  time to look up  $Score(S, i, t, t_{-2}, t_{-1})$ )
- $\Rightarrow O(n|\mathcal{T}|^3)$  time

23

## The Viterbi Algorithm: Recursive Definitions

- **Base case:**

$$\pi[0, *, *] = \log 1 = 0$$

$$\pi[0, t_{-2}, t_{-1}] = \log 0 = -\infty \text{ for all other } t_{-2}, t_{-1}$$

here  $*$  is a special tag padding the beginning of the sentence.

- **Recursive case:** for  $i = 1 \dots n$ , for all  $t_{-2}, t_{-1}$ ,

$$\pi[i, t_{-2}, t_{-1}] = \max_{t \in \mathcal{T} \cup \{*\}} \{ \pi[i-1, t, t_{-2}] + Score(S, i, t, t_{-2}, t_{-1}) \}$$

Backpointers allow us to recover the max probability sequence:

$$BP[i, t_{-2}, t_{-1}] = \operatorname{argmax}_{t \in \mathcal{T} \cup \{*\}} \{ \pi[i-1, t, t_{-2}] + Score(S, i, t, t_{-2}, t_{-1}) \}$$

---

**Where**  $Score(S, i, t, t_{-2}, t_{-1}) = \log P(t_{-1} | t, t_{-2}) + \log P(w_i | t_{-1})$

Complexity is  $O(nk^3)$ , where  $n$  = length of sentence,  $k$  is number of possible tags

22

## Pros and Cons

- Hidden markov model taggers are very simple to train (compile counts from the training corpus)
- Perform relatively well (over 90% performance on named entities)
- Main difficulty is modeling

$$P(\text{word} | \text{tag})$$

can be very difficult if “words” are complex

24

## Log-Linear Models

- We have an input sentence  $S = w_1, w_2, \dots, w_n$   
( $w_i$  is the  $i$ 'th word in the sentence)
- We have a tag sequence  $T = t_1, t_2, \dots, t_n$   
( $t_i$  is the  $i$ 'th tag in the sentence)

- We'll use an log-linear model to define

$$P(t_1, t_2, \dots, t_n | w_1, w_2, \dots, w_n)$$

for any sentence  $S$  and tag sequence  $T$  of the same length.

(Note: contrast with HMM that defines  
 $P(t_1, t_2, \dots, t_n, w_1, w_2, \dots, w_n)$ )

- Then the most likely tag sequence for  $S$  is

$$T^* = \operatorname{argmax}_T P(T|S)$$

25

## An Example

Hispaniola/**NNP** quickly/**RB** became/**VB** an/**DT**  
important/**JJ** base/**??** from which Spain expanded  
its empire into the rest of the Western Hemisphere .

- There are many possible tags in the position **??**

$$\mathcal{Y} = \{\text{NN, NNS, Vt, Vi, IN, DT, \dots}\}$$

- The input domain  $\mathcal{X}$  is the set of all possible **histories** (or contexts)

- Need to learn a function from (history, tag) pairs to a probability  
 $P(\text{tag}|\text{history})$

27

## How to model $P(T|S)$ ?

### A Trigram Log-Linear Tagger:

$$P(T|S) = \prod_{j=1}^n P(t_j | w_1 \dots w_n, t_1 \dots t_{j-1}) \quad \text{Chain rule}$$

$$= \prod_{j=1}^n P(t_j | t_{j-2}, t_{j-1}, w_1, \dots, w_n) \quad \text{Independence assumptions}$$

- We take  $t_0 = t_{-1} = \text{START}$
- Assumption: each tag only depends on previous two tags  
 $P(t_j | t_{j-1}, t_{j-2}, w_1, \dots, w_n)$

26

## Representation: Histories

- A **history** is a 4-tuple  $\langle t_{-1}, t_{-2}, w_{[1:n]}, i \rangle$
- $t_{-1}, t_{-2}$  are the previous two tags.
- $w_{[1:n]}$  are the  $n$  words in the input sentence.
- $i$  is the index of the word being tagged
- $\mathcal{X}$  is the set of all possible histories

Hispaniola/**NNP** quickly/**RB** became/**VB** an/**DT** important/**JJ**  
base/**??** from which Spain expanded its empire into the rest of the  
Western Hemisphere .

- $t_{-1}, t_{-2} = \text{DT, JJ}$
- $w_{[1:n]} = \langle \text{Hispaniola, quickly, became, \dots, Hemisphere, .} \rangle$
- $i = 6$

28

## Feature Vector Representations

- We have some input domain  $\mathcal{X}$ , and a finite label set  $\mathcal{Y}$ . Aim is to provide a conditional probability  $P(y | x)$  for any  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ .
- A **feature** is a function  $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$   
(Often **binary features** or **indicator functions**  $f : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ ).
- Say we have  $m$  features  $\phi_k$  for  $k = 1 \dots m$   
 $\Rightarrow$  A **feature vector**  $\phi(x, y) \in \mathbb{R}^m$  for any  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ .

29

## The Full Set of Features in [(Ratnaparkhi, 96)]

- Word/tag features for all word/tag pairs, e.g.,

$$\phi_{100}(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ is base and } t = \text{Vt} \\ 0 & \text{otherwise} \end{cases}$$

- Spelling features for all prefixes/suffixes of length  $\leq 4$ , e.g.,

$$\phi_{101}(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in ing and } t = \text{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{102}(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ starts with pre and } t = \text{NN} \\ 0 & \text{otherwise} \end{cases}$$

31

## An Example (continued)

- $\mathcal{X}$  is the set of all possible histories of form  $\langle t_{-1}, t_{-2}, w_{[1:n]}, i \rangle$
- $\mathcal{Y} = \{\text{NN, NNS, Vt, Vi, IN, DT, \dots}\}$
- We have  $m$  features  $\phi_k : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$  for  $k = 1 \dots m$

For example:

$$\phi_1(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ is base and } t = \text{Vt} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_2(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in ing and } t = \text{VBG} \\ 0 & \text{otherwise} \end{cases}$$

...

$$\phi_1(\langle \text{JJ, DT, } \langle \text{Hispaniola, \dots} \rangle, 6 \rangle, \text{Vt}) = 1$$

$$\phi_2(\langle \text{JJ, DT, } \langle \text{Hispaniola, \dots} \rangle, 6 \rangle, \text{Vt}) = 0$$

...

30

## The Full Set of Features in [(Ratnaparkhi, 96)]

- Contextual Features, e.g.,

$$\phi_{103}(h, t) = \begin{cases} 1 & \text{if } \langle t_{-2}, t_{-1}, t \rangle = \langle \text{DT, JJ, Vt} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{104}(h, t) = \begin{cases} 1 & \text{if } \langle t_{-1}, t \rangle = \langle \text{JJ, Vt} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{105}(h, t) = \begin{cases} 1 & \text{if } \langle t \rangle = \langle \text{Vt} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{106}(h, t) = \begin{cases} 1 & \text{if previous word } w_{i-1} = \text{the and } t = \text{Vt} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{107}(h, t) = \begin{cases} 1 & \text{if next word } w_{i+1} = \text{the and } t = \text{Vt} \\ 0 & \text{otherwise} \end{cases}$$

32

## The Final Result

- We can come up with practically any questions (*features*) regarding history/tag pairs.
- For a given history  $x \in \mathcal{X}$ , each label in  $\mathcal{Y}$  is mapped to a different feature vector

$$\begin{aligned} \phi(\langle \text{JJ, DT}, \langle \text{Hispaniola}, \dots \rangle, \text{6} \rangle, \text{Vt}) &= 1001011001001100110 \\ \phi(\langle \text{JJ, DT}, \langle \text{Hispaniola}, \dots \rangle, \text{6} \rangle, \text{JJ}) &= 0110010101011110010 \\ \phi(\langle \text{JJ, DT}, \langle \text{Hispaniola}, \dots \rangle, \text{6} \rangle, \text{NN}) &= 0001111101001100100 \\ \phi(\langle \text{JJ, DT}, \langle \text{Hispaniola}, \dots \rangle, \text{6} \rangle, \text{IN}) &= 0001011011000000010 \\ &\dots \end{aligned}$$

33

## Training the Log-Linear Model

- To train a log-linear model, we need a training set  $(x_i, y_i)$  for  $i = 1 \dots n$ . Then search for

$$\mathbf{W}^* = \operatorname{argmax}_{\mathbf{W}} \left( \underbrace{\sum_i \log P(y_i | x_i, \mathbf{W})}_{\text{Log-Likelihood}} - C \underbrace{\sum_k \mathbf{W}_k^2}_{\text{Gaussian Prior}} \right)$$

(see last lecture on log-linear models)

- Training set is simply all history/tag pairs seen in the training data

35

## Log-Linear Models

- We have some input domain  $\mathcal{X}$ , and a finite label set  $\mathcal{Y}$ . Aim is to provide a conditional probability  $P(y | x)$  for any  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ .
- A feature is a function  $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$   
(Often binary features or indicator functions  $f : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ ).
- Say we have  $m$  features  $\phi_k$  for  $k = 1 \dots m$   
 $\Rightarrow$  A feature vector  $\phi(x, y) \in \mathbb{R}^m$  for any  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ .
- We also have a **parameter vector**  $\mathbf{W} \in \mathbb{R}^m$
- We define

$$P(y | x, \mathbf{W}) = \frac{e^{\mathbf{W} \cdot \phi(x, y)}}{\sum_{y' \in \mathcal{Y}} e^{\mathbf{W} \cdot \phi(x, y')}}$$

34

## The Viterbi Algorithm for Log-Linear Models

- Question: how do we calculate the following?:

$$T^* = \operatorname{argmax}_T \log P(T | S)$$

- Define  $n$  to be the length of the sentence
- Define a dynamic programming table

$$\pi[i, t_{-2}, t_{-1}] = \text{maximum log probability of a tag sequence ending in tags } t_{-2}, t_{-1} \text{ at position } i$$

- Our goal is to calculate  $\max_{t_{-2}, t_{-1} \in \mathcal{T}} \pi[n, t_{-2}, t_{-1}]$

36

## The Viterbi Algorithm: Recursive Definitions

- **Base case:**

$$\pi[0, *, *] = \log 1 = 0$$

$$\pi[0, t_{-2}, t_{-1}] = \log 0 = -\infty \text{ for all other } t_{-2}, t_{-1}$$

here \* is a special tag padding the beginning of the sentence.

- **Recursive case:** for  $i = 1 \dots n$ , for all  $t_{-2}, t_{-1}$ ,

$$\pi[i, t_{-2}, t_{-1}] = \max_{t \in \mathcal{T} \cup \{*\}} \{ \pi[i-1, t, t_{-2}] + \text{Score}(S, i, t, t_{-2}, t_{-1}) \}$$

Backpointers allow us to recover the max probability sequence:

$$\text{BP}[i, t_{-2}, t_{-1}] = \operatorname{argmax}_{t \in \mathcal{T} \cup \{*\}} \{ \pi[i-1, t, t_{-2}] + \text{Score}(S, i, t, t_{-2}, t_{-1}) \}$$

---

Where  $\text{Score}(S, i, t, t_{-2}, t_{-1}) = \log P(t_{-1} | t, t_{-2}, w_1, \dots, w_n, i)$

Identical to Viterbi for HMMs, except for the definition of  $\text{Score}(S, i, t, t_{-2}, t_{-1})$

## FAQ Segmentation: McCallum et. al

```
<head>X-NNTP-POSTER: NewsHound v1.33
<head>
<head>Archive name: acorn/faq/part2
<head>Frequency: monthly
<head>
<question>2.6) What configuration of serial cable should I use
<answer>
<answer> Here follows a diagram of the necessary connections
<answer>programs to work properly. They are as far as I know t
<answer>agreed upon by commercial comms software developers fo
<answer>
<answer> Pins 1, 4, and 8 must be connected together inside
<answer>is to avoid the well known serial port chip bugs. The
```

## FAQ Segmentation: McCallum et. al

- McCallum et. al compared HMM and log-linear taggers on a *FAQ Segmentation* task

- Main point: in an HMM, modeling

$$P(\text{word} | \text{tag})$$

is difficult in this domain

## FAQ Segmentation: Line Features

```
begins-with-number
begins-with-ordinal
begins-with-punctuation
begins-with-question-word
begins-with-subject
blank
contains-alphanum
contains-bracketed-number
contains-http
contains-non-space
contains-number
contains-pipe
contains-question-mark
ends-with-question-mark
first-alpha-is-capitalized
indented-1-to-4
indented-5-to-10
more-than-one-third-space
only-punctuation
prev-is-blank
prev-begins-with-ordinal
shorter-than-30
```

## FAQ Segmentation: The Log-Linear Tagger

```
<head>X-NNTP-POSTER: NewsHound v1.33
<head>
<head>Archive name: acorn/faq/part2
<head>Frequency: monthly
<head>
<question>2.6) What configuration of serial cable should I use

    Here follows a diagram of the necessary connections
    programs to work properly. They are as far as I know t
    agreed upon by commercial comms software developers fo

    Pins 1, 4, and 8 must be connected together inside
    is to avoid the well known serial port chip bugs. The
```

⇒ “tag=question;prev=head;begins-with-number”  
“tag=question;prev=head;contains-alphanum”  
“tag=question;prev=head;contains-non-space”  
“tag=question;prev=head;contains-number”  
“tag=question;prev=head;prev-is-blank”

41

## FAQ Segmentation: McCallum et. al

- Second solution: first map each sentence to string of features:

```
<question>2.6) What configuration of serial cable should I use

⇒

<question>begins-with number contains-alphanum contains-non-space
```

- Use a language model again:

$P(\text{"2.6) What configuration of serial cable should I use"} | \text{question}) =$   
 $P(\text{begins-with-number} | \text{question}) \times$   
 $P(\text{contains-alphanum} | \text{question}) \times$   
 $P(\text{contains-non-space} | \text{question}) \times$   
 $P(\text{contains-number} | \text{question}) \times$   
 $P(\text{prev-is-blank} | \text{question}) \times$

43

## FAQ Segmentation: An HMM Tagger

```
<question>2.6) What configuration of serial cable should I use
```

- First solution for  $P(\text{word} | \text{tag})$ :

$P(\text{"2.6) What configuration of serial cable should I use"} | \text{question}) =$   
 $P(2.6 | \text{question}) \times$   
 $P(\text{What} | \text{question}) \times$   
 $P(\text{configuration} | \text{question}) \times$   
 $P(\text{of} | \text{question}) \times$   
 $P(\text{serial} | \text{question}) \times$   
...

- i.e. have a **language model** for each *tag*

42

## FAQ Segmentation: Results

Method	COAP	SegPrec	SegRec
ME-Stateless	0.520	0.038	0.362
TokenHMM	0.865	0.276	0.140
FeatureHMM	0.941	0.413	0.529
MEMM	0.965	0.867	0.681

44

## Overview

- The Tagging Problem
- Hidden Markov Model (HMM) taggers
- Log-linear taggers
- Log-linear models for parsing and other problems

45

## A General Approach: (Conditional) History-Based Models

- We've shown how to define  $P(T | S)$  where  $T$  is a tag sequence
- How do we define  $P(T | S)$  if  $T$  is a parse tree (or another structure)?

47

## Log-Linear Taggers: Summary

- The input sentence is  $S = w_1 \dots w_n$
  - Each tag sequence  $T$  has a conditional probability
- $$P(T | S) = \prod_{j=1}^n P(t_j | w_1 \dots w_n, j, t_1 \dots t_{j-1}) \quad \text{Chain rule}$$
- $$= \prod_{j=1}^n P(t_j | w_1 \dots w_n, j, t_{j-2}, t_{j-1}) \quad \text{Independence assumptions}$$
- Estimate  $P(t_j | w_1 \dots w_n, j, t_{j-2}, t_{j-1})$  using log-linear models
  - Use the Viterbi algorithm to compute

$$\operatorname{argmax}_{T \in \mathcal{T}^n} \log P(T | S)$$

46

## A General Approach: (Conditional) History-Based Models

- Step 1: represent a tree as a sequence of **decisions**  $d_1 \dots d_m$

$$T = \langle d_1, d_2, \dots, d_m \rangle$$

$m$  is **not** necessarily the length of the sentence

- Step 2: the probability of a tree is

$$P(T | S) = \prod_{i=1}^m P(d_i | d_1 \dots d_{i-1}, S)$$

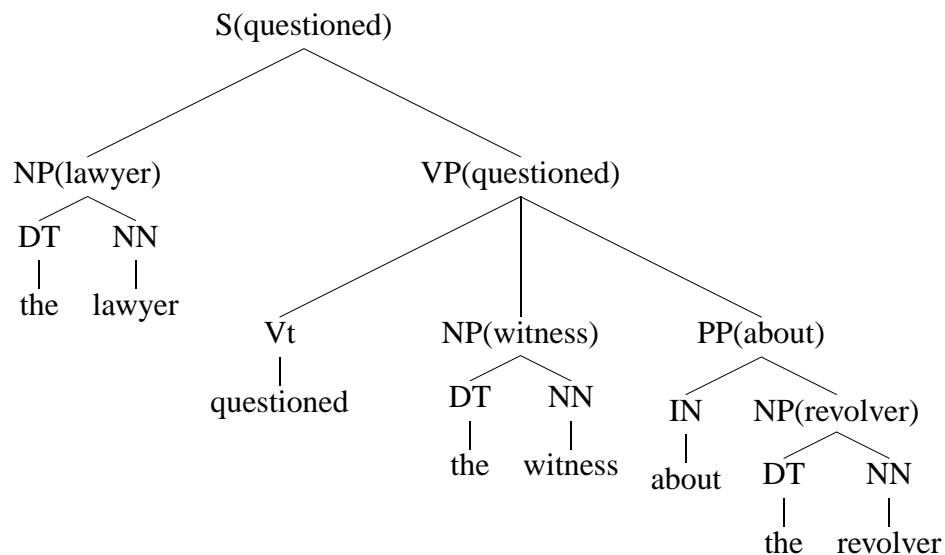
- Step 3: Use a log-linear model to estimate

$$P(d_i | d_1 \dots d_{i-1}, S)$$

- Step 4: Search?? (answer we'll get to later: beam or heuristic search)

48

## An Example Tree



49

## Layer 1: Part-of-Speech Tags

DT    NN            Vt    DT    NN    IN    DT    NN  
|    |            |    |    |    |    |  
the lawyer questioned the witness about the revolver

---

- Step 1: represent a tree as a sequence of **decisions**  $d_1 \dots d_m$

$$T = \langle d_1, d_2, \dots, d_m \rangle$$

- **First  $n$  decisions are tagging decisions**

$$\langle d_1 \dots d_n \rangle = \langle \text{DT, NN, Vt, DT, NN, IN, DT, NN} \rangle$$

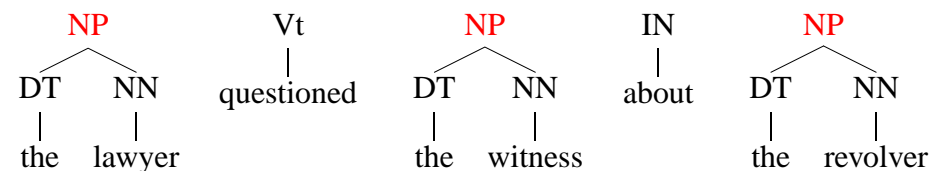
51

## Ratnaparkhi's Parser: Three Layers of Structure

1. Part-of-speech tags
2. Chunks
3. Remaining structure

50

## Layer 2: Chunks

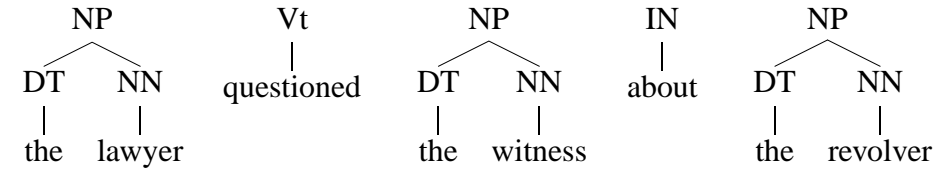
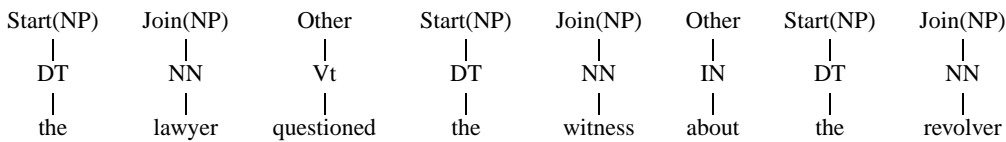


**Chunks are defined as any phrase where all children are part-of-speech tags**

(Other common chunks are ADJP, QP)

52

## Layer 2: Chunks



- Step 1: represent a tree as a sequence of **decisions**  $d_1 \dots d_n$

$$T = \langle d_1, d_2, \dots, d_n \rangle$$

- First  $n$  decisions are tagging decisions

Next  $n$  decisions are chunk tagging decisions

$$\langle d_1 \dots d_{2n} \rangle = \langle \text{DT, NN, Vt, DT, NN, IN, DT, NN, Start(NP), Join(NP), Other, Start(NP), Join(NP), Other, Start(NP), Join(NP)} \rangle$$

53

55

## Layer 3: Remaining Structure

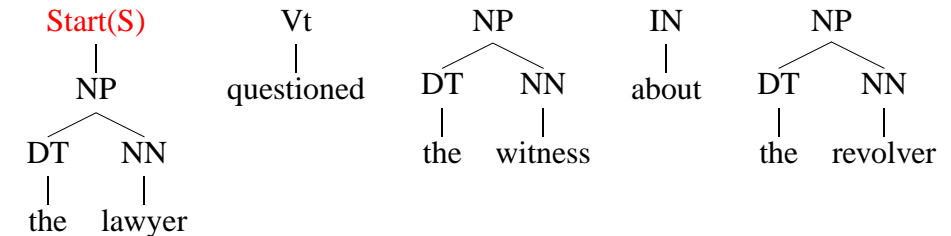
### Alternate Between Two Classes of Actions:

- Join(X) or Start(X), where X is a label (NP, S, VP etc.)
- Check=YES or Check=NO

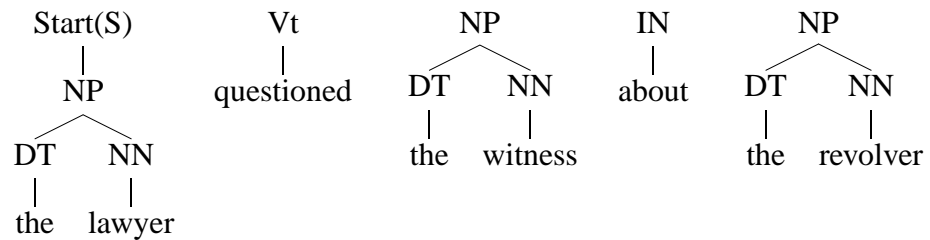
### Meaning of these actions:

- Start(X) starts a new constituent with label X  
(always acts on leftmost constituent with no start or join label above it)
- Join(X) continues a constituent with label X  
(always acts on leftmost constituent with no start or join label above it)
- Check=NO does nothing
- Check=YES takes previous Join or Start action, and converts it into a completed constituent

54

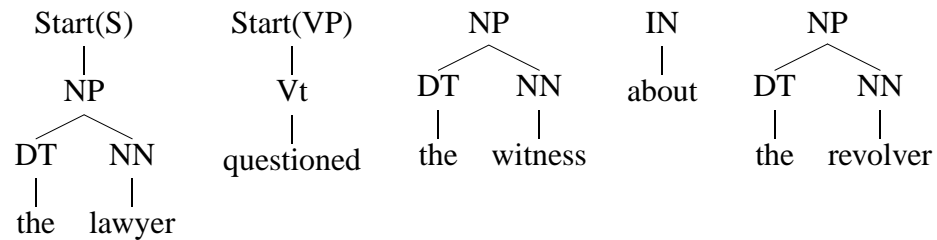


56



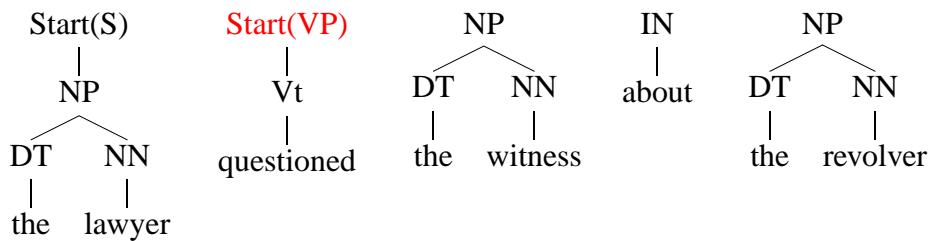
Check=NO

57

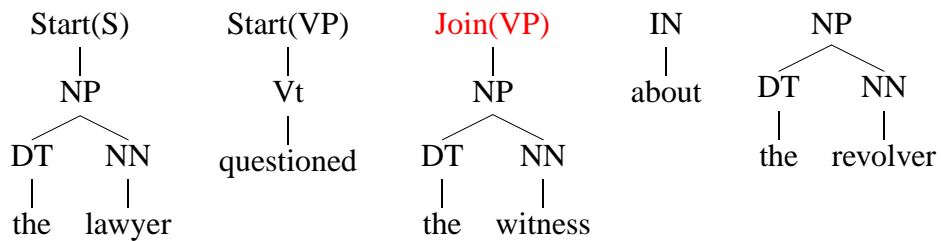


Check=NO

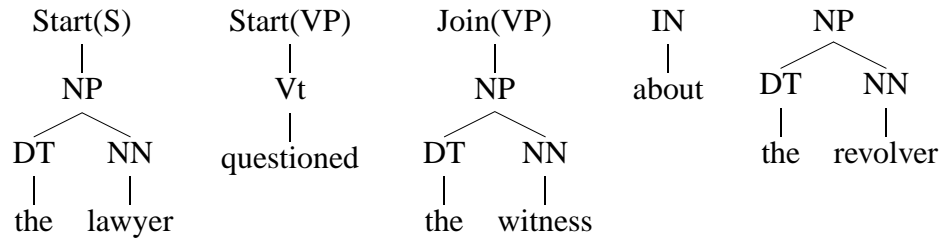
59



58

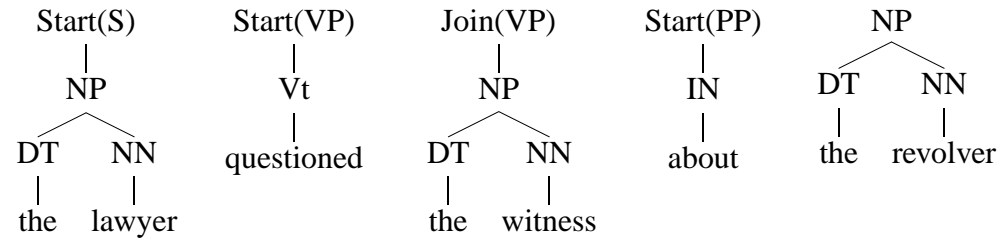


60



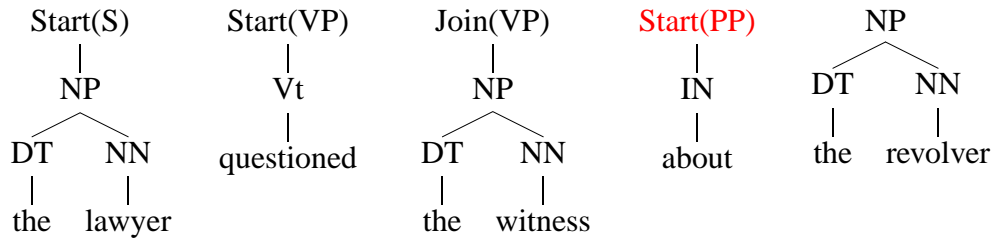
Check=NO

61

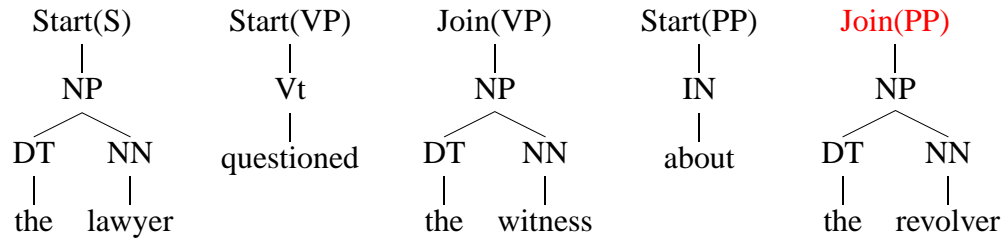


Check=NO

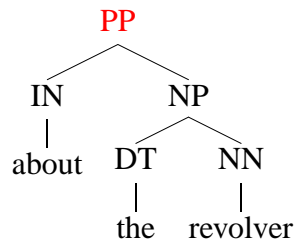
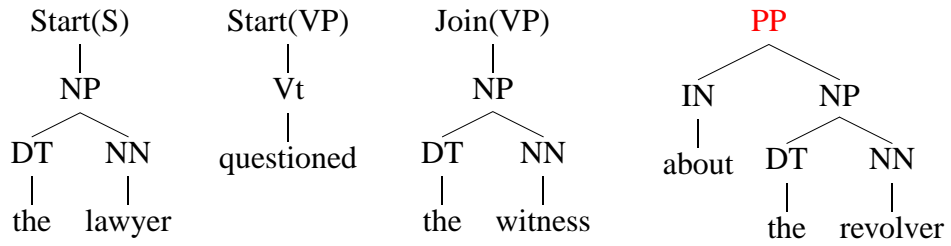
63



62

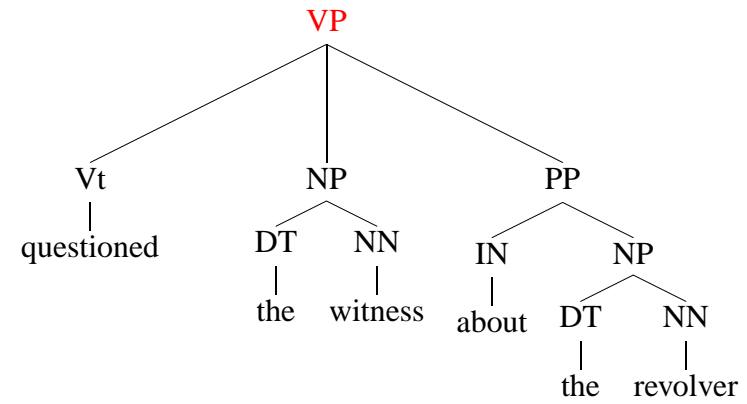
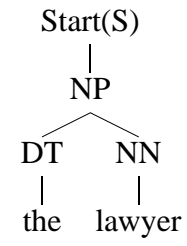


64



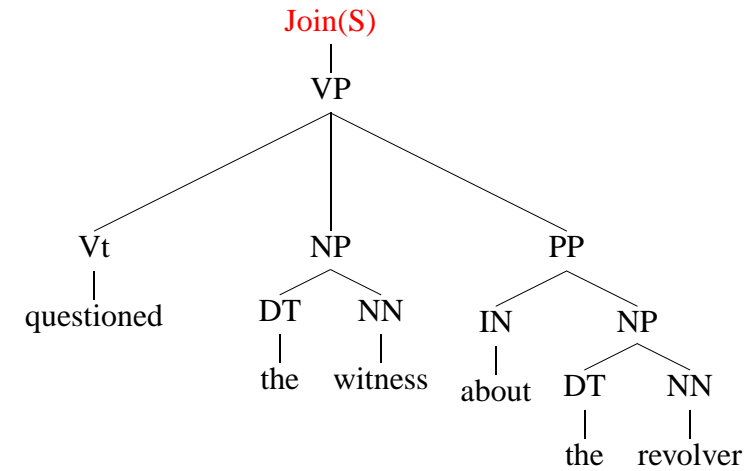
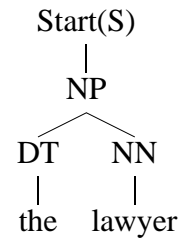
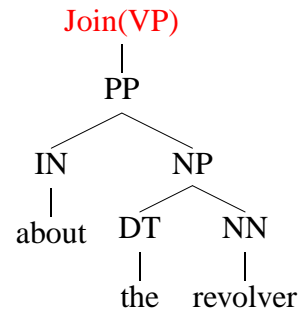
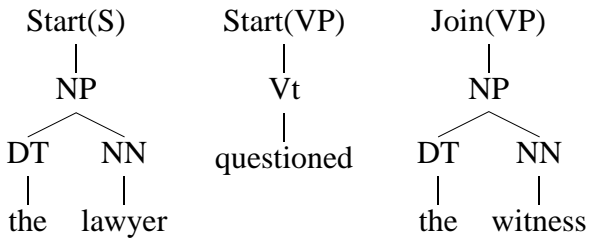
Check=YES

65



Check=YES

67



66

68

## A General Approach: (Conditional) History-Based Models

- Step 1: represent a tree as a sequence of **decisions**  $d_1 \dots d_m$

$$T = \langle d_1, d_2, \dots, d_m \rangle$$

$m$  is **not** necessarily the length of the sentence

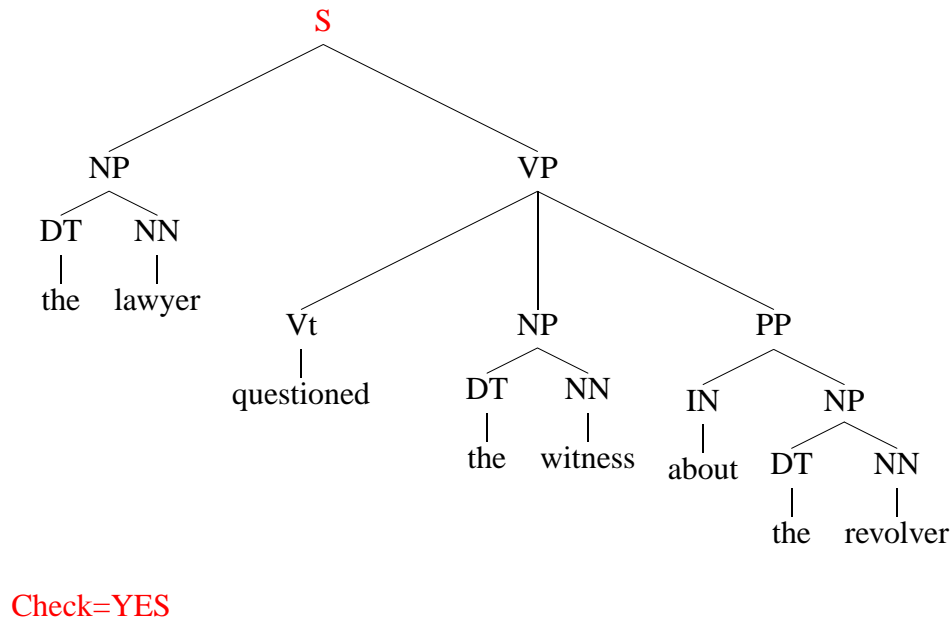
- Step 2: the probability of a tree is

$$P(T | S) = \prod_{i=1}^m P(d_i | d_1 \dots d_{i-1}, S)$$

- Step 3: Use a log-linear model to estimate

$$P(d_i | d_1 \dots d_{i-1}, S)$$

- Step 4: Search?? (answer we'll get to later: beam or heuristic search)



69

### The Final Sequence of decisions

$\langle d_1 \dots d_m \rangle = \langle$  DT, NN, Vt, DT, NN, IN, DT, NN,  
 Start(NP), Join(NP), Other, Start(NP), Join(NP),  
 Other, Start(NP), Join(NP),  
 Start(S), Check=NO, Start(VP), Check=NO,  
 Join(VP), Check=NO, Start(PP), Check=NO,  
 Join(PP), Check=YES, Join(VP), Check=YES,  
 Join(S), Check=YES  $\rangle$

70

### Applying a Log-Linear Model

- Step 3: Use a log-linear model to estimate

$$P(d_i | d_1 \dots d_{i-1}, S)$$

- A reminder:

$$P(d_i | d_1 \dots d_{i-1}, S) = \frac{e^{\phi(\langle d_1 \dots d_{i-1}, S \rangle, d_i) \cdot \mathbf{W}}}{\sum_{d \in \mathcal{A}} e^{\phi(\langle d_1 \dots d_{i-1}, S \rangle, d) \cdot \mathbf{W}}}$$

where:

$\langle d_1 \dots d_{i-1}, S \rangle$  is the history

$d_i$  is the outcome

$\phi$  maps a history/outcome pair to a feature vector

$\mathbf{W}$  is a parameter vector

$\mathcal{A}$  is set of possible actions

(may be context dependent)

72

## Applying a Log-Linear Model

- Step 3: Use a log-linear model to estimate

$$P(d_i | d_1 \dots d_{i-1}, S) = \frac{e^{\phi((d_1 \dots d_{i-1}, S), d_i) \cdot \mathbf{W}}}{\sum_{d \in \mathcal{A}} e^{\phi((d_1 \dots d_{i-1}, S), d) \cdot \mathbf{W}}}$$

- **The big question: how do we define  $\phi$ ?**
- Ratnaparkhi's method defines  $\phi$  differently depending on whether next decision is:
  - A tagging decision  
(same features as before for POS tagging!)
  - A chunking decision
  - A start/join decision after chunking
  - A check=no/check=yes decision

73

## Layer 3: Join or Start

- Looks at head word, constituent (or POS) label, and start/join annotation of  $n$ 'th tree relative to the decision, where  $n = -2, -1$
- Looks at head word, constituent (or POS) label of  $n$ 'th tree relative to the decision, where  $n = 0, 1, 2$
- Looks at bigram features of the above for  $(-1, 0)$  and  $(0, 1)$
- Looks at trigram features of the above for  $(-2, -1, 0)$ ,  $(-1, 0, 1)$  and  $(0, 1, 2)$
- The above features with all combinations of head words excluded
- Various punctuation features

75

## Layer 2: Chunks

Start(NP)	Join(NP)	Other	Start(NP)	Join(NP)	IN	DT	NN
DT	NN	Vt	DT	NN	about	the	revolver
the	lawyer	questioned	the	witness			

- ⇒ “TAG=Join(NP); Word0=witness; POS0=NN”  
 “TAG=Join(NP); POS0=NN”  
 “TAG=Join(NP); Word+1=about; POS+1=IN”  
 “TAG=Join(NP); POS+1=IN”  
 “TAG=Join(NP); Word+2=the; POS+2=DT”  
 “TAG=Join(NP); POS+2=IN”  
 “TAG=Join(NP); Word-1=the; POS-1=DT; TAG-1=Start(NP)”  
 “TAG=Join(NP); POS-1=DT; TAG-1=Start(NP)”  
 “TAG=Join(NP); TAG-1=Start(NP)”  
 ...

74

## Layer 3: Check=NO or Check=YES

- A variety of questions concerning the proposed constituent

76

## The Search Problem

- In POS tagging, we could use the Viterbi algorithm because

$$P(t_j | w_1 \dots w_n, j, t_1 \dots t_{j-1}) = P(t_j | w_1 \dots w_n, j, t_{j-2} \dots t_{j-1})$$

- Now: Decision  $d_i$  could depend on arbitrary decisions in the “past”  $\Rightarrow$  no chance for dynamic programming
- Instead, Ratnaparkhi uses a beam search method