Statistical Methods in Natural Language Processing

Michael Collins
AT&T Labs-Research

Overview

Some NLP problems:

- Information extraction
  (Named entities, Relationships between entities, etc.)
- Finding linguistic structure
  Part-of-speech tagging, “Chunking”, Parsing

Techniques:

- Log-linear (maximum-entropy) taggers
- Probabilistic context-free grammars (PCFGs)
  PCFGs with enriched non-terminals
- Discriminative methods:
  Conditional MRFs, Perceptron algorithms, Kernel methods

Some NLP Problems

- Information extraction
  - Named entities
  - Relationships between entities
  - More complex relationships
- Finding linguistic structure
  - Part-of-speech tagging
  - “Chunking” (low-level syntactic structure)
  - Parsing
- Machine translation

Common Themes

- Need to learn mapping from one discrete structure to another
  - Strings to hidden state sequences
    Named-entity extraction, part-of-speech tagging
  - Strings to strings
    Machine translation
  - Strings to underlying trees
    Parsing
  - Strings to relational data structures
    Information extraction
- Speech recognition is similar (and shares many techniques)

Two Fundamental Problems

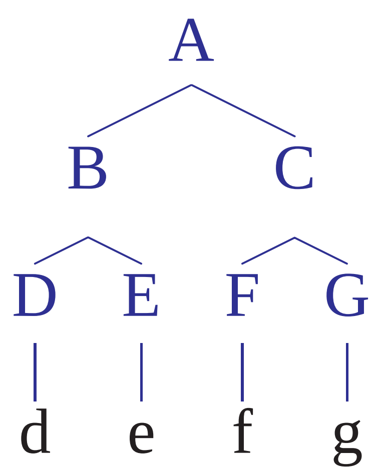
TAGGING: Strings to Tagged Sequences

a b e e a f h j ⇒ a/C b/D e/C e/C a/D f/C h/D j/C

PARSING: Strings to Trees

d e f g ⇒ (A (B (D d) (E e)) (C (F f) (G g)))

d e f g ⇒



Information Extraction: Named Entities

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.

Information Extraction: Relationships between Entities

INPUT:

Boeing is located in Seattle. Alan Mulally is the CEO.

OUTPUT:

{Relationship = Company-Location
Company = Boeing
Location = Seattle}

{Relationship = Employer-Employee
Employer = Boeing Co.
Employee = Alan Mulally}

Information Extraction: More Complex Relationships

INPUT:

Alan Mulally resigned as Boeing CEO yesterday. He will be succeeded by Jane Swift, who was previously the president at Rolls Royce.

OUTPUT:

{Relationship = Management Succession
Company = Boeing Co.
Role = CEO
Out = Alan Mulally
In = Jane Swift}

{Relationship = Management Succession
Company = Rolls Royce
Role = president
Out = Jane Swift}

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
. . .

“Chunking” (Low-level syntactic structure)

INPUT:

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

OUTPUT:

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

[NP . . .] = non-recursive **noun phrase**

Chunking 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/S soared/N at/N Boeing/S Co./C ,/N easily/N topping/N forecasts/S on/N Wall/S Street/C ,/N as/N their/S CEO/C Alan/C Mulally/C announced/N first/S quarter/C results/C ./N

N = Not part of noun-phrase
S = Start noun-phrase
C = Continue noun-phrase

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

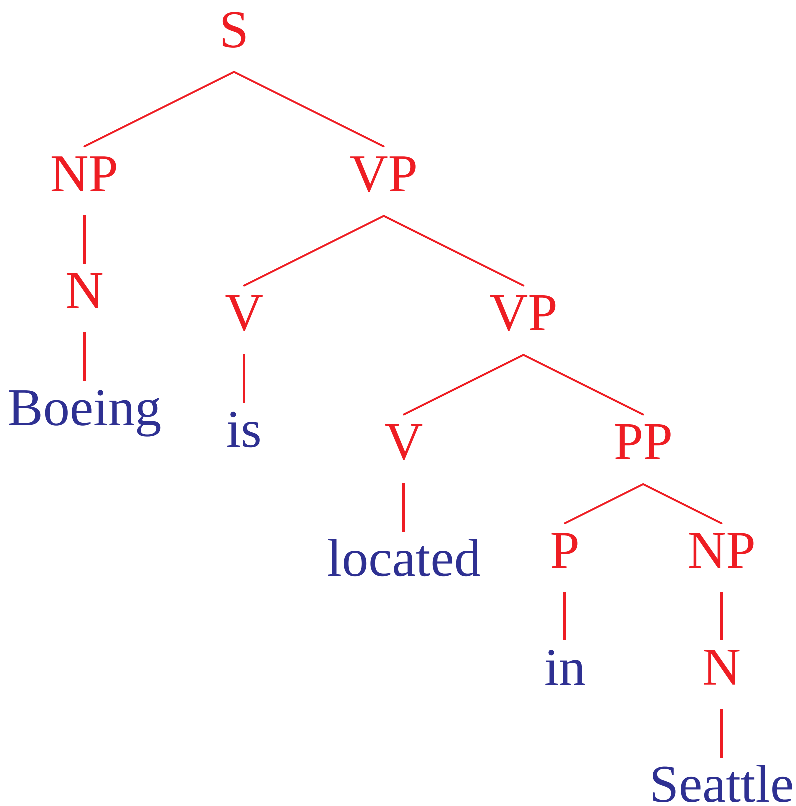
NA = No entity
SC = Start Company
CC = Continue Company
SL = Start Location
CL = Continue Location
. . .

Parsing (Syntactic Structure)

INPUT:

Boeing is located in Seattle.

OUTPUT:



Machine Translation

INPUT:

Boeing is located in Seattle. Alan Mulally is the CEO.

OUTPUT:

Boeing ist in Seattle. Alan Mulally ist der CEO.

Summary

Problem	Well-Studied Learning Approaches?	Class of Problem
Named entity extraction	Yes	Tagging
Relationships between entities	A little	Parsing
More complex relationships	No	??
Part-of-speech tagging	Yes	Tagging
Chunking	Yes	Tagging
Syntactic Structure	Yes	Parsing
Machine translation	Yes	??

Techniques Covered in this Tutorial

- Log-linear (maximum-entropy) taggers
- Probabilistic context-free grammars (PCFGs)
- PCFGs with enriched non-terminals
- Discriminative methods:
  - Conditional Markov Random Fields
  - Perceptron algorithms
  - Kernels over NLP structures

Log-Linear Taggers: Notation

- Set of possible words = $\mathcal{V}$, possible tags = $\mathcal{T}$
- Word sequence $w_{[1:n]} = [w_1, w_2 \ldots w_n]$
- Tag sequence $t_{[1:n]} = [t_1, t_2 \ldots t_n]$
- Training data is n tagged sentences, where the i'th sentence is of length n_i

$$(w^i_{[1:n_i]}, t^i_{[1:n_i]}) \text{ for } i = 1 \ldots n$$

Log-Linear Taggers: Independence Assumptions

- The basic idea

$$P(t_{[1:n]} \mid w_{[1:n]}) = \prod_{j=1}^{n} P(t_j \mid t_{j-1} \ldots t_1, w_{[1:n]}, j)$$ Chain rule

$$= \prod_{j=1}^{n} P(t_j \mid t_{j-1}, t_{j-2}, w_{[1:n]}, j)$$ Independence assumptions

- Two questions:

1. How to parameterize $P(t_j \mid t_{j-1}, t_{j-2}, w_{[1:n]}, j)$?
2. How to find $\arg\max_{t_{[1:n]}} P(t_{[1:n]} \mid w_{[1:n]})$?

The Parameterization Problem

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 ??

- Need to learn a function from (context, tag) pairs to a probability $P(tag|context)$

Representation: Histories

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

Representation: 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 .

- **History** $= \langle t_{-1}, t_{-2}, w_{[1:n]}, j \rangle$
- $t_{-1}, t_{-2} =$ DT, JJ
- $w_{[1:n]} = \langle Hispaniola, quickly, became, \ldots \rangle$
- $j = 6$

Feature–Vector Representations

- Take a history/tag pair (h, t).
- $\phi_s(h, t)$ for $s = 1 \ldots d$ are **features** representing tagging decision t in context h.

$$\phi_{1000}(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ is } \texttt{base} \\ & \text{and } t = \textsf{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{1001}(h, t) = \begin{cases} 1 & \text{if } \langle t_{-2}, t_{-1}, t \rangle = \langle \text{DT, JJ, VB} \rangle \\ 0 & \text{otherwise} \end{cases}$$

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

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

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

Feature–Vector Representations

- Take a history/tag pair (h, t).
- $\phi_s(h, t)$ for $s = 1 \dots d$ are **features** representing tagging decision t in context h.

Example: POS Tagging [Ratnaparkhi 96]

- **Word/tag features**

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

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

- **Contextual Features**

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

Part-of-Speech (POS) Tagging [Ratnaparkhi 96]

- Word/tag features

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

- Spelling features

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

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

Ratnaparkhi's POS Tagger

- Contextual Features

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

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

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

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

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

Log-Linear (Maximum-Entropy) Models

- Take a history/tag pair (h, t).
- $\phi_s(h, t)$ for $s = 1 \ldots d$ are **features**
- $\mathbf{W}_s$ for $s = 1 \ldots d$ are parameters
- Parameters define a conditional distribution

$$P(t|h) = \frac{e^{\sum_s \mathbf{W}_s \phi_s(h,t)}}{Z(h, \mathbf{W})}$$

where

$$Z(h, \mathbf{W}) = \sum_{t' \in \mathcal{T}} e^{\sum_s \mathbf{W}_s \phi_s(h,t')}$$

Log-Linear (Maximum Entropy) Models

- Word sequence $w_{[1:n]} = [w_1, w_2 \dots w_n]$
- Tag sequence $t_{[1:n]} = [t_1, t_2 \dots t_n]$
- Histories $h_i = \langle t_{i-1}, t_{i-2}, w_{[1:n]}, i \rangle$

$$\log P(t_{[1:n]} \mid w_{[1:n]}) = \sum_{i=1}^{n} \log P(t_i \mid h_i)$$

$$= \underbrace{\sum_{i=1}^{n} \sum_{s} \mathbf{W}_s \phi_s(h_i, t_i)}_{\text{Linear Score}} \quad - \quad \underbrace{\sum_{i=1}^{n} \log Z(h_i, \mathbf{W})}_{\text{Local Normalization Terms}}$$

Log-Linear Models

- Word sequence $w_{[1:n]} = [w_1, w_2 \dots w_n]$
- Tag sequence $t_{[1:n]} = [t_1, t_2 \dots t_n]$

$$\log P(t_{[1:n]} \mid w_{[1:n]}) = \sum_{j=1}^{n} \log P(t_j \mid h_j)$$

$$= \sum_{j=1}^{n} \sum_{s} \mathbf{W}_s \phi_s(h_j, t_j) - \sum_{j=1}^{n} \log Z(h_j, \mathbf{W})$$

where

$$h_j = \langle t_{j-2}, t_{j-1}, w_{[1:n]}, j \rangle$$

Log-Linear Models

- Parameter estimation:
  Maximize likelihood of training data through gradient descent, iterative scaling

- Search for $\arg\max_{t_{[1:n]}} P(t_{[1:n]} \mid w_{[1:n]})$:
  Dynamic programming, $O(n|\mathcal{T}|^3)$ complexity

- Experimental results:
  - Almost 97% accuracy for POS tagging [Ratnaparkhi 96]
  - Over 90% accuracy for named-entity extraction [Borthwick et. al 98]
  - Around 93% precision/recall for NP chunking
  - Better results than an HMM for FAQ segmentation [McCallum et al. 2000]

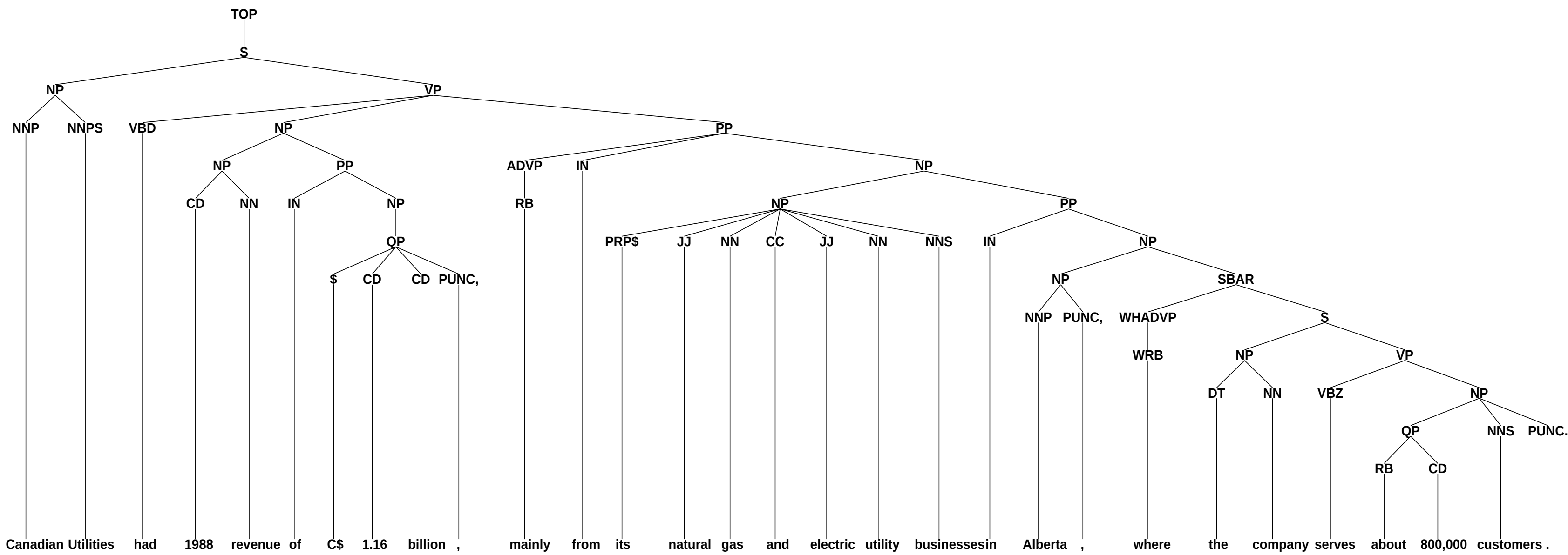
Techniques Covered in this Tutorial

- Log-linear (maximum-entropy) taggers
- Probabilistic context-free grammars (PCFGs)
- PCFGs with enriched non-terminals
- Discriminative methods:
  - Conditional Markov Random Fields
  - Perceptron algorithms
  - Kernels over NLP structures

Data for Parsing Experiments

- Penn WSJ Treebank = 50,000 sentences with associated trees
- Usual set-up: 40,000 training sentences, 2400 test sentences

An example tree:

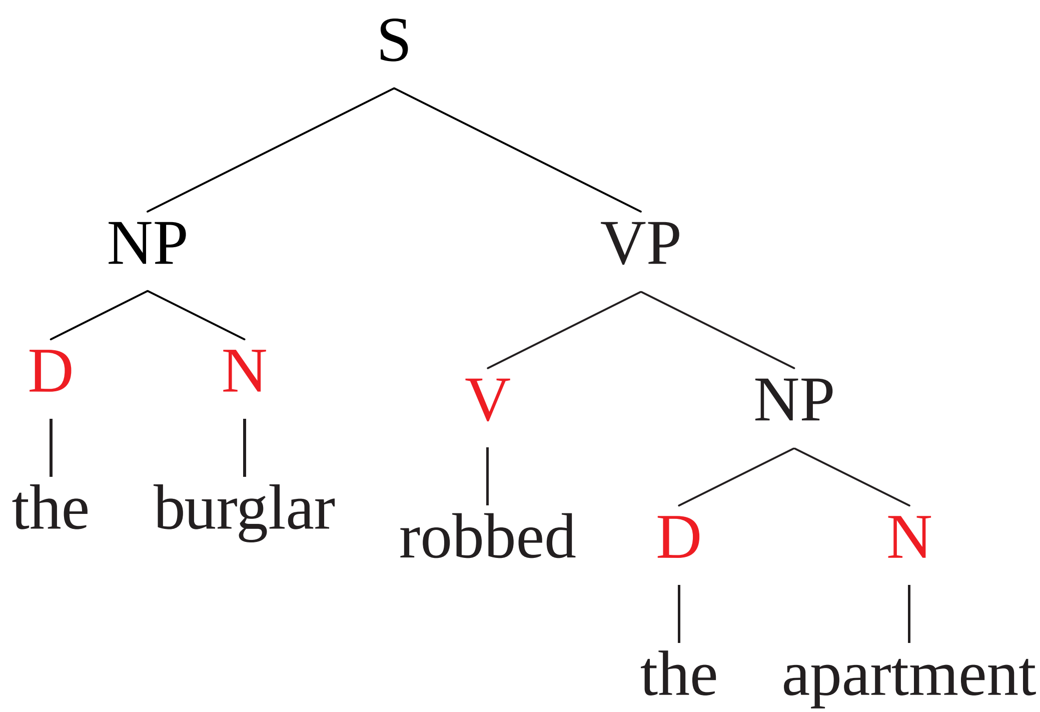


Canadian Utilities had 1988 revenue of C$ 1.16 billion , mainly from its natural gas and electric utility businesses in Alberta , where the company serves about 800,000 customers .

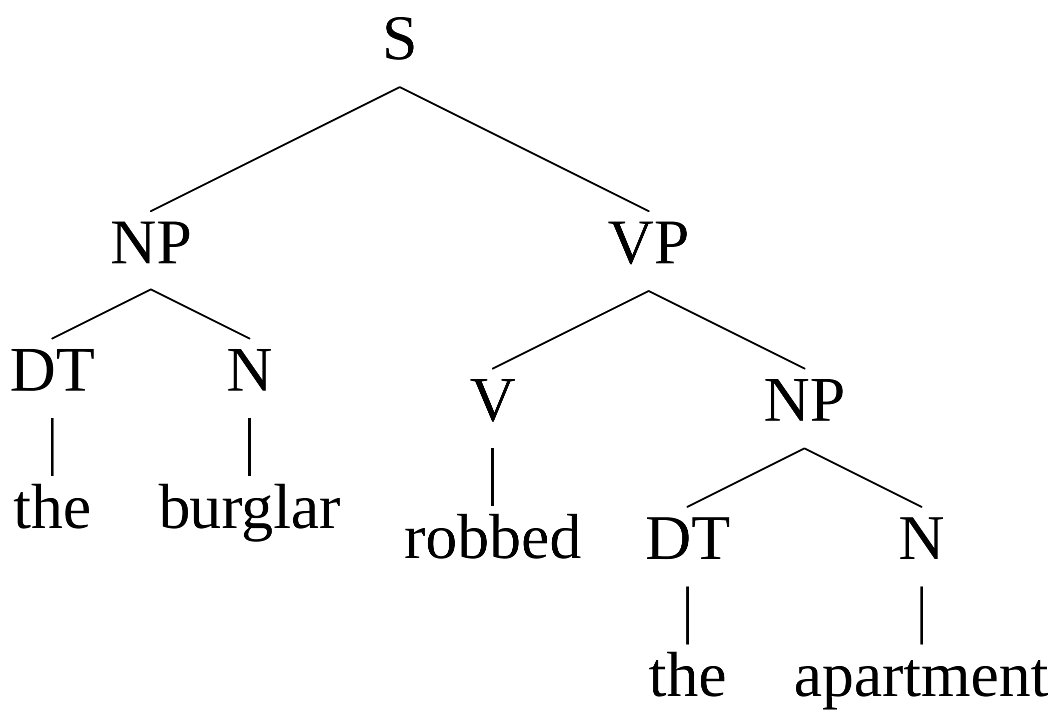
The Information Conveyed by Parse Trees

1) Part of speech for each word

(N = noun, V = verb, D = determiner)



2) Phrases

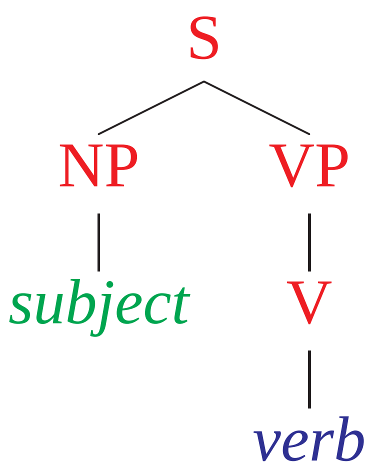


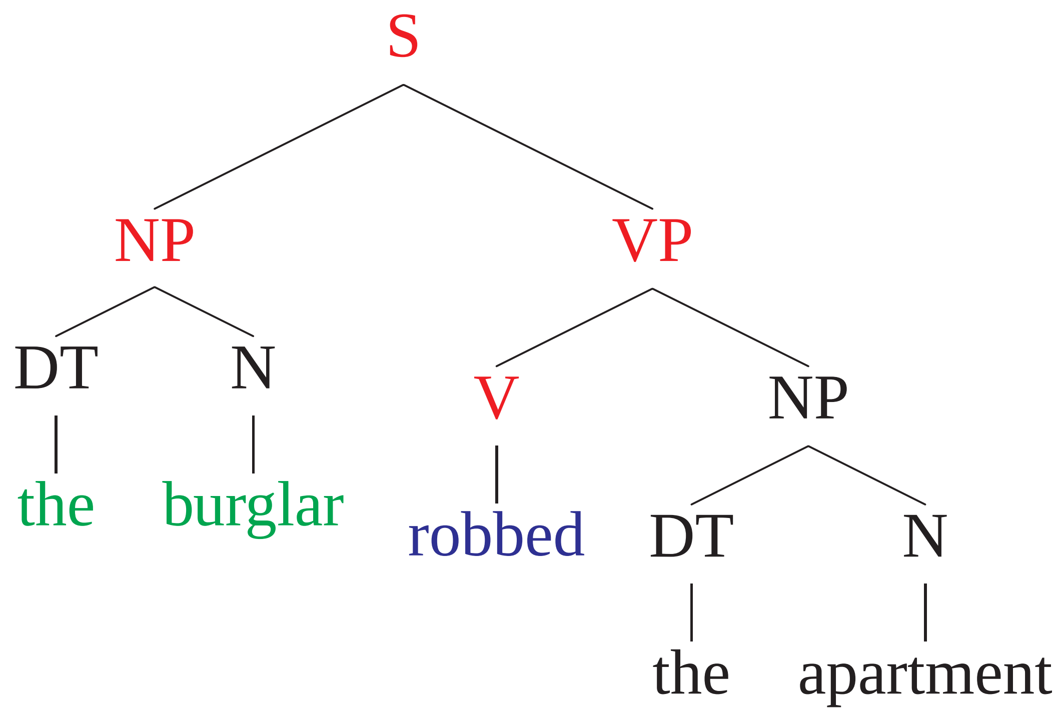
Noun Phrases (NP): “the burglar”, “the apartment”

Verb Phrases (VP): “robbed the apartment”

Sentences (S): “the burglar robbed the apartment”

3) Useful Relationships





⇒ "the burglar" is the subject of "robbed"

An Example Application: Machine Translation

- English word order is *subject – verb – object*

- Japanese word order is *subject – object – verb*

English: IBM bought Lotus
Japanese: *IBM Lotus bought*

English: Sources said that IBM bought Lotus yesterday
Japanese: *Sources yesterday IBM Lotus bought that said*

Context-Free Grammars

[Hopcroft and Ullman 1979]
A context free grammar $G = (N, \Sigma, R, S)$ where:

- N is a set of non-terminal symbols
- Σ is a set of terminal symbols
- R is a set of rules of the form $X \rightarrow Y_1 Y_2 \ldots Y_n$ for $n \geq 0$, $X \in N$, $Y_i \in (N \cup \Sigma)$
- $S \in N$ is a distinguished start symbol

A Context-Free Grammar for English

N = {S, NP, VP, PP, D, Vi, Vt, N, P}
S = S
Σ = {sleeps, saw, man, woman, telescope, the, with, in}

$R =$

S	$\Rightarrow$	NP	VP
VP	$\Rightarrow$	Vi	
VP	$\Rightarrow$	Vt	NP
VP	$\Rightarrow$	VP	PP
NP	$\Rightarrow$	D	N
NP	$\Rightarrow$	NP	PP
PP	$\Rightarrow$	P	NP

Vi	$\Rightarrow$	sleeps
Vt	$\Rightarrow$	saw
N	$\Rightarrow$	man
N	$\Rightarrow$	woman
N	$\Rightarrow$	telescope
D	$\Rightarrow$	the
P	$\Rightarrow$	with
P	$\Rightarrow$	in

Note: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, D=determiner, Vi=intransitive verb, Vt=transitive verb, N=noun, P=preposition

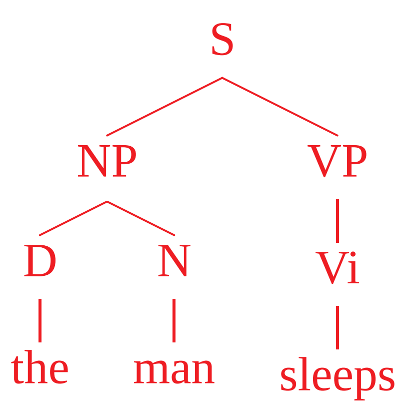
Left-Most Derivations

A left-most derivation is a sequence of strings $s_1 \ldots s_n$, where

- $s_1 = S$, the start symbol
- $s_n \in \Sigma^*$, i.e. s_n is made up of terminal symbols only
- Each s_i for $i = 2 \ldots n$ is derived from s_{i-1} by picking the left-most non-terminal X in s_{i-1} and replacing it by some β where $X \rightarrow \beta$ is a rule in R

For example: [S], [NP VP], [D N VP], [the N VP], [the man VP], [the man Vi], [the man sleeps]

Representation of a derivation as a tree:



Notation

- We use $\mathcal{D}$ to denote the set of all left-most derivations (trees) allowed by a grammar
- We use $\mathcal{D}(x)$ for a string $x \in \Sigma^*$ to denote the set of all derivations whose final string ("yield") is x.

The Problem with Parsing: Ambiguity

INPUT:

She announced a program to promote safety in trucks and vans

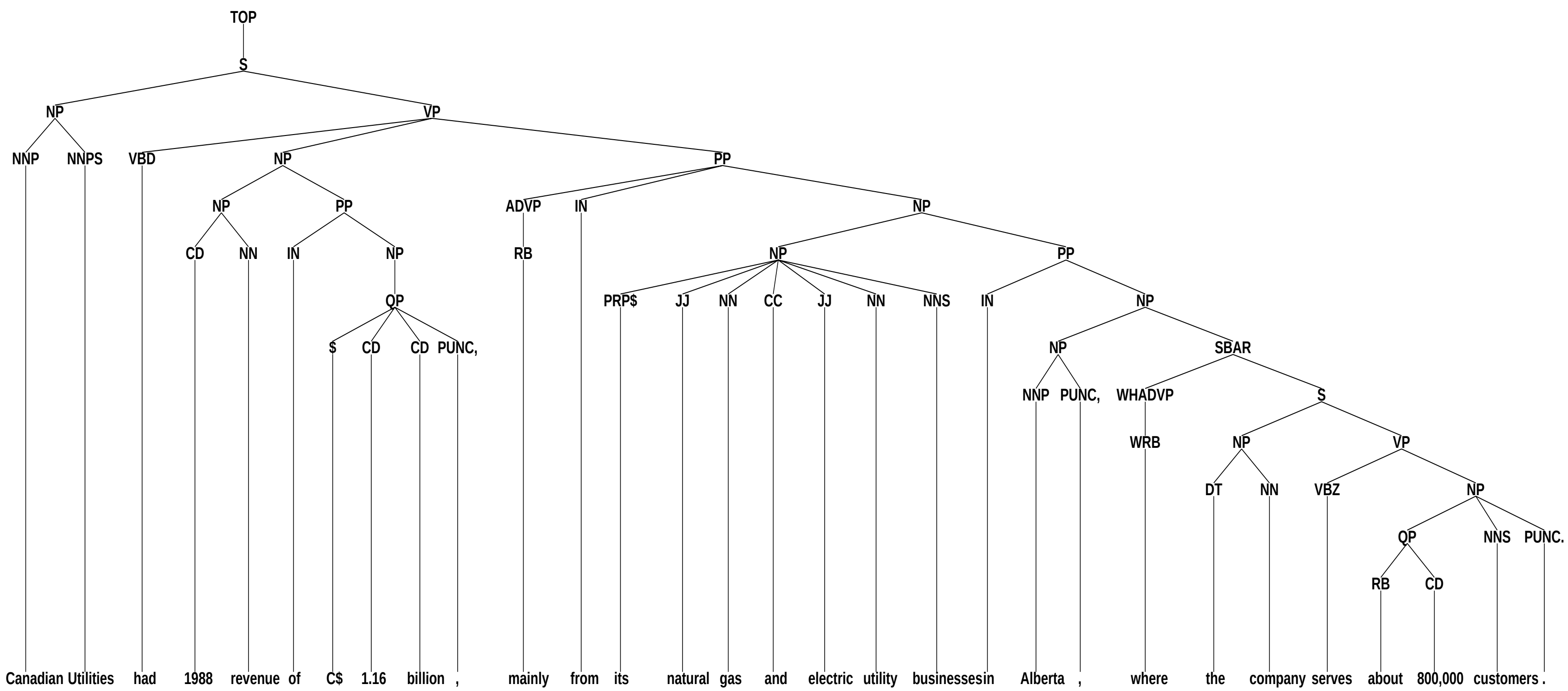
$\Downarrow$

POSSIBLE OUTPUTS:

And there are more...

An Example Tree

Canadian Utilities had 1988 revenue of C$ 1.16 billion , mainly from its natural gas and electric utility businesses in Alberta , where the company serves about 800,000 customers .



A Probabilistic Context-Free Grammar

S	$\Rightarrow$	NP	VP	1.0
VP	$\Rightarrow$	Vi		0.4
VP	$\Rightarrow$	Vt	NP	0.4
VP	$\Rightarrow$	VP	PP	0.2
NP	$\Rightarrow$	D	N	0.3
NP	$\Rightarrow$	NP	PP	0.7
PP	$\Rightarrow$	P	NP	1.0

Vi	$\Rightarrow$	sleeps	1.0
Vt	$\Rightarrow$	saw	1.0
N	$\Rightarrow$	man	0.7
N	$\Rightarrow$	woman	0.2
N	$\Rightarrow$	telescope	0.1
D	$\Rightarrow$	the	1.0
P	$\Rightarrow$	with	0.5
P	$\Rightarrow$	in	0.5

- Probability of a tree with rules $\alpha_i \rightarrow \beta_i$ is $\prod_i P(\alpha_i \rightarrow \beta_i | \alpha_i)$
- Maximum Likelihood estimation

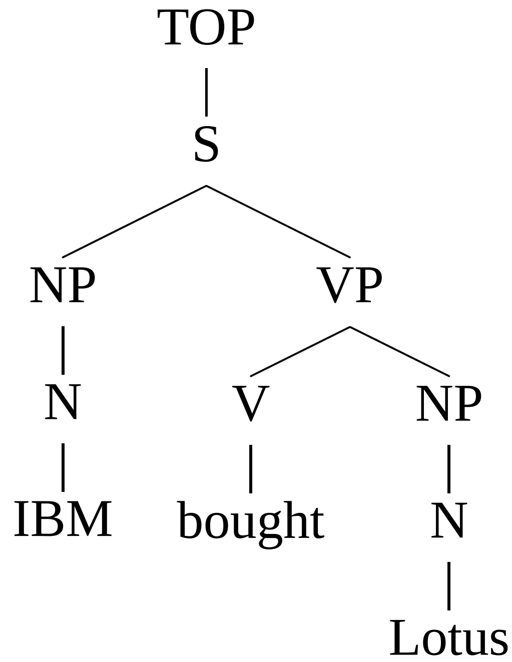
$$P(\text{VP} \Rightarrow \text{V NP} \mid \text{VP}) = \frac{Count(\text{VP} \Rightarrow \text{V NP})}{Count(\text{VP})}$$

PCFGs

[Booth and Thompson 73] showed that a CFG with rule probabilities correctly defines a distribution over the set of derivations $\mathcal{D}$ provided that:

1. The rule probabilities define conditional distributions over the different ways of rewriting each non-terminal.

2. A technical condition on the rule probabilities ensuring that the probability of the derivation terminating in a finite number of steps is 1. (This condition is not really a practical concern.)



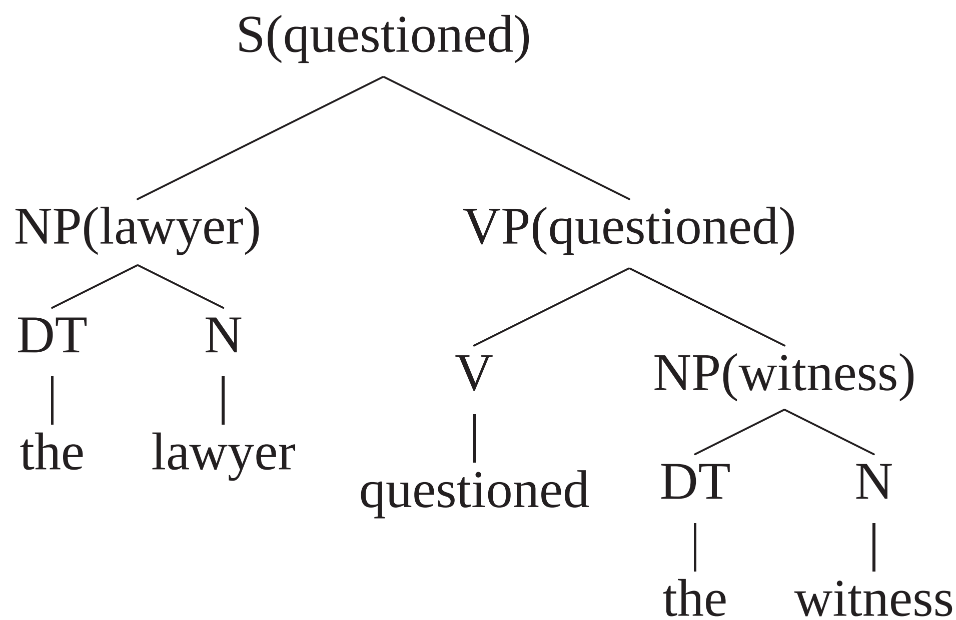
$$
\begin{aligned}
\text{PROB} = \quad & P(\text{TOP} \rightarrow \text{S}) \\
& \times P(\text{S} \rightarrow \text{NP VP}) \qquad \times P(\text{N} \rightarrow IBM) \\
& \times P(\text{VP} \rightarrow \text{V NP}) \qquad \times P(\text{V} \rightarrow bought) \\
& \times P(\text{NP} \rightarrow \text{N}) \qquad \times P(\text{N} \rightarrow Lotus) \\
& \times P(\text{NP} \rightarrow \text{N})
\end{aligned}
$$

The SPATTER Parser: (Magerman 95;Jelinek et al 94)

- For each rule, identify the “head” child

S	$\Rightarrow$	NP	**VP**
VP	$\Rightarrow$	**V**	NP
NP	$\Rightarrow$	DT	**N**

- Add word to each non-terminal

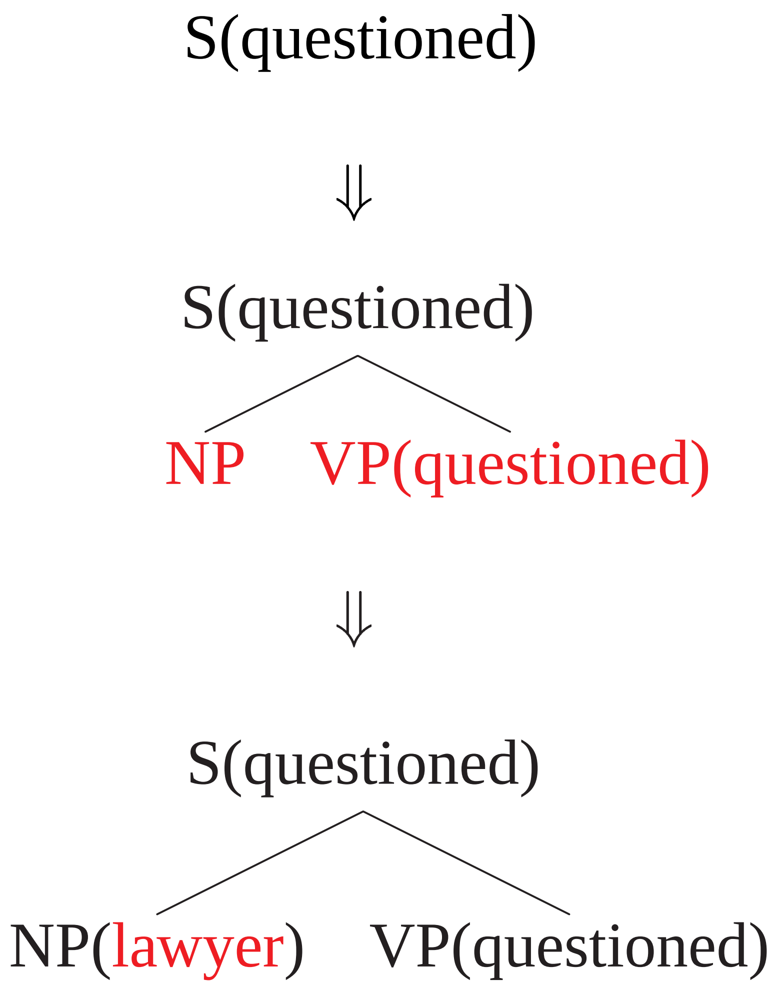


A Lexicalized PCFG

S(questioned)	$\Rightarrow$	NP(lawyer)	VP(questioned)	??
VP(questioned)	$\Rightarrow$	V(questioned)	NP(witness)	??
NP(lawyer)	$\Rightarrow$	D(the)	N(lawyer)	??
NP(witness)	$\Rightarrow$	D(the)	N(witness)	??

- The big question: how to estimate rule probabilities??

CHARNIAK (1997)



$P(\text{NP VP} \mid \text{S(questioned)})$

$P(\text{lawyer} \mid \text{S,VP,NP, questioned})$

Smoothed Estimation

$$P(\text{NP VP} \mid \text{S(questioned)}) =$$

$$\lambda_1 \times \frac{Count(\text{S(questioned)}\rightarrow\text{NP VP})}{Count(\text{S(questioned)})}$$

$$+\lambda_2 \times \frac{Count(\text{S}\rightarrow\text{NP VP})}{Count(\text{S})}$$

- Where $0 \leq \lambda_1, \lambda_2 \leq 1$, and $\lambda_1 + \lambda_2 = 1$

Smoothed Estimation

$$P(\text{lawyer} \mid \text{S,NP,VP,questioned}) =$$

$$\lambda_1 \times \frac{Count(\text{lawyer} \mid \text{S,NP,VP,questioned})}{Count(\text{S,NP,VP,questioned})}$$

$$+\lambda_2 \times \frac{Count(\text{lawyer} \mid \text{S,NP,VP})}{Count(\text{S,NP,VP})}$$

$$+\lambda_3 \times \frac{Count(\text{lawyer} \mid \text{NP})}{Count(\text{NP})}$$

- Where $0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1$, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$

$$P(\text{NP(lawyer) VP(questioned)} \mid \text{S(questioned)}) =$$

$$\left(\lambda_1 \times \frac{Count(\text{S(questioned)}\rightarrow\text{NP VP})}{Count(\text{S(questioned)})}\right.$$

$$\left.+\lambda_2 \times \frac{Count(\text{S}\rightarrow\text{NP VP})}{Count(\text{S})}\right)$$

$$\times \left(\lambda_1 \times \frac{Count(\text{lawyer} \mid \text{S,NP,VP,questioned})}{Count(\text{S,NP,VP,questioned})}\right.$$

$$+\lambda_2 \times \frac{Count(\text{lawyer} \mid \text{S,NP,VP})}{Count(\text{S,NP,VP})}$$

$$\left.+\lambda_3 \times \frac{Count(\text{lawyer} \mid \text{NP})}{Count(\text{NP})}\right)$$

Lexicalized Probabilistic Context-Free Grammars

- Transformation to lexicalized rules

  $S \rightarrow NP\ VP$

  vs. $S(questioned) \rightarrow NP(lawyer)\ VP(questioned)$

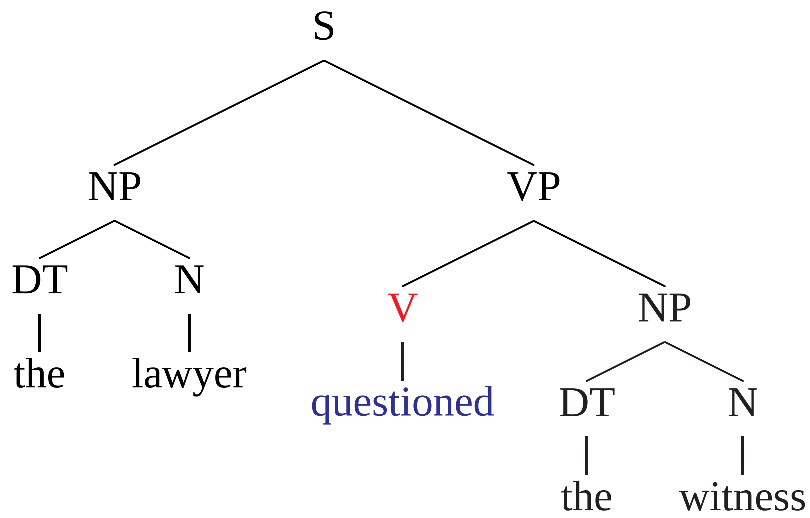
- Smoothed estimation techniques “blend” different counts

- Search for most probable tree through dynamic programming

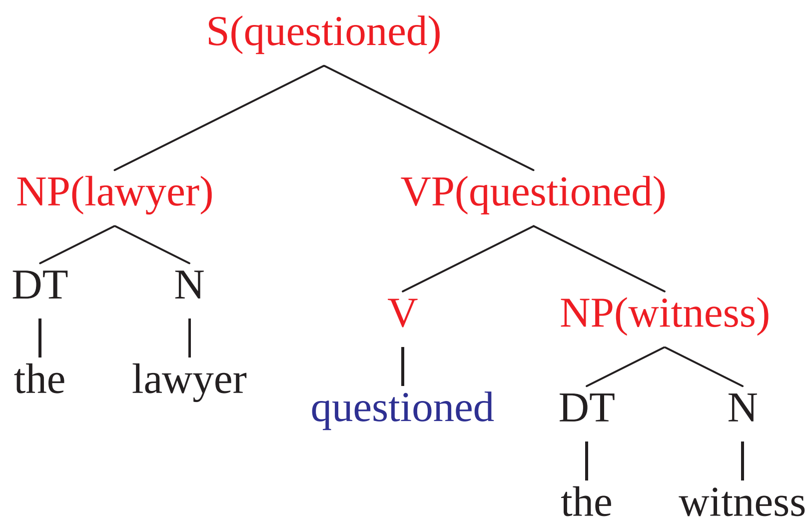
- Perform vastly better than PCFGs (88% vs. 73% accuracy)

Independence Assumptions

- PCFGs



- Lexicalized PCFGs



Results

Method	Accuracy
PCFGs (Charniak 97)	73.0%
Conditional Models – Decision Trees (Magerman 95)	84.2%
Lexical Dependencies (Collins 96)	85.5%
Conditional Models – Logistic (Ratnaparkhi 97)	86.9%
Generative Lexicalized Model (Charniak 97)	86.7%
Generative Lexicalized Model (Collins 97)	88.2%
Logistic-inspired Model (Charniak 99)	89.6%
Boosting (Collins 2000)	89.8%

- Accuracy = average recall/precision

Parsing for Information Extraction: Relationships between Entities

INPUT:

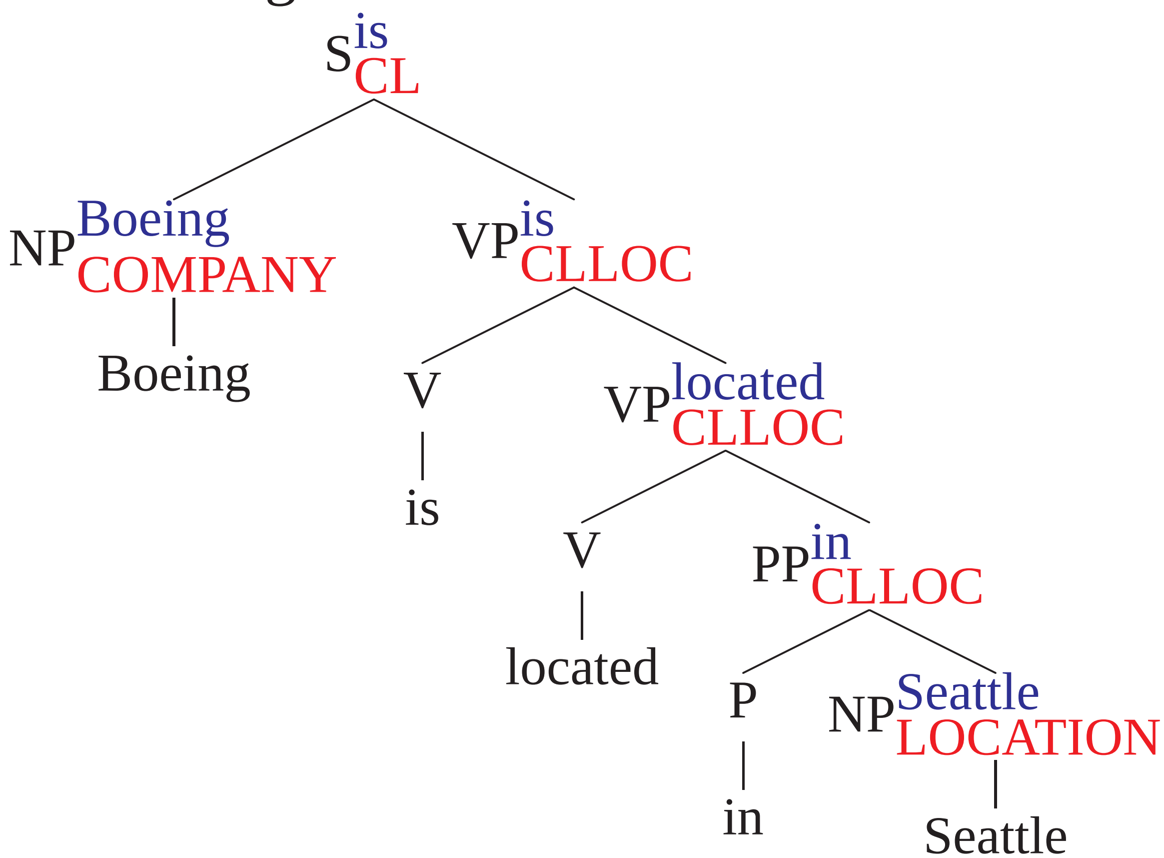
Boeing is located in Seattle.

OUTPUT:

{Relationship = Company-Location
Company = Boeing
Location = Seattle}

A Generative Model (Miller et. al)

[Miller et. al 2000] use non-terminals to carry lexical items and semantic tags



PP^{in}_{CLLOC} ← lexical head ← semantic tag

A Generative Model [Miller et. al 2000]

We're now left with an even more complicated estimation problem,

$$P(S^{is}_{CL} \Rightarrow NP^{Boeing}_{COMPANY}\ VP^{is}_{CLLOC})$$

See [Miller et. al 2000] for the details

- Parsing algorithm recovers annotated trees
  $\Rightarrow$ Simultaneously recovers syntactic structure and named entity relationships
- Accuracy (precision/recall) is greater than 80% in recovering relations

Techniques Covered in this Tutorial

- Log-linear (maximum-entropy) taggers
- Probabilistic context-free grammars (PCFGs)
- PCFGs with enriched non-terminals
- Discriminative methods:
  - Conditional Markov Random Fields
  - Perceptron algorithms
  - Kernels over NLP structures

Linear Models for Parsing and Tagging

- Three components:

  GEN is a function from a string to a set of candidates

  $\mathbf{\Phi}$ maps a candidate to a feature vector

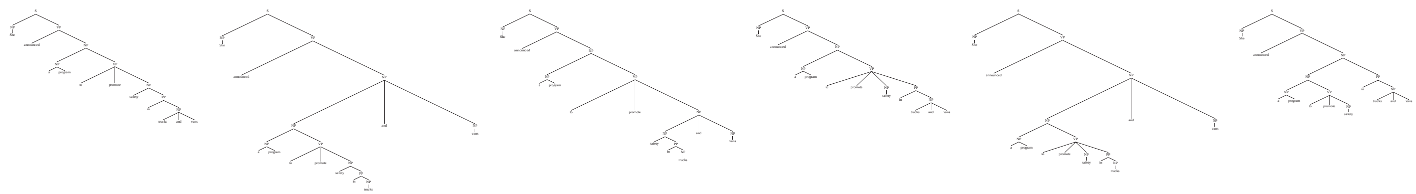
  W is a parameter vector

Component 1: GEN

- **GEN** enumerates a set of **candidates** for a sentence

She announced a program to promote safety in trucks and vans

$\Downarrow$ **GEN**



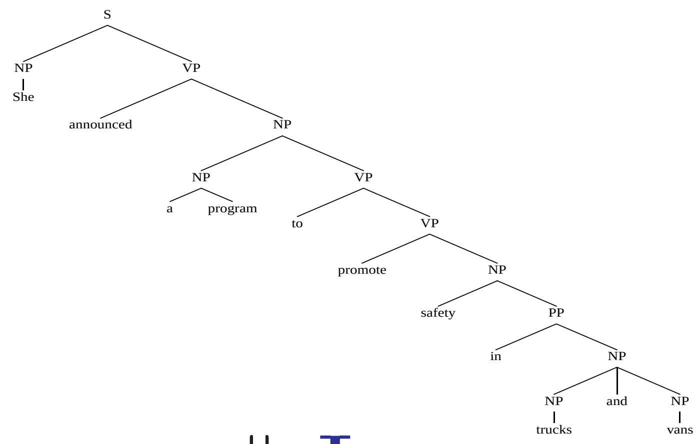
Examples of GEN

- A context-free grammar
- A finite-state machine
- Top N most probable analyses from a probabilistic grammar

Component 2: Φ

- Φ maps a candidate to a **feature vector** $\in \mathbb{R}^d$

- Φ defines the **representation** of a candidate

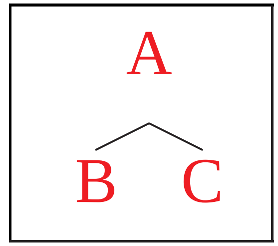


$$\Downarrow \Phi$$

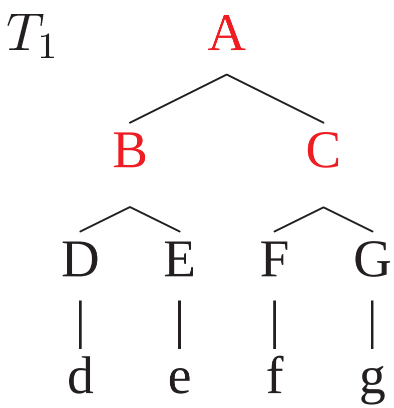
$$\langle 1, 0, 2, 0, 0, 15, 5 \rangle$$

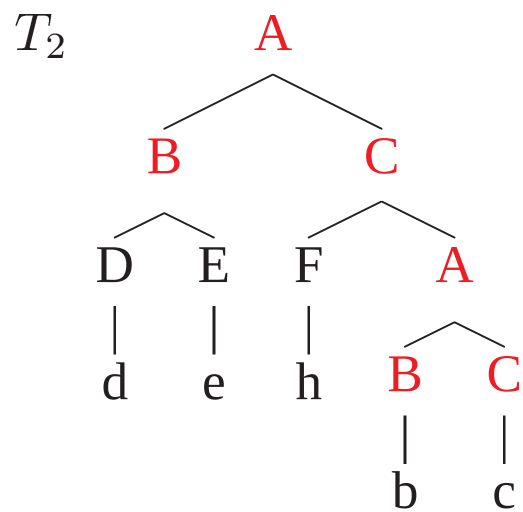
Features

- A “feature” is a function on a structure, e.g.,

$h(x) =$ Number of times  is seen in x

---



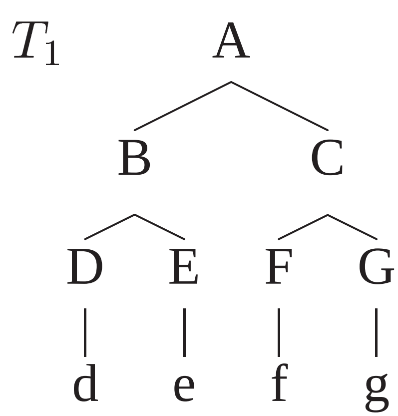


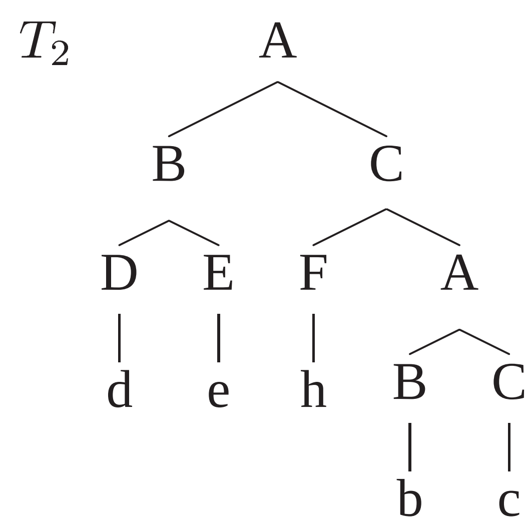
$$h(T_1) = 1 \qquad h(T_2) = 2$$

Feature Vectors

- A set of functions $h_1 \ldots h_d$ define a **feature vector**

$$\mathbf{\Phi}(x) = \langle h_1(x), h_2(x) \ldots h_d(x) \rangle$$



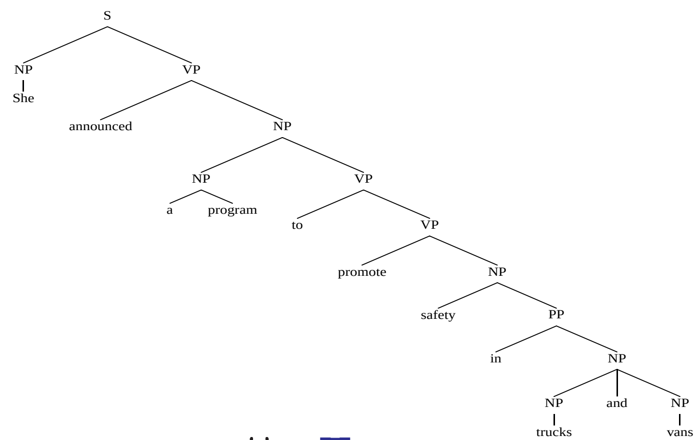


$$\mathbf{\Phi}(T_1) = \langle 1, 0, 0, 3 \rangle \qquad \mathbf{\Phi}(T_2) = \langle 2, 0, 1, 1 \rangle$$

Component 3: W

- **W** is a **parameter vector** $\in \mathbb{R}^d$
- Φ and **W** together map a candidate to a real-valued score

---



$$\Downarrow \Phi$$

$$\langle 1, 0, 2, 0, 0, 15, 5 \rangle$$

$$\Downarrow \Phi \cdot \mathbf{W}$$

$$\langle 1, 0, 2, 0, 0, 15, 5 \rangle \cdot \langle 1.9, -0.3, 0.2, 1.3, 0, 1.0, -2.3 \rangle = 5.8$$

Putting it all Together

- $\mathcal{X}$ is set of sentences, $\mathcal{Y}$ is set of possible outputs (e.g. trees)

- Need to learn a function $F : \mathcal{X} \rightarrow \mathcal{Y}$

- **GEN**, $\mathbf{\Phi}$, **W** define

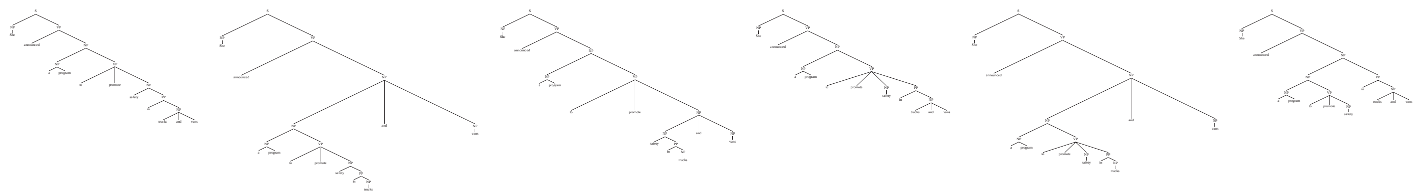
$$F(x) = \arg\max_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(y) \cdot \mathbf{W}$$

Choose the highest scoring tree as the most plausible structure

- Given examples (x_i, y_i), how to set **W**?

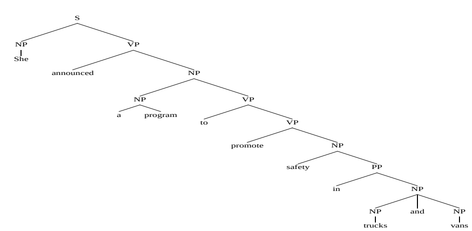
She announced a program to promote safety in trucks and vans

$\Downarrow$ **GEN**



$\Downarrow \boldsymbol{\Phi}$	$\Downarrow \boldsymbol{\Phi}$	$\Downarrow \boldsymbol{\Phi}$	$\Downarrow \boldsymbol{\Phi}$	$\Downarrow \boldsymbol{\Phi}$	$\Downarrow \boldsymbol{\Phi}$
$\langle 1, 1, 3, 5\rangle$	$\langle 2, 0, 0, 5\rangle$	$\langle 1, 0, 1, 5\rangle$	$\langle 0, 0, 3, 0\rangle$	$\langle 0, 1, 0, 5\rangle$	$\langle 0, 0, 1, 5\rangle$
$\Downarrow \boldsymbol{\Phi} \cdot \mathbf{W}$	$\Downarrow \boldsymbol{\Phi} \cdot \mathbf{W}$	$\Downarrow \boldsymbol{\Phi} \cdot \mathbf{W}$	$\Downarrow \boldsymbol{\Phi} \cdot \mathbf{W}$	$\Downarrow \boldsymbol{\Phi} \cdot \mathbf{W}$	$\Downarrow \boldsymbol{\Phi} \cdot \mathbf{W}$
13.6	12.2	12.1	3.3	9.4	11.1

$\Downarrow$ arg max



Markov Random Fields

- Parameters **W** define a conditional distribution over candidates:

$$P(y_i \mid x_i, \mathbf{W}) = \frac{e^{\mathbf{\Phi}(y_i)\cdot\mathbf{W}}}{\sum_{y \in \mathbf{GEN}(x_i)} e^{\mathbf{\Phi}(y)\cdot\mathbf{W}}}$$

- Gaussian prior: $\log P(\mathbf{W}) \sim -C||\mathbf{W}||^2/2$

- MAP parameter estimates maximise

$$\sum_i \log \frac{e^{\mathbf{\Phi}(y_i)\cdot\mathbf{W}}}{\sum_{y \in \mathbf{GEN}(x_i)} e^{\mathbf{\Phi}(y)\cdot\mathbf{W}}} - C\frac{||\mathbf{W}||^2}{2}$$

Note: This is a “globally normalised” model

Markov Random Fields Example 1: [Johnson et. al 1999]

GEN is the set of parses for a sentence with a hand-crafted grammar (a Lexical Functional Grammar)

Φ can include arbitrary features of the candidate parses

W is estimated using conjugate gradient descent

Markov Random Fields Example 2: [Lafferty et al. 2001]

Going back to tagging:

- Inputs x are sentences $w_{[1:n]}$
- $\mathbf{GEN}(w_{[1:n]}) = \mathcal{T}^n$ i.e. all tag sequences of length n
- Global representations $\mathbf{\Phi}$ are composed from local feature vectors ϕ

$$\mathbf{\Phi}(w_{[1:n]}, t_{[1:n]}) = \sum_{j=1}^{n} \phi(h_j, t_j)$$

where $h_j = \langle t_{j-2}, t_{j-1}, w_{[1:n]}, j \rangle$

Markov Random Fields Example 2: [Lafferty et al. 2001]

- Typically, local features are indicator functions, e.g.,

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

- and global features are then counts,

$\mathbf{\Phi}_{101}(w_{[1:n]}, t_{[1:n]}) =$ Number of times a word ending in `ing` is tagged as `VBG` in $(w_{[1:n]}, t_{[1:n]})$

Markov Random Fields Example 2: [Lafferty et al. 2001]

Conditional random fields are globally normalised models:

$$\log P(t_{[1:n]} \mid w_{[1:n]}) = \mathbf{\Phi}(w_{[1:n]}, t_{[1:n]}) \cdot \mathbf{W} - \log Z(w_{[1:n]}, \mathbf{W})$$

$$= \underbrace{\sum_{j=1}^{n} \sum_{s} \mathbf{W}_s \phi_s(h_j, t_j)}_{\text{Linear model}} - \underbrace{\log Z(w_{[1:n]}, \mathbf{W})}_{\text{Normalization}}$$

where $Z(w_{[1:n]}, \mathbf{W}) = \sum_{t_{[1:n]} \in \mathcal{T}^n} e^{\mathbf{\Phi}(w_{[1:n]}, t_{[1:n]}) \cdot \mathbf{W}}$

Log-linear taggers (see earlier part of the tutorial) are locally normalised models:

$$\log P(t_{[1:n]} \mid w_{[1:n]}) = \underbrace{\sum_{j=1}^{n} \sum_{s} \mathbf{W}_s \phi_s(h_j, t_j)}_{\text{Linear Model}} - \underbrace{\sum_{j=1}^{n} \log Z(h_j, \mathbf{W})}_{\text{Local Normalization}}$$

Problems with Locally Normalized Models

- "Label bias" problem [Lafferty et al. 2001]
  See also [Klein and Manning 2002]

- Example of a conditional distribution that locally normalized models can't capture (under bigram tag representation):

$$\text{a b c} \Rightarrow \begin{array}{ccccc} \text{A} & \text{—} & \text{B} & \text{—} & \text{C} \\ | & & | & & | \\ \text{a} & & \text{b} & & \text{c} \end{array} \quad \text{with } P(\text{A B C} \mid \text{a b c}) = 1$$

$$\text{a b e} \Rightarrow \begin{array}{ccccc} \text{A} & \text{—} & \text{D} & \text{—} & \text{E} \\ | & & | & & | \\ \text{a} & & \text{b} & & \text{e} \end{array} \quad \text{with } P(\text{A D E} \mid \text{a b e}) = 1$$

- Impossible to find parameters that satisfy

$$P(A \mid a) \times P(B \mid b, A) \times P(C \mid c, B) = 1$$

$$P(A \mid a) \times P(D \mid b, A) \times P(E \mid e, D) = 1$$

Markov Random Fields Example 2: [Lafferty et al. 2001]

Parameter Estimation

- Need to calculate gradient of the log-likelihood,

$$\frac{d}{d\mathbf{W}} \sum_i \log P(t^i_{[1:n_i]} \mid w^i_{[1:n_i]}, \mathbf{W})$$

$$= \frac{d}{d\mathbf{W}} \left(\sum_i \mathbf{\Phi}(w^i_{[1:n_i]}, t^i_{[1:n_i]}) \cdot \mathbf{W} - \sum_i \log Z(w^i_{[1:n_i]}, \mathbf{W}) \right)$$

$$= \sum_i \mathbf{\Phi}(w^i_{[1:n_i]}, t^i_{[1:n_i]})$$
$$- \sum_i \sum_{u_{[1:n_i]} \in \mathcal{T}^{n_i}} P(u_{[1:n_i]} \mid w^i_{[1:n_i]}, \mathbf{W}) \mathbf{\Phi}(w^i_{[1:n_i]}, u_{[1:n_i]})$$

Last term looks difficult to compute. But because $\mathbf{\Phi}$ is defined through "local" features, it can be calculated efficiently using dynamic programming. (Very similar problem to that solved by the EM algorithm for HMMs.) See [Lafferty et al. 2001].

Techniques Covered in this Tutorial

- Log-linear (maximum-entropy) taggers
- Probabilistic context-free grammars (PCFGs)
- PCFGs with enriched non-terminals
- Discriminative methods:
  - Conditional Markov Random Fields
  - Perceptron algorithms
  - Kernels over NLP structures

A Variant of the Perceptron Algorithm

Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{W} = 0$

Define: $F(x) = \text{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(y) \cdot \mathbf{W}$

Algorithm: For $t = 1 \ldots T$, $i = 1 \ldots n$

$z_i = F(x_i)$

If $(z_i \neq y_i)$ $\quad \mathbf{W} = \mathbf{W} + \mathbf{\Phi}(y_i) - \mathbf{\Phi}(z_i)$

Output: Parameters $\mathbf{W}$

Theory Underlying the Algorithm

- **Definition:** $\overline{\mathbf{GEN}}(x_i) = \mathbf{GEN}(x_i) - \{y_i\}$

- **Definition:** The training set is **separable with margin** δ, if there is a vector $\mathbf{U} \in \mathbb{R}^d$ with $||\mathbf{U}|| = 1$ such that

$$\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i) \quad \mathbf{U} \cdot \mathbf{\Phi}(y_i) - \mathbf{U} \cdot \mathbf{\Phi}(z) \geq \delta$$

Theorem: For any training sequence (x_i, y_i) which is separable with margin δ, then for the perceptron algorithm

$$\text{Number of mistakes} \leq \frac{R^2}{\delta^2}$$

where R is a constant such that $\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i) \quad ||\mathbf{\Phi}(y_i) - \mathbf{\Phi}(z)|| \leq R$

Proof: Direct modification of the proof for the classification case. See [Collins 2002]

More Theory for the Perceptron Algorithm

- Question 1: what if the data is not separable?

[Freund and Schapire 99] give a modified theorem for this case

- Question 2: performance on training data is all very well, but what about performance on new test examples?

Assume some distribution $P(x, y)$ underlying examples

Theorem [Helmbold and Warmuth 95]: For any distribution $P(x, y)$ generating examples, if e = expected number of mistakes of an online algorithm on a sequence of $m + 1$ examples, then a randomized algorithm trained on m samples will have probability $\frac{e}{m+1}$ of making an error on a newly drawn example from P.

[Freund and Schapire 99] use this to define the **Voted Perceptron**

Perceptron Algorithm 1: Tagging

- Score for a $(w_{[1:n]}, t_{[1:n]})$ pair is

$$F(w_{[1:n]}, t_{[1:n]}) = \sum_i \sum_s \mathbf{W}_s \phi_s(h_i, t_i) = \sum_s \mathbf{W}_s \mathbf{\Phi}_s(t_{[1:n]}, w_{[1:n]})$$

- Note: no normalization terms
- Note: $F(w_{[1:n]}, t_{[1:n]})$ is not a log probability
- Viterbi algorithm for

$$\arg \max_{t_{[1:n]} \in \mathcal{T}^n} F(w_{[1:n]}, t_{[1:n]})$$

Training the Parameters

Inputs: Training set $(w^i_{[1:n_i]}, t^i_{[1:n_i]})$ for $i = 1 \ldots n$.

Initialization: $\mathbf{W} = 0$

Algorithm: For $t = 1 \ldots T, i = 1 \ldots n$

$$z_{[1:n_i]} = \arg \max_{u_{[1:n_i]} \in \mathcal{T}^{n_i}} \sum_s \mathbf{W}_s \mathbf{\Phi}_s(w^i_{[1:n_i]}, u_{[1:n_i]})$$

$z_{[1:n_i]}$ is output on i'th sentence with current parameters

If $z_{[1:n_i]} \neq t^i_{[1:n_i]}$ then

$$\mathbf{W}_s = \mathbf{W}_s + \underbrace{\mathbf{\Phi}_s(w^i_{[1:n_i]}, t^i_{[1:n_i]})}_{\text{Correct tags' feature value}} - \underbrace{\mathbf{\Phi}_s(w^i_{[1:n_i]}, z_{[1:n_i]})}_{\text{Incorrect tags' feature value}}$$

Output: Parameter vector $\mathbf{W}$.

An Example

Say the correct tags for i'th sentence are

the/DT man/NN bit/VBD the/DT dog/NN

Under current parameters, output is

the/DT man/NN bit/NN the/DT dog/NN

Assume also that features track: (1) all bigrams; (2) word/tag pairs

Parameters incremented:

$$\langle \text{NN, VBD} \rangle, \langle \text{VBD, DT} \rangle, \langle \text{VBD} \rightarrow \text{bit} \rangle$$

Parameters decremented:

$$\langle \text{NN, NN} \rangle, \langle \text{NN, DT} \rangle, \langle \text{NN} \rightarrow \text{bit} \rangle$$

Experiments

- Wall Street Journal part-of-speech tagging data

  Perceptron = 2.89%, Max-ent = 3.28%
  (11.9% relative error reduction)

- [Ramshaw and Marcus 95] NP chunking data

  Perceptron = 93.63%, Max-ent = 93.29%
  (5.1% relative error reduction)

See [Collins 2002]

Perceptron Algorithm 2: Reranking Approaches

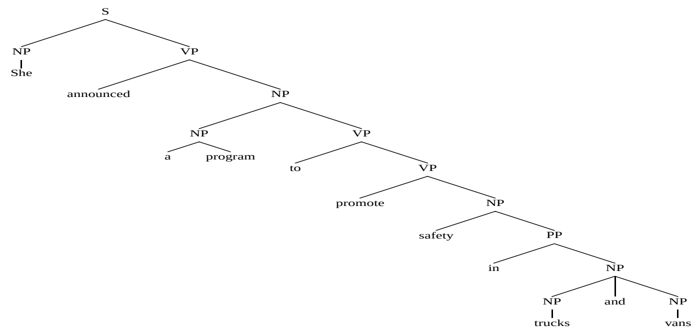
- **GEN** is the top n most probable candidates from a base model
  - Parsing: a lexicalized probabilistic context-free grammar
  - Tagging: “maximum entropy” tagger
  - Speech recognition: existing recogniser

Parsing Experiments

GEN Beam search used to parse training and test sentences: around 27 parses for each sentence

$\Phi = \langle L(x), h_1(x) \ldots h_m(x) \rangle$, where $L(x)$ = log-likelihood from first-pass parser, $h_1 \ldots h_m$ are $\approx 500,000$ indicator functions

$$e.g., \quad h_1(x) = \begin{cases} 1 & \text{if } x \text{ contains} \langle S \rightarrow NP \;\; VP \rangle \\ 0 & \text{otherwise} \end{cases}$$



$$\Downarrow \Phi$$

$$\langle -15.65, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, \ldots 1, 0, 0 \rangle$$

Named Entities

GEN Top 20 segmentations from a "maximum-entropy" tagger

$$\mathbf{\Phi} = \langle L(x), h_1(x) \ldots h_m(x) \rangle,$$

$$e.g., \quad h_1(x) = \begin{cases} 1 & \text{if } x \text{ contains a boundary = "[The"} \\ 0 & \text{otherwise} \end{cases}$$

Whether you're an aging flower child or a clueless [Gen-Xer], "[The Day They Shot John Lennon]," playing at the [Dougherty Arts Center], entertains the imagination.

$$\Downarrow \mathbf{\Phi}$$

$$\langle -3.17, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, \ldots 0, 1, 1 \rangle$$

Whether you're an aging flower child or a clueless [Gen-Xer], "[The Day They Shot John Lennon]," playing at the [Dougherty Arts Center], entertains the imagination.

$$\Downarrow \Phi$$

$$\langle -3.17, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, \ldots 0, 1, 1 \rangle$$

Whether you're an aging flower child or a clueless Gen-Xer, "The Day [They Shot John Lennon]," playing at the [Dougherty Arts Center], entertains the imagination.

$$\Downarrow \Phi$$

$$\langle -3.51, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, \ldots 0, 1, 0 \rangle$$

Whether you're an aging flower child or a clueless [Gen-Xer], "The Day [They Shot John Lennon]," playing at the [Dougherty Arts Center], entertains the imagination.

$$\Downarrow \Phi$$

$$\langle -2.87, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, \ldots 0, 1, 0 \rangle$$

Experiments

Parsing Wall Street Journal Treebank

Training set = 40,000 sentences, test = 2,416 sentences
State-of-the-art parser: 88.2% F-measure
Reranked model: 89.5% F-measure (11% relative error reduction)
Boosting: 89.7% F-measure (13% relative error reduction)

Recovering Named-Entities in Web Data

Training data = 53,609 sentences (1,047,491 words),
test data = 14,717 sentences (291,898 words)
State-of-the-art tagger: 85.3% F-measure
Reranked model: 87.9% F-measure (17.7% relative error reduction)
Boosting: 87.6% F-measure (15.6% relative error reduction)

Perceptron Algorithm 3: Kernel Methods
(Work with Nigel Duffy)

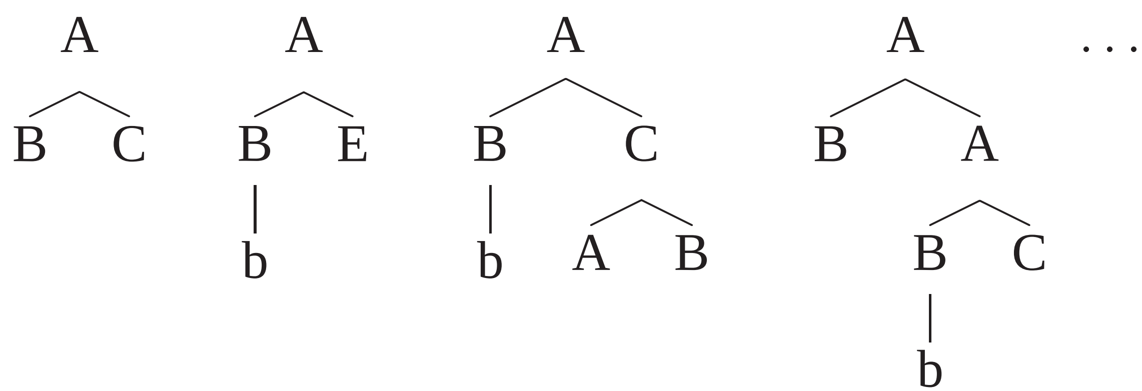
- It's simple to derive a "dual form" of the perceptron algorithm

If we can compute $\Phi(x) \cdot \Phi(y)$ efficiently
we can learn efficiently with the representation Φ

“All Subtrees” Representation [Bod 98]

- Given: Non-Terminal symbols $\{A, B, \ldots\}$
  Terminal symbols $\{a, b, c \ldots\}$

- An infinite set of subtrees



- **Step 1:**
  Choose an (arbitrary) mapping from subtrees to integers

$$h_i(x) = \text{Number of times subtree } i \text{ is seen in } x$$

$$\mathbf{\Phi}(x) = \langle h_1(x), h_2(x), h_3(x) \ldots \rangle$$

All Subtrees Representation

- $\mathbf{\Phi}$ is now huge
- **But** inner product $\mathbf{\Phi}(T_1) \cdot \mathbf{\Phi}(T_2)$ can be computed efficiently using dynamic programming.
  See [Collins and Duffy 2001, Collins and Duffy 2002]

Similar Kernels Exist for Tagged Sequences

Whether you're an aging flower child or a clueless [Gen-Xer], "[The Day They Shot John Lennon]," playing at the [Dougherty Arts Center], entertains the imagination.

$$\Downarrow \Phi$$

Whether | **[Gen-Xer],** | **Day They** | **John Lennon]," playing**

Whether you're an aging flower child or a clueless [Gen

. . .

Experiments

Parsing Wall Street Journal Treebank

Training set = 40,000 sentences, test = 2,416 sentences
State-of-the-art parser: 88.5% F-measure
Reranked model: 89.1% F-measure
(5% relative error reduction)

Recovering Named-Entities in Web Data

Training data = 53,609 sentences (1,047,491 words),
test data = 14,717 sentences (291,898 words)
State-of-the-art tagger: 85.3% F-measure
Reranked model: 87.6% F-measure
(15.6% relative error reduction)

Conclusions

Some Other Topics in Statistical NLP:

- Machine translation
- Unsupervised/partially supervised methods
- Finite state machines
- Generation
- Question answering
- Coreference
- Language modeling for speech recognition
- Lexical semantics
- Word sense disambiguation
- Summarization

MACHINE TRANSLATION (BROWN ET. AL)

- Training corpus: Canadian parliament (French-English translations)
- Task: learn mapping from French Sentence $\rightarrow$ English Sentence
- Noisy channel model:

$$translation(F) = \arg\max_E P(E|F) = \arg\max_E P(E)P(F|E)$$

- Parameterization

$$P(F|E) = \sum_A P(A|E)P(F|A, E)$$

- $\sum_A$ is a sum over possible alignments from English to French
  Model estimation through EM

References

[Bod 98] Bod, R. (1998). *Beyond Grammar: An Experience-Based Theory of Language*. CSLI Publications/Cambridge University Press.

[Booth and Thompson 73] Booth, T., and Thompson, R. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22(5), pages 442–450.

[Borthwick et. al 98] Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. (1998). Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. *Proc. of the Sixth Workshop on Very Large Corpora*.

[Collins and Duffy 2001] Collins, M. and Duffy, N. (2001). Convolution Kernels for Natural Language. In *Proceedings of NIPS 14*.

[Collins and Duffy 2002] Collins, M. and Duffy, N. (2002). New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL 2002*.

[Collins 2002] Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with the Perceptron Algorithm. In *Proceedings of EMNLP 2002*.

[Freund and Schapire 99] Freund, Y. and Schapire, R. (1999). Large Margin Classification using the Perceptron Algorithm. In *Machine Learning*, 37(3):277–296.

[Helmbold and Warmuth 95] Helmbold, D., and Warmuth, M. On Weak Learning. *Journal of Computer and System Sciences*, 50(3):551-573, June 1995.

[Hopcroft and Ullman 1979] Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to automata theory, languages, and computation*. Reading, Mass.: Addison–Wesley.

[Johnson et. al 1999] Johnson, M., Geman, S., Canon, S., Chi, S., & Riezler, S. (1999). Estimators for stochastic ‘unification-based” grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. San Francisco: Morgan Kaufmann.

[Lafferty et al. 2001] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of ICML-01, pages 282-289, 2001.

[MSM93] Marcus, M., Santorini, B., & Marcinkiewicz, M. (1993). Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics*, *19*, 313-330.

[McCallum et al. 2000] McCallum, A., Freitag, D., and Pereira, F. (2000) Maximum entropy markov models for information extraction and segmentation. In *Proceedings of ICML 2000*.

[Miller et. al 2000] Miller, S., Fox, H., Ramshaw, L., and Weischedel, R. 2000. A Novel Use of Statistical Parsing to Extract Information from Text. In *Proceedings of ANLP 2000*.

[Ramshaw and Marcus 95] Ramshaw, L., and Marcus, M. P. (1995). Text Chunking Using Transformation-Based Learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, Association for Computational Linguistics, 1995.

[Ratnaparkhi 96] A maximum entropy part-of-speech tagger. In *Proceedings of the empirical methods in natural language processing conference*.