6.864: Lecture 2, Fall 2007
Parsing and Syntax I

Overview

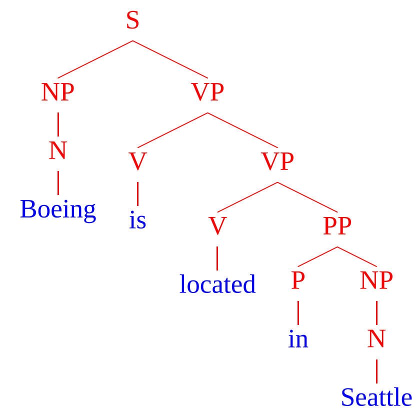
- An introduction to the parsing problem
- Context free grammars
- A brief(!) sketch of the syntax of English
- Examples of ambiguous structures
- PCFGs, their formal properties, and useful algorithms
- Weaknesses of PCFGs

Parsing (Syntactic Structure)

INPUT:

Boeing is located in Seattle.

OUTPUT:



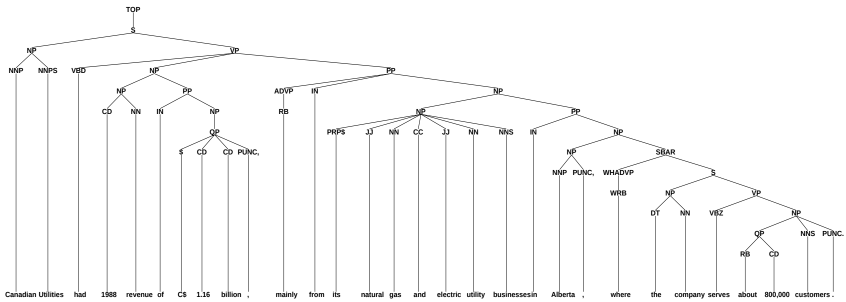
Syntactic Formalisms

- Work in formal syntax goes back to Chomsky's PhD thesis in the 1950s
- Examples of current formalisms: minimalism, lexical functional grammar (LFG), head-driven phrase-structure grammar (HPSG), tree adjoining grammars (TAG), categorial grammars

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:

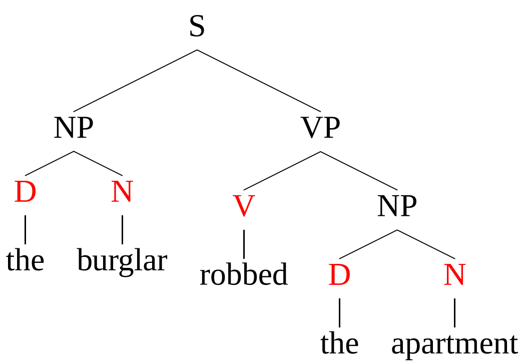


Canadian Utilities had 1988 revenue of C$ 1.16 billion , mainly from its natural gas and electric utility businesses in Alberta , where the company serves about 800,000 customers .

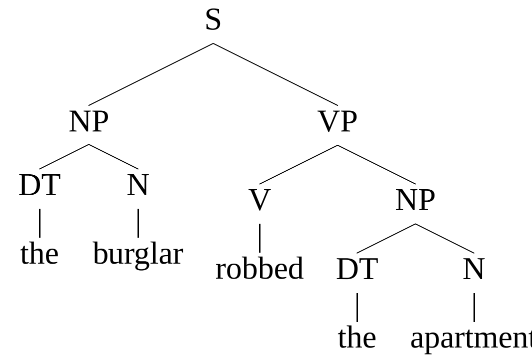
The Information Conveyed by Parse Trees

1) Part of speech for each word

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



2) Phrases

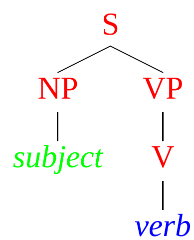


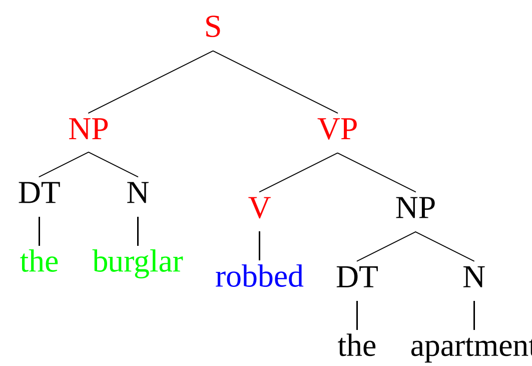
Noun Phrases (NP): “the burglar”, “the apartment”

Verb Phrases (VP): “robbed the apartment”

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

3) Useful Relationships





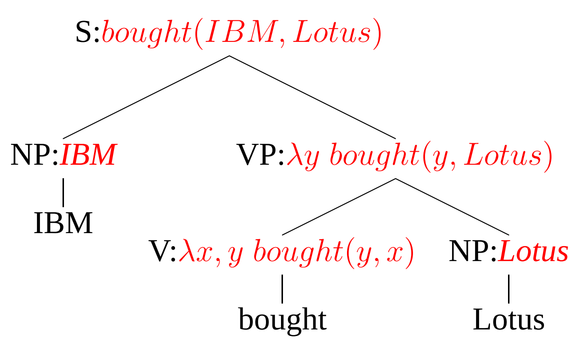
⇒ “the burglar” is the subject of “robbed”

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*

Syntax and Compositional Semantics



- Each syntactic non-terminal now has an associated semantic expression

Overview

- An introduction to the parsing problem
- Context free grammars
- A brief(!) sketch of the syntax of English
- Examples of ambiguous structures
- PCFGs, their formal properties, and useful algorithms
- Weaknesses of PCFGs

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, DT, Vi, Vt, NN, IN}
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	$\Rightarrow$	DT	NN
NP	$\Rightarrow$	NP	PP
PP	$\Rightarrow$	IN	NP

Vi	$\Rightarrow$	sleeps
Vt	$\Rightarrow$	saw
NN	$\Rightarrow$	man
NN	$\Rightarrow$	woman
NN	$\Rightarrow$	telescope
DT	$\Rightarrow$	the
IN	$\Rightarrow$	with
IN	$\Rightarrow$	in

Note: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

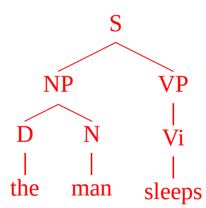
Left-Most Derivations

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

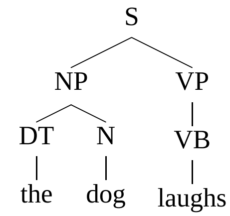
- $s_1 = S$, the start symbol
- $s_n \in \Sigma^*$, i.e. s_n is made up of terminal symbols only
- Each s_i for $i = 2 \ldots n$ is derived from s_{i-1} by picking the left-most non-terminal X in s_{i-1} and replacing it by some β where $X \rightarrow \beta$ is a rule in R

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:



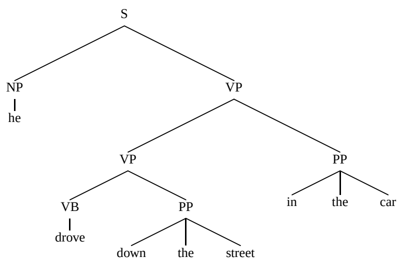
DERIVATION	RULES USED
S	S $\rightarrow$ NP VP
NP VP	NP $\rightarrow$ DT N
DT N VP	DT $\rightarrow$ the
the N VP	N $\rightarrow$ dog
the dog VP	VP $\rightarrow$ VB
the dog VB	VB $\rightarrow$ laughs
the dog laughs	



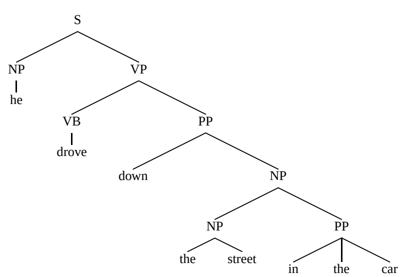
Properties of CFGs

- A CFG defines a set of possible derivations
- A string $s \in \Sigma^*$ is in the *language* defined by the CFG if there is at least one derivation which yields s
- Each string in the language generated by the CFG may have more than one derivation ("ambiguity")

DERIVATION	RULES USED
S	S → NP VP
NP VP	NP → he
he VP	VP → VP PP
he VP PP	VP → VB PP
he VB PP PP	VB→ drove
he drove PP PP	PP→ down the street
he drove down the street PP	PP→ in the car
he drove down the street in the car	



DERIVATION	RULES USED
S	S → NP VP
NP VP	NP → he
he VP	VP → VB PP
he VB PP	VB → drove
he drove PP	PP → down NP
he drove down NP	NP → NP PP
he drove down NP PP	NP → the street
he drove down the street PP	PP → in the car
he drove down the street in the car	

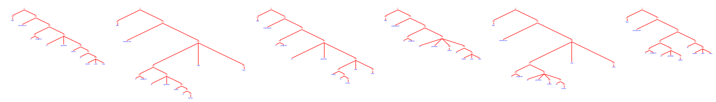


The Problem with Parsing: Ambiguity

INPUT:
She announced a program to promote safety in trucks and vans

⇓

POSSIBLE OUTPUTS:



And there are more...

Overview

- An introduction to the parsing problem
- Context free grammars
- A brief(!) sketch of the syntax of English
- Examples of ambiguous structures
- PCFGs, their formal properties, and useful algorithms
- Weaknesses of PCFGs

SEARCH
Books
GO!

WEB SEARCH
GO! Powered by Google

BOOK INFORMATION
buying info
customer reviews

RATE THIS ITEM
I dislike it I love it!
1 2 3 4 5
Submit
Edit your ratings

So You'd Like to...

READY TO BUY?
Add to Shopping Cart
or
Sign in to turn on 1-Click ordering.

MORE BUYING CHOICES
9 used & new from $131.99
Have one to sell? Sell yours here

Add to Wish List
Add to Wedding Registry
Don't have one? We'll set one up for you.

A Comprehensive Grammar of the English Language
by Randolph Quirk, Jan Svartvik (Contributor), Geoffry Leech (Contributor), Sidney Greenbaum

see larger photo
List Price: $272.20
Price: $272.20
Availability: Usually ships within 24 hours
Only 3 left in stock--order soon (more on the way).

9 used & new from $131.99

Edition: Hardcover

See more product details

- Special Shipping Information: This item is not eligible for FREE Super Saver Shipping. This product will incur a shipping surcharge of $1.40 in addition to the standard shipping fees.

Better Together

Buy this book with Hardcover, Longman Grammar of Spoken and Written English by Douglas Biber (Author), et al today!

Learn about the English langua...: A guide by Benjamin Lukoff, B.A. (Washington), English language & literature;...

Create your guide

Buy Together Today: $372.15
Buy both now!

Customers who bought this book also bought:

- The Cambridge Grammar of the English Language by Rodney D. Huddleston (Author), Geoffrey K. Pullum (Author) (Hardcover)
- The Oxford Dictionary of English Grammar by Sylvia Chalker (Editor), Edmund Weiner (Editor) (Hardcover)
- The Oxford English Grammar by Sidney Greenbaum (Hardcover)
- A Student's Grammar of the English Language by Sidney Greenbaum, Randolph Quirk (Contributor) (Paperback)
- The Grammar Book: An Esl/Efl Teacher's Course by Marianne Celce-Murcia, Diane Larsen-Freeman (Hardcover)

Explore Similar Items: 20 in Books

Product Details

- Hardcover: 1779 pages ; Dimensions (in inches): 3.29 x 11.92 x 8.25
- Publisher: Addison-Wesley Pub Co; (February 1989)
- ISBN: 0582517346
- Average Customer Review: Based on 10 reviews. Write a review.
- Amazon.com Sales Rank: 114,478

Shipping: Due to this item's unusual size or weight, it requires special handling and will ship separately from other items in your order. Read More.

What's Your Advice?
Is there an item you'd recommend instead of or in addition to this one? Let the world know! Enter the item's ASIN (what's an ASIN?) in the box below, select advice type, then click Submit.

I recommend: in addition to this product / instead of this product Submit

Spotlight Reviews (What's this?)
Write an online review and share your thoughts with other customers.

16 of 16 people found the following review helpful:

Still Useful, but..., March 11, 2003
Reviewer: metrist (see more about me) from Glendale, Ca. USA
As the title and price suggest, this is a reference grammar of English, not a textbook. It's written for people who already have a grasp of basic grammatical principles. This is the sort of book that you pick up when you want to look up patterns of verb complementation, etc. Only a masochist would try to read it straight through, or to learn grammar from it.

The Comprehensive Grammar is an expanded and revised version of a series of grammars

A Brief Overview of English Syntax

Parts of Speech (tags from the Brown corpus):

- Nouns

  NN = singular noun e.g., man, dog, park
  NNS = plural noun e.g., telescopes, houses, buildings
  NNP = proper noun e.g., Smith, Gates, IBM

- Determiners
  DT = determiner e.g., the, a, some, every

- Adjectives
  JJ = adjective e.g., red, green, large, idealistic

A Fragment of a Noun Phrase Grammar

$$
\begin{array}{lcll}
\bar{N} & \Rightarrow & NN & \\
\bar{N} & \Rightarrow & NN & \bar{N} \\
\bar{N} & \Rightarrow & JJ & \bar{N} \\
\bar{N} & \Rightarrow & \bar{N} & \bar{N} \\
NP & \Rightarrow & DT & \bar{N}
\end{array}
$$

$$
\begin{array}{lcl}
NN & \Rightarrow & \text{box} \\
NN & \Rightarrow & \text{car} \\
NN & \Rightarrow & \text{mechanic} \\
NN & \Rightarrow & \text{pigeon} \\
 & & \\
DT & \Rightarrow & \text{the} \\
DT & \Rightarrow & \text{a} \\
 & & \\
JJ & \Rightarrow & \text{fast} \\
JJ & \Rightarrow & \text{metal} \\
JJ & \Rightarrow & \text{idealistic} \\
JJ & \Rightarrow & \text{clay}
\end{array}
$$

Generates:
a box, the box, the metal box, the fast car mechanic, . . .

Prepositions, and Prepositional Phrases

- Prepositions
  IN = preposition   e.g., of, in, out, beside, as

An Extended Grammar

$\bar{N}$	$\Rightarrow$	NN	
$\bar{N}$	$\Rightarrow$	NN	$\bar{N}$
$\bar{N}$	$\Rightarrow$	JJ	$\bar{N}$
$\bar{N}$	$\Rightarrow$	$\bar{N}$	$\bar{N}$
NP	$\Rightarrow$	DT	$\bar{N}$
PP	$\Rightarrow$	IN	NP
$\bar{N}$	$\Rightarrow$	$\bar{N}$	PP

NN	$\Rightarrow$	box
NN	$\Rightarrow$	car
NN	$\Rightarrow$	mechanic
NN	$\Rightarrow$	pigeon
DT	$\Rightarrow$	the
DT	$\Rightarrow$	a

JJ	$\Rightarrow$	fast
JJ	$\Rightarrow$	metal
JJ	$\Rightarrow$	idealistic
JJ	$\Rightarrow$	clay
IN	$\Rightarrow$	in
IN	$\Rightarrow$	under
IN	$\Rightarrow$	of
IN	$\Rightarrow$	on
IN	$\Rightarrow$	with
IN	$\Rightarrow$	as

Generates:
in a box, under the box, the fast car mechanic under the pigeon in the box, . . .

Verbs, Verb Phrases, and Sentences

- Basic Verb Types
  Vi = Intransitive verb   e.g., sleeps, walks, laughs
  Vt = Transitive verb   e.g., sees, saw, likes
  Vd = Ditransitive verb   e.g., gave

- Basic VP Rules
  VP $\rightarrow$ Vi
  VP $\rightarrow$ Vt NP
  VP $\rightarrow$ Vd NP NP

- Basic S Rule
  S $\rightarrow$ NP VP

Examples of VP:
sleeps, walks, likes the mechanic, gave the mechanic the fast car, gave the fast car mechanic the pigeon in the box, . . .

Examples of S:
the man sleeps, the dog walks, the dog likes the mechanic, the dog in the box gave the mechanic the fast car,. . .

PPs Modifying Verb Phrases

A new rule:
VP $\rightarrow$ VP PP

New examples of VP:
sleeps in the car, walks like the mechanic, gave the mechanic the fast car on Tuesday, . . .

Complementizers, and SBARs

- Complementizers
  COMP = complementizer e.g., that

- SBAR
  SBAR $\rightarrow$ COMP S

Examples:
that the man sleeps, that the mechanic saw the dog . . .

More Verbs

- New Verb Types
  V[5] e.g., said, reported
  V[6] e.g., told, informed
  V[7] e.g., bet

- New VP Rules

VP	$\rightarrow$	V[5]	SBAR		
VP	$\rightarrow$	V[6]	NP	SBAR	
VP	$\rightarrow$	V[7]	NP	NP	SBAR

Examples of New VPs:
said that the man sleeps
told the dog that the mechanic likes the pigeon
bet the pigeon $50 that the mechanic owns a fast car

Coordination

- A New Part-of-Speech:
  CC = Coordinator e.g., and, or, but

- New Rules

NP	$\rightarrow$	NP	CC	NP
$\bar{N}$	$\rightarrow$	$\bar{N}$	CC	$\bar{N}$
VP	$\rightarrow$	VP	CC	VP
S	$\rightarrow$	S	CC	S
SBAR	$\rightarrow$	SBAR	CC	SBAR

Overview

- An introduction to the parsing problem
- Context free grammars
- A brief(!) sketch of the syntax of English
- Examples of ambiguous structures
- PCFGs, their formal properties, and useful algorithms
- Weaknesses of PCFGs

Sources of Ambiguity

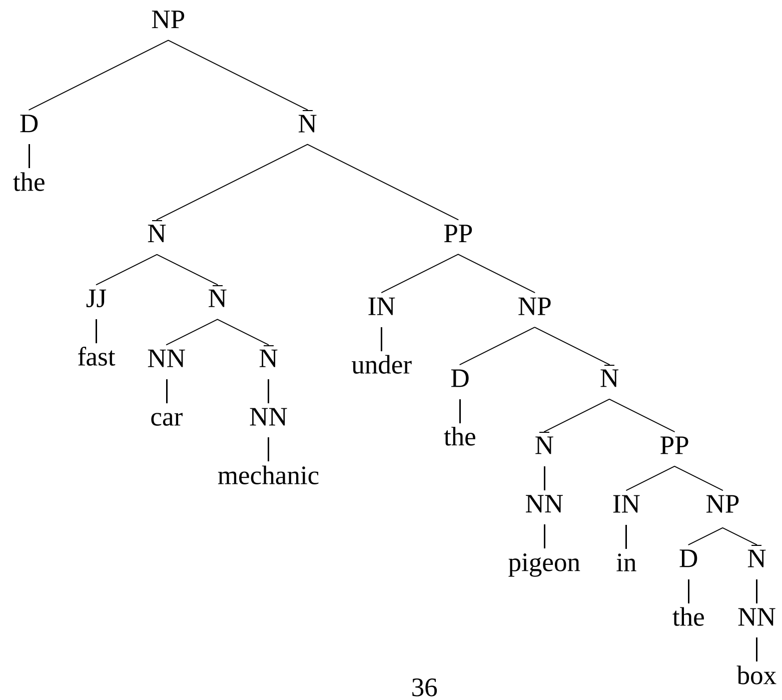
- Part-of-Speech ambiguity

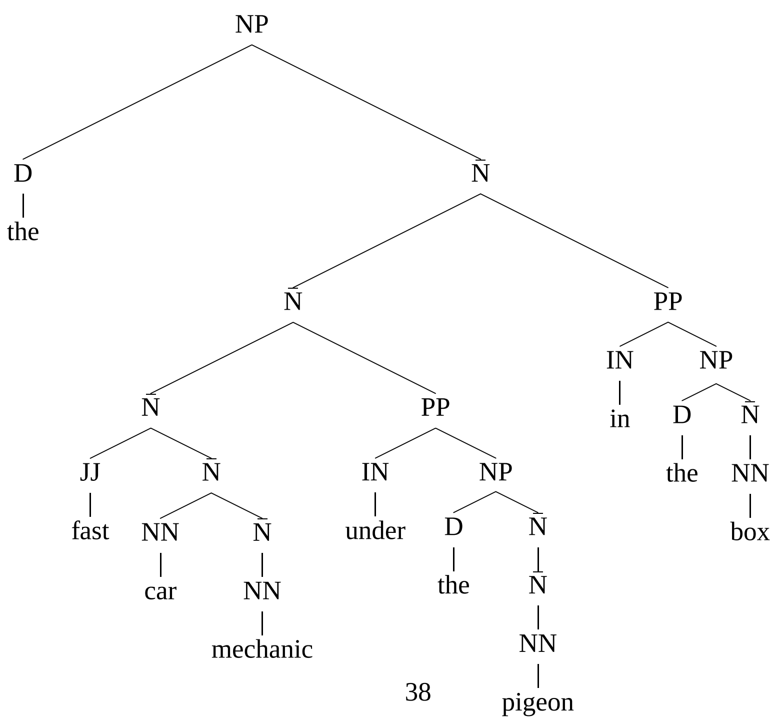
  NNS $\rightarrow$ walks

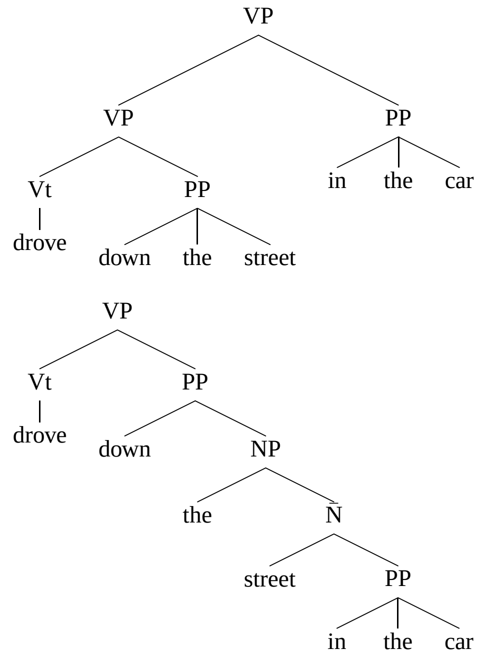
  Vi $\rightarrow$ walks

- Prepositional Phrase Attachment

  the fast car mechanic under the pigeon in the box



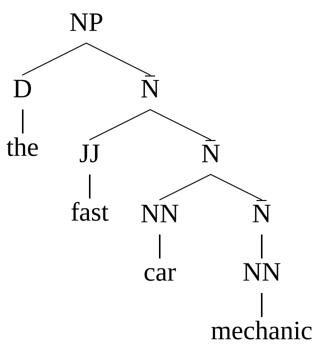


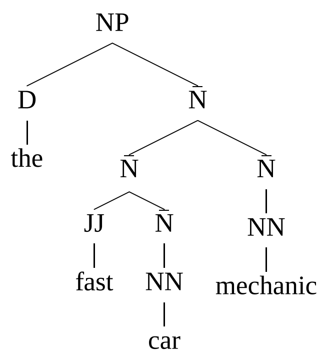


Two analyses for: John was believed to have been shot by Bill

Sources of Ambiguity: Noun Premodifiers

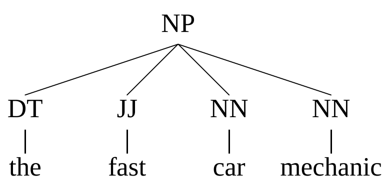
- Noun premodifiers:

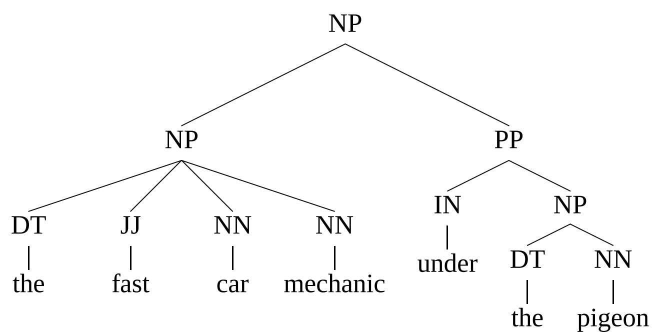




A Funny Thing about the Penn Treebank

Leaves NP premodifier structure flat, or underspecified:





Overview

- An introduction to the parsing problem
- Context free grammars
- A brief(!) sketch of the syntax of English
- Examples of ambiguous structures
- PCFGs, their formal properties, and useful algorithms
- Weaknesses of PCFGs

A Probabilistic Context-Free Grammar (PCFG)

S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	P	NP	1.0

Vi	⇒	sleeps	1.0
Vt	⇒	saw	1.0
NN	⇒	man	0.7
NN	⇒	woman	0.2
NN	⇒	telescope	0.1
DT	⇒	the	1.0
IN	⇒	with	0.5
IN	⇒	in	0.5

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

DERIVATION	RULES USED	PROBABILITY
S	S $\rightarrow$ NP VP	1.0
NP VP	NP $\rightarrow$ DT N	0.3
DT N VP	DT $\rightarrow$ the	1.0
the N VP	N $\rightarrow$ dog	0.1
the dog VP	VP $\rightarrow$ VB	0.4
the dog VB	VB $\rightarrow$ laughs	0.5
the dog laughs		

TOTAL PROBABILITY = $1.0 \times 0.3 \times 1.0 \times 0.1 \times 0.4 \times 0.5$

Properties of PCFGs

- Assigns a probability to each *left-most derivation*, or parse-tree, allowed by the underlying CFG

- Say we have a sentence S, set of derivations for that sentence is $\mathcal{T}(S)$. Then a PCFG assigns a probability to each member of $\mathcal{T}(S)$. i.e., *we now have a ranking in order of probability*.

- The probability of a string S is

$$\sum_{T \in \mathcal{T}(S)} P(T, S)$$

Deriving a PCFG from a Corpus

- Given a set of example trees, the underlying CFG can simply be **all rules seen in the corpus**

- Maximum Likelihood estimates:

$$P_{ML}(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

  where the counts are taken from a training set of example trees.

- **If the training data is generated by a PCFG**, then as the training data size goes to infinity, the maximum-likelihood PCFG will converge to the same distribution as the "true" PCFG.

PCFGs

[Booth and Thompson 73] showed that a CFG with rule probabilities correctly defines a distribution over the set of derivations 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.)

Algorithms for PCFGs

- Given a PCFG and a sentence S, define $\mathcal{T}(S)$ to be the set of trees with S as the yield.

- Given a PCFG and a sentence S, how do we find

$$\arg \max_{T \in \mathcal{T}(S)} P(T, S)$$

- Given a PCFG and a sentence S, how do we find

$$P(S) = \sum_{T \in \mathcal{T}(S)} P(T, S)$$

Chomsky Normal Form

A context free grammar $G = (N, \Sigma, R, S)$ in Chomsky Normal Form is as follows

- N is a set of non-terminal symbols
- Σ is a set of terminal symbols
- R is a set of rules which take one of two forms:
  - $X \rightarrow Y_1 Y_2$ for $X \in N$, and $Y_1, Y_2 \in N$
  - $X \rightarrow Y$ for $X \in N$, and $Y \in \Sigma$
- $S \in N$ is a distinguished start symbol

A Dynamic Programming Algorithm

- Given a PCFG and a sentence S, how do we find

$$\max_{T \in \mathcal{T}(S)} P(T, S)$$

- Notation:

n = number of words in the sentence
N_k for $k = 1 \ldots K$ is k'th non-terminal
$N_1 = S$ (the start symbol)

- Define a dynamic programming table

$\pi[i, j, k]$ = maximum probability of a constituent with non-terminal N_k spanning words $i \ldots j$ inclusive

- Our goal is to calculate $\max_{T \in \mathcal{T}(S)} P(T, S) = \pi[1, n, 1]$

A Dynamic Programming Algorithm

- Base case definition: for all $i = 1 \ldots n$, for $k = 1 \ldots K$

$$\pi[i, i, k] = P(N_k \rightarrow w_i \mid N_k)$$

(note: define $P(N_k \rightarrow w_i \mid N_k) = 0$ if $N_k \rightarrow w_i$ is not in the grammar)

- Recursive definition: for all $i = 1 \ldots n$, $j = (i + 1) \ldots n$, $k = 1 \ldots K$,

$$\pi[i, j, k] = \max_{\substack{i \leq s < j \\ 1 \leq l \leq K \\ 1 \leq m \leq K}} \{P(N_k \rightarrow N_l N_m \mid N_k) \times \pi[i, s, l] \times \pi[s + 1, j, m]\}$$

(note: define $P(N_k \rightarrow N_l N_m \mid N_k) = 0$ if $N_k \rightarrow N_l N_m$ is not in the grammar)

Initialization:

For i = 1 ... n, k = 1 ... K

$\pi[i,i,k] = P(N_k \rightarrow w_i | N_k)$

Main Loop:

For $length = 1 \ldots (n-1)$, $i = 1 \ldots (n - 1ength)$, $k = 1 \ldots K$

$j \leftarrow i + length$

$max \leftarrow 0$

For $s = i \ldots (j-1)$,

For N_l, N_m such that $N_k \rightarrow N_l N_m$ is in the grammar

$prob \leftarrow P(N_k \rightarrow N_l N_m) \times \pi[i,s,l] \times \pi[s+1,j,m]$

If $prob > max$

$max \leftarrow prob$

//Store backpointers which imply the best parse

$Split(i,j,k) = \{s,l,m\}$

$\pi[i,j,k] = max$

A Dynamic Programming Algorithm for the Sum

- Given a PCFG and a sentence S, how do we find

$$\sum_{T \in \mathcal{T}(S)} P(T,S)$$

- Notation:

n = number of words in the sentence

N_k for $k = 1 \ldots K$ is k'th non-terminal

$N_1 = S$ (the start symbol)

- Define a dynamic programming table

$\pi[i,j,k]$ = sum of probability of parses with root label N_k spanning words $i \ldots j$ inclusive

- Our goal is to calculate $\sum_{T \in \mathcal{T}(S)} P(T,S) = \pi[1,n,1]$

A Dynamic Programming Algorithm for the Sum

- Base case definition: for all $i = 1 \ldots n$, for $k = 1 \ldots K$

$$\pi[i,i,k] = P(N_k \rightarrow w_i \mid N_k)$$

(note: define $P(N_k \rightarrow w_i \mid N_k) = 0$ if $N_k \rightarrow w_i$ is not in the grammar)

- Recursive definition: for all $i = 1 \ldots n$, $j = (i+1) \ldots n$, $k = 1 \ldots K$,

$$\pi[i,j,k] = \sum_{\substack{i \le s < j \\ 1 \le l \le K \\ 1 \le m \le K}} \{P(N_k \rightarrow N_l N_m \mid N_k) \times \pi[i,s,l] \times \pi[s+1,j,m]\}$$

(note: define $P(N_k \rightarrow N_l N_m \mid N_k) = 0$ if $N_k \rightarrow N_l N_m$ is not in the grammar)

Initialization:

For i = 1 ... n, k = 1 ... K

$\pi[i,i,k] = P(N_k \rightarrow w_i | N_k)$

Main Loop:

For $length = 1 \ldots (n-1)$, $i = 1 \ldots (n - 1ength)$, $k = 1 \ldots K$

$j \leftarrow i + length$

$sum \leftarrow 0$

For $s = i \ldots (j-1)$,

For N_l, N_m such that $N_k \rightarrow N_l N_m$ is in the grammar

$prob \leftarrow P(N_k \rightarrow N_l N_m) \times \pi[i,s,l] \times \pi[s+1,j,m]$

$sum \leftarrow sum + prob$

$\pi[i,j,k] = sum$

Overview

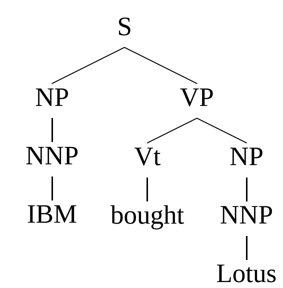
- An introduction to the parsing problem
- Context free grammars
- A brief(!) sketch of the syntax of English
- Examples of ambiguous structures
- PCFGs, their formal properties, and useful algorithms
- Weaknesses of PCFGs

Overview

- An introduction to the parsing problem
- Context free grammars
- A brief(!) sketch of the syntax of English
- Examples of ambiguous structures
- PCFGs, their formal properties, and useful algorithms
- Weaknesses of PCFGs

Weaknesses of PCFGs

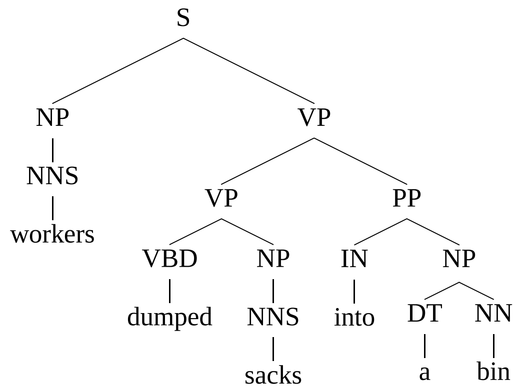
- Lack of sensitivity to lexical information
- Lack of sensitivity to structural frequencies



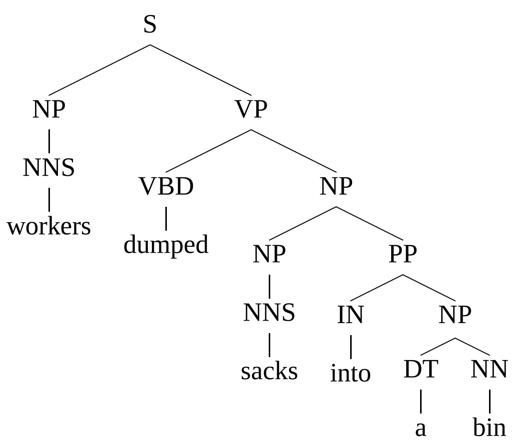
$$\begin{aligned} \text{PROB} = \;& P(\text{S} \rightarrow \text{NP VP} \mid \text{S}) && \times P(\text{NNP} \rightarrow \textit{IBM} \mid \text{NNP}) \\ & \times P(\text{VP} \rightarrow \text{V NP} \mid \text{VP}) && \times P(\text{Vt} \rightarrow \textit{bought} \mid \text{Vt}) \\ & \times P(\text{NP} \rightarrow \text{NNP} \mid \text{NP}) && \times P(\text{NNP} \rightarrow \textit{Lotus} \mid \text{NNP}) \\ & \times P(\text{NP} \rightarrow \text{NNP} \mid \text{NP}) && \end{aligned}$$

Another Case of PP Attachment Ambiguity

(a)



(b)



(a)

Rules
S $\rightarrow$ NP VP
NP $\rightarrow$ NNS
VP $\rightarrow$ VP PP
VP $\rightarrow$ VBD NP
NP $\rightarrow$ NNS
PP $\rightarrow$ IN NP
NP $\rightarrow$ DT NN
NNS $\rightarrow$ workers
VBD $\rightarrow$ dumped
NNS $\rightarrow$ sacks
IN $\rightarrow$ into
DT $\rightarrow$ a
NN $\rightarrow$ bin

(b)

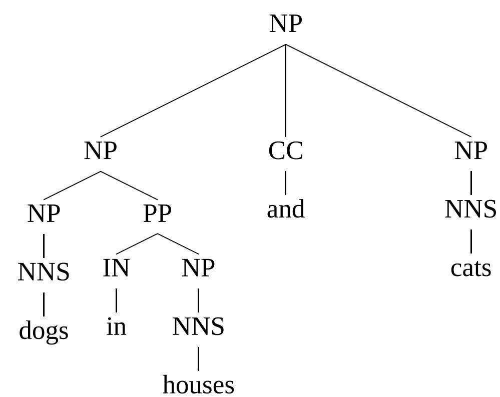
Rules
S $\rightarrow$ NP VP
NP $\rightarrow$ NNS
NP $\rightarrow$ NP PP
VP $\rightarrow$ VBD NP
NP $\rightarrow$ NNS
PP $\rightarrow$ IN NP
NP $\rightarrow$ DT NN
NNS $\rightarrow$ workers
VBD $\rightarrow$ dumped
NNS $\rightarrow$ sacks
IN $\rightarrow$ into
DT $\rightarrow$ a
NN $\rightarrow$ bin

If $P(\text{NP} \rightarrow \text{NP PP} \mid \text{NP}) > P(\text{VP} \rightarrow \text{VP PP} \mid \text{VP})$ then (b) is more probable, else (a) is more probable.

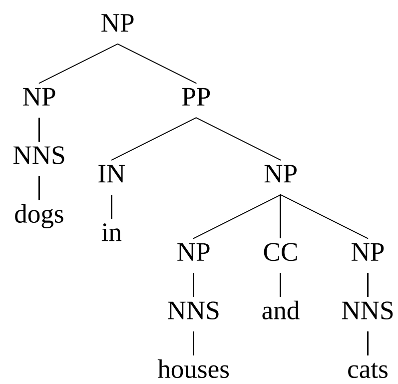
Attachment decision is completely independent of the words

A Case of Coordination Ambiguity

(a)



(b)



(a)

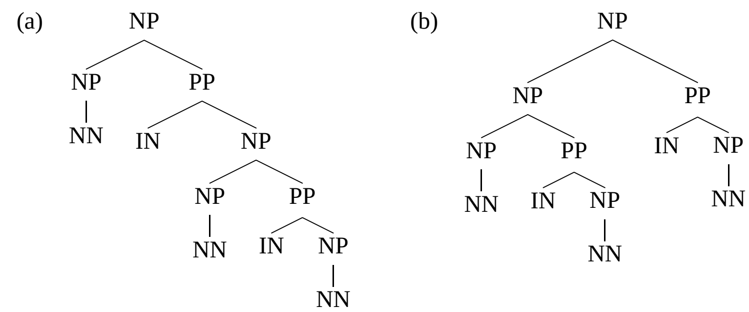
Rules
NP → NP CC NP
NP → NP PP
NP → NNS
PP → IN NP
NP → NNS
NP → NNS
NNS → dogs
IN → in
NNS → houses
CC → and
NNS → cats

(b)

Rules
NP → NP CC NP
NP → NP PP
NP → NNS
PP → IN NP
NP → NNS
NP → NNS
NNS → dogs
IN → in
NNS → houses
CC → and
NNS → cats

Here the two parses have identical rules, and therefore have identical probability under any assignment of PCFG rule probabilities

Structural Preferences: Close Attachment



- Example: president of a company in Africa
- Both parses have the same rules, therefore receive same probability under a PCFG
- “Close attachment” (structure (a)) is twice as likely in Wall Street Journal text.

Structural Preferences: Close Attachment

Previous example: John was believed to have been shot by Bill

Here the low attachment analysis (Bill does the *shooting*) contains same rules as the high attachment analysis (Bill does the *believing*), so the two analyses receive same probability.

References

[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.

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