Statistical Natural Language Processing

Lecture 9: Parsing

Dr. Momtazi
Amirkabir University of Technology

Parsing

- Finding structural relationship between words in a sentence

- Applications
  - Spell checking
  - Speech recognition
  - Machine translation
  - Language modeling

Outline

Outline

1. Constituency vs. Dependency
2. Context Free Grammar
   Syntactic Parsing
3. Probabilistic Context Free Grammar
   Statistical Parsing
4. Lexicalized PCFG
5. Dependency Parsing
6. Evaluation

Constituency

- Working based on Constituency (Phrase structure)
  - Organizing words into nested constituents
  - Showing that groups of words within utterances can act as single units
  - Forming coherent classes from these units that can behave in similar ways
    - With respect to their internal structure
    - With respect to other units in the language
  - Considering a head word for each constituent

Constituency

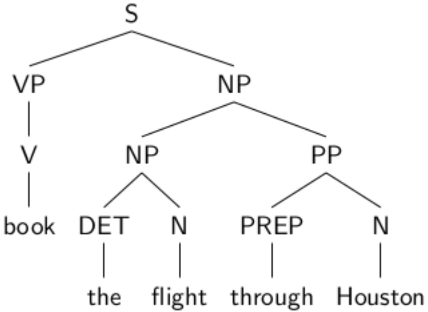
the writer talked to the audiences about his new book.

the writer talked about his new book to the audiences. ✔

about his new book the writer talked to the audiences. ✔

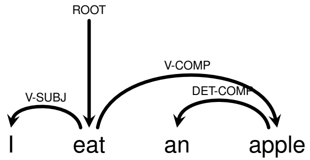
the writer talked book to the audiences about his new. ✗

Constituency



Dependency

- Identifying which words depend on (modify or arguments of) which other words



Outline

Context Free Grammar (CFG)

- Grammar G consists of
  - Terminals (T)
  - Non-terminals (N)
  - Start symbol (S)
  - Rules (R)

CFG

- Terminals
  - The set of words in the text
- Non-Terminals
  - The constituents in a language
    (noun phrase, verb phrase, ...)
- Start symbol
  - The main constituent of the language
    (sentence)
- Rules
  - Equations that consist of a single non-terminal on the left and any number of terminals and non-terminals on the right

CFG

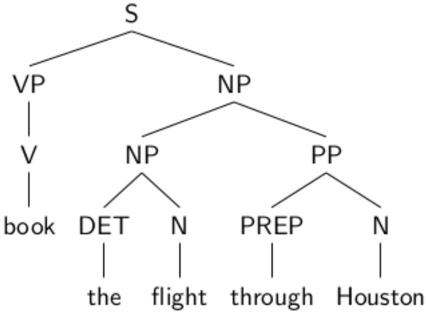
$S \rightarrow NP \;\; VP$
$S \rightarrow VP$
$NP \rightarrow N$
$NP \rightarrow Det \; N$
$NP \rightarrow NP \;\; NP$
$NP \rightarrow NP \;\; PP$
$VP \rightarrow V$
$VP \rightarrow VP \;\; PP$
$VP \rightarrow VP \;\; NP$
$PP \rightarrow Prep \;\; NP$

$N \rightarrow$ book
$V \rightarrow$ book
$Det \rightarrow$ the
$N \rightarrow$ flight
$Prep \rightarrow$ through
$N \rightarrow$ Houston

CFG

V	DET	N	PREP	N
Book	*the*	*flight*	*through*	*Houston*

CFG



Outline

1. Constituency vs. Dependency
2. Context Free Grammar
   - Syntactic Parsing
3. Probabilistic Context Free Grammar
4. Lexicalized PCFG
5. Dependency Parsing
6. Evaluation

Parsing

- Taking a string and a grammar and returning proper parse tree(s) for that string
- Covering all and only the elements of the input string
- Reaching the start symbol at the top of the string

- The system cannot select the correct tree among all the possible trees

Main Grammar Fragments

- Sentence
- Noun Phrase
  - Agreement
- Verb Phrase
  - Sub-categorization

Grammar Fragments: Sentence

- Declaratives
  A plane left.
  $S \rightarrow NP \;\; VP$

- Imperatives
  Leave!
  $S \rightarrow VP$

- Yes-No Questions
  Did the plane leave?
  $S \rightarrow Aux \;\; NP \;\; VP$

- WH Questions
  When did the plane leave?
  $S \rightarrow NP_{WH} \;\; Aux \;\; NP \;\; VP$

Grammar Fragments: NP

- Each NP has a central critical noun called head
- The head of an NP can be expressed using
  - Pre-nominals: the words that can come before the head
  - Post-nominals: the words that can come after the head

Grammar Fragments: NP

- Pre-nominals
  - Simple lexical items: *the, this, a, an, ...*
    a car
  - Simple possessives
    John's car
  - Complex recursive possessives
    John's sister's friend's car
  - Quantifiers, cardinals, ordinals...
    three cars
  - Adjectives
    large cars

Grammar Fragments: NP

- Post-nominals
  - Prepositional phrases
    flight from Seattle
  - Non-finite clauses
    flight arriving before noon
  - Relative clauses
    flight that serves breakfast

Agreement

- Having constraints that hold among various constituents
- Considering these constraints in a rule or set of rules

Example: determiners and the head nouns in NPs have to agree in number

This flight ✔
Those flights ✔
This flights ✗
Those flight ✗

- Grammars that do not consider constraints will over-generate
  - Accepting and assigning correct structures to grammatical examples (*this flight*)
  - But also accepting incorrect examples (*these flight*)

Agreement at sentence level

- Considering similar constraints at sentence level

Example: subject and verb in sentences have to agree in number and person

John flies ✔
We fly ✔
John fly ✗
We flies ✗

Agreement

- Possible CFG solution

$S_{sg} \rightarrow NP_{sg} \;\; VP_{sg}$

$S_{pl} \rightarrow NP_{pl} \;\; VP_{pl}$

$NP_{sg} \rightarrow Det_{sg} \;\; N_{sg}$

$NP_{pl} \rightarrow Det_{pl} \;\; N_{pl}$

$VP_{sg} \rightarrow V_{sg} \;\; NP_{sg}$

$VP_{pl} \rightarrow V_{pl} \;\; NP_{pl}$

...

- Shortcoming:
  - Introducing many rules in the system

Grammar Fragments: VP

- VPs consist of a head verb along with zero or more constituents called arguments

$VP \rightarrow V$	*disappear*
$VP \rightarrow V \quad NP$	*prefer a morning flight*
$VP \rightarrow V \quad PP$	*fly on Thursday*
$VP \rightarrow V \quad NP \quad PP$	*leave Boston in the morning*
$VP \rightarrow V \quad NP \quad NP$	*give me the flight number*

- Arguments
  - Obligatory: complement
  - Optional: adjunct

Sub-categorization

- Even though there are many valid VP rules, not all verbs are allowed to participate in all VP rules

  disappear a morning flight ✗

- Solution:
  - Subcategorizing the verbs according to the sets of VP rules that they can participate in
  - This is a modern take on the traditional notion of transitive/intransitive
  - Modern grammars may have 100s or such classes

Sub-categorization

- Example:

Sneeze	*John sneezed*
Find	*Please find [a flight to NY]$_{NP}$*
Give	*Give [me]$_{NP}$[a cheaper fair]$_{NP}$*
Help	*Can you help [me]$_{NP}$[with a flight]$_{PP}$*
Prefer	*I prefer [to leave earlier]$_{TO-VP}$*
Told	*I was told [United has a flight]$_{S}$*

John sneezed the book ✗
I prefer United has a flight ✗
Give with a flight ✗

Sub-categorization

- The over-generation problem also exists in VP rules
  - Permitting the presence of strings containing verbs and arguments that do not go together

John sneezed the book
$VP \rightarrow V \;\; NP$

- Solution:
  - Similar to agreement phenomena, we need a way to formally express the constraints

Parsing Algorithms

- Top-Down
  - Starting with the rules that give us an S, since trees should be rooted with an S
  - Working on the way down from S to the words

- Bottom-Up
  - Starting with trees that link up with the words, since trees should cover the input words
  - Working on the way up from words to larger and larger trees

Top-Down vs. Bottom-Up

- Top-Down
  - Only searches for trees that can be answers (i.e. S’s)
  - But also suggests trees that are not consistent with any of the words
- Bottom-Up
  - Only forms trees consistent with the words
  - But suggests trees that make no sense globally

Top-Down vs. Bottom-Up

- In both cases, we left out how to keep track of the search space and how to make choices

- Solutions
  - Backtracking
    - Making a choice, if it works out then fine
    - If not, then back up and make a different choice
      $\Rightarrow$ duplicated work

  - Dynamic programming
    - Avoiding repeated work
    - Solving exponential problems in polynomial time
    - Storing ambiguous structures efficiently

Dynamic Programming Methods

- CKY: bottom-up
- Early: top-down

Chomsky Normal Form

- Each grammar can be represented by a set of binary rules

$$A \rightarrow B\ C$$

$$A \rightarrow w$$

A, B, C are noun-terminals w is a terminal

Chomsky Normal Form

- Converting to Chomsky normal form

$$A \rightarrow B\ C\ D$$

$$X \rightarrow B\ C$$

$$A \rightarrow X\ D$$

X does not occur anywhere else in the the grammar

Chomsky Normal Form

- Converting to Chomsky normal form

$$A \rightarrow B$$
$$B \rightarrow C\ D$$

$$A \rightarrow C\ D$$

CKY Parsing

$$A \rightarrow B\ C$$

- If there is an A somewhere in the input, then there must be a B followed by a C in the input
- If the A spans from i to j in the input, then there must be a k such that $i < k < j$
  - B spans from i to k
  - C spans from k to j

CKY Parsing

[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	[1,2]	[1,3]	[1,4]	[1,5]
		[2,3]	[2,4]	[2,5]
			[3,4]	[3,5]
				[4,5]
Book	the	flight	through	Houston

0 Book 1 the 2 flight 3 through 4 Houston 5

CKY Parsing

N → book$_{[0,1]}$ **V → book**$_{[0,1]}$ [0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	[1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ [2,3]	[2,4]	[2,5]
			Prep → through$_{[3,4]}$ [3,4]	[3,5]
				N → houston$_{[4,5]}$ [4,5]

Book the flight through Houston

0 1 2 3 4 5

CKY Parsing

$N \rightarrow book_{[0,1]}$ $V \rightarrow book_{[0,1]}$ $NP \rightarrow N_{[0,1]}$ $VP \rightarrow V_{[0,1]}$ $S \rightarrow VP_{[0,1]}$ [0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	$Det \rightarrow the_{[1,2]}$ [1,2]	[1,3]	[1,4]	[1,5]
		$N \rightarrow flight_{[2,3]}$ $NP \rightarrow N_{[2,3]}$ [2,3]	[2,4]	[2,5]
			$Prep \rightarrow through_{[3,4]}$ [3,4]	[3,5]
				$N \rightarrow houston_{[4,5]}$ $NP \rightarrow N_{[4,5]}$ [4,5]

Book the flight through Houston

0 1 2 3 4 5

CKY Parsing

N → book$_{[0,1]}$ **V → book**$_{[0,1]}$ **NP → N**$_{[0,1]}$ **VP → V**$_{[0,1]}$ **S → VP**$_{[0,1]}$ [0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	**NP → Det**$_{[1,2]}$**, N**$_{[2,3]}$ [1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ **NP → N**$_{[2,3]}$ [2,3]	[2,4]	[2,5]
			Prep → through$_{[3,4]}$ [3,4]	[3,5]
				N → houston$_{[4,5]}$ **NP → N**$_{[4,5]}$ [4,5]
Book	the	flight	through	Houston

0 1 2 3 4 5

CKY Parsing

N → book$_{[0,1]}$ **V → book**$_{[0,1]}$ **NP → N**$_{[0,1]}$ **VP → V**$_{[0,1]}$ **S → VP**$_{[0,1]}$ [0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	**NP → Det**$_{[1,2]}$**, N**$_{[2,3]}$ [1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ **NP → N**$_{[2,3]}$ [2,3]	[2,4]	[2,5]
			Prep → through$_{[3,4]}$ [3,4]	**PP → Prep**$_{[3,4]}$**,NP**$_{[4,5]}$ [3,5]
				N → houston$_{[4,5]}$ **NP → N**$_{[4,5]}$ [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

CKY Parsing

N → book$_{[0,1]}$ V → book$_{[0,1]}$ NP → N$_{[0,1]}$ VP → V$_{[0,1]}$ S → VP$_{[0,1]}$ [0,1]	[0,2]	NP → NP$_{[0,1]}$, NP$_{[1,3]}$ VP → VP$_{[0,1]}$, NP$_{[1,3]}$ S → VP$_{[0,3]}$ [0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	NP → Det$_{[1,2]}$, N$_{[2,3]}$ [1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ NP → N$_{[2,3]}$ [2,3]	[2,4]	[2,5]
			Prep → through$_{[3,4]}$ [3,4]	PP → Prep$_{[3,4]}$,NP$_{[4,5]}$ [3,5]
				N → houston$_{[4,5]}$ NP → N$_{[4,5]}$ [4,5]

Book	the	flight	through	Houston

0 1 2 3 4 5

CKY Parsing

N → book$_{[0,1]}$ **V → book**$_{[0,1]}$ **NP → N**$_{[0,1]}$ **VP → V**$_{[0,1]}$ **S → VP**$_{[0,1]}$ [0,1]	[0,2]	**NP → NP**$_{[0,1]}$**, NP**$_{[1,3]}$ **VP → VP**$_{[0,1]}$**, NP**$_{[1,3]}$ **S → VP**$_{[0,3]}$ [0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	**NP → Det**$_{[1,2]}$**, N**$_{[2,3]}$ [1,3]	[1,4]	[1,5]
		N → flight$_{[2,3]}$ **NP → N**$_{[2,3]}$ [2,3]	[2,4]	**NP → NP**$_{[2,3]}$**, PP**$_{[3,5]}$ [2,5]
			Prep → through$_{[3,4]}$ [3,4]	**PP → Prep**$_{[3,4]}$**,NP**$_{[4,5]}$ [3,5]
				N → houston$_{[4,5]}$ **NP → N**$_{[4,5]}$ [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

CKY Parsing

N → book$_{[0,1]}$ V → book$_{[0,1]}$ NP → N$_{[0,1]}$ VP → V$_{[0,1]}$ S → VP$_{[0,1]}$ [0,1]	[0,2]	NP → NP$_{[0,1]}$, NP$_{[1,3]}$ VP → VP$_{[0,1]}$, NP$_{[1,3]}$ S → VP$_{[0,3]}$ [0,3]	[0,4]	[0,5]
	Det → the$_{[1,2]}$ [1,2]	NP → Det$_{[1,2]}$, N$_{[2,3]}$ [1,3]	[1,4]	NP → NP$_{[1,3]}$, PP$_{[3,5]}$ [1,5]
		N → flight$_{[2,3]}$ NP → N$_{[2,3]}$ [2,3]	[2,4]	NP → NP$_{[2,3]}$, PP$_{[3,5]}$ [2,5]
			Prep → through$_{[3,4]}$ [3,4]	PP → Prep$_{[3,4]}$,NP$_{[4,5]}$ [3,5]
				N → houston$_{[4,5]}$ NP → N$_{[4,5]}$ [4,5]
Book	the	flight	through	Houston

0 1 2 3 4 5

CKY Parsing

N → **book**$_{[0,1]}$ V → **book**$_{[0,1]}$ NP → N$_{[0,1]}$ VP → V$_{[0,1]}$ S → VP$_{[0,1]}$ [0,1]	[0,2]	NP → NP$_{[0,1]}$, NP$_{[1,3]}$ VP → VP$_{[0,1]}$, NP$_{[1,3]}$ S → VP$_{[0,3]}$ [0,3]	[0,4]	VP → VP$_{[0,1]}$, NP$_{[1,5]}$ VP' → VP$_{[0,3]}$, PP$_{[3,5]}$ S → VP$_{[0,5]}$ S → VP'$_{[0,5]}$ [0,5]
	Det → **the**$_{[1,2]}$ [1,2]	NP → Det$_{[1,2]}$, N$_{[2,3]}$ [1,3]	[1,4]	NP → NP$_{[1,3]}$, PP$_{[3,5]}$ [1,5]
		N → **flight**$_{[2,3]}$ NP → N$_{[2,3]}$ [2,3]	[2,4]	NP → NP$_{[2,3]}$, PP$_{[3,5]}$ [2,5]
			Prep → **through**$_{[3,4]}$ [3,4]	PP → Prep$_{[3,4]}$,NP$_{[4,5]}$ [3,5]
				N → **houston**$_{[4,5]}$ NP → N$_{[4,5]}$ [4,5]

ambiguity

0 Book 1 the 2 flight 3 through 4 Houston 5

Outline

Probabilistic Context Free Grammar (PCFG)

- Grammar G consists of
  - Terminals (T)
  - Non-terminals (N)
  - Start symbol (S)
  - Rules (R)
  - Probability function (P)
    - $P: \; R \rightarrow [0, 1]$
    - $\forall X \in N, \sum_{X \rightarrow \lambda \in R} P(X \rightarrow \lambda) = 1$

CFG

$S \rightarrow NP \;\; VP$
$S \rightarrow VP$
$NP \rightarrow N$
$NP \rightarrow Det\ N$
$NP \rightarrow NP \;\; NP$
$NP \rightarrow NP \;\; PP$
$VP \rightarrow V$
$VP \rightarrow VP \;\; PP$
$VP \rightarrow VP \;\; NP$
$PP \rightarrow Prep \;\; NP$

$N \rightarrow$ book
$V \rightarrow$ book
$Det \rightarrow$ the
$N \rightarrow$ flight
$Prep \rightarrow$ through
$N \rightarrow$ Houston

PCFG

$S \rightarrow NP \;\; VP$	0.9				
$S \rightarrow VP$	0.1				
$NP \rightarrow N$	0.3		$N \rightarrow$ book	0.5	
$NP \rightarrow Det \; N$	0.4		$V \rightarrow$ book	1.0	
$NP \rightarrow NP \;\; NP$	0.1		$Det \rightarrow$ the	1.0	
$NP \rightarrow NP \;\; PP$	0.2		$N \rightarrow$ flight	0.4	
$VP \rightarrow V$	0.1		$Prep \rightarrow$ through	1.0	
$VP \rightarrow VP \;\; PP$	0.3		$N \rightarrow$ Houston	0.1	
$VP \rightarrow VP \;\; NP$	0.6				
$PP \rightarrow Prep \;\; NP$	1.0				

Treebank

- A treebank is a corpus in which each sentence has been paired with a parse tree

- These are generally created by
  - Parsing the collection with an automatic parser
  - Correcting each parse by human annotators if required

- Requirement:
  detailed annotation guidelines that provide
  - A POS tagset
  - A grammar
  - Annotation schema
    - Instructions for how to deal with particular grammatical constructions

Penn Treebank

- Penn Treebank is a widely used treebank for English
  - Most well-known section: Wall Street Journal Section
    - 1 M words from 1987-1989

```
(S (NP (NNP John))
   (VP (VPZ flies)
       (PP (IN to)
           (NNP Paris)))
   (. .))
```

Annotated Data

- Providing a valuable linguistic resource
- Can be used by many taggers and parsers (reusablity)
- Broad coverage
- Providing various information
  - Frequencies and distributions
- Can be used for evaluating systems

Outline

Statistical Parsing

- Considering the corresponding probabilities while parsing a sentence
- Selecting the parse tree which has the highest probability

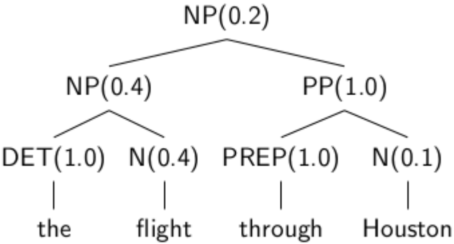
- $P(t)$: the probability of a tree t
  - Product of the probabilities of the rules used to generate the tree

PCFG

$S \rightarrow NP \;\; VP$	0.9
$S \rightarrow VP$	0.1
$NP \rightarrow N$	0.3
$NP \rightarrow Det\ N$	0.4
$NP \rightarrow NP \;\; NP$	0.1
$NP \rightarrow NP \;\; PP$	0.2
$VP \rightarrow V$	0.1
$VP \rightarrow VP \;\; PP$	0.3
$VP \rightarrow VP \;\; NP$	0.6
$PP \rightarrow Prep \;\; NP$	1.0

$N \rightarrow$ book	0.5
$V \rightarrow$ book	1.0
$Det \rightarrow$ the	1.0
$N \rightarrow$ flight	0.4
$Prep \rightarrow$ through	1.0
$N \rightarrow$ Houston	0.1

Statistical Parsing



$$P(t) = 0.2 \times 0.4 \times 1.0 \times 1.0 \times 0.4 \times 1.0 \times 0.1 = 0.0032$$

Probabilistic CKY Parsing

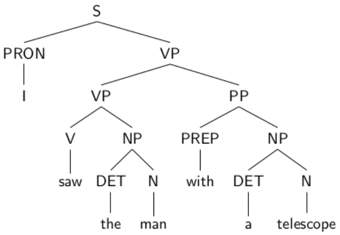
0.5 **N** → **book**$_{[0,1]}$ 1.0 **V** → **book**$_{[0,1]}$ 0.3*0.5=0.15 **NP** → **N**$_{[0,1]}$ 0.1*1.0=0.1 **VP** → **V**$_{[0,1]}$ 0.1*0.1=0.01 **S** → **VP**$_{[0,1]}$ [0,1]	0.1*0.15*0.16=0.0024 0.6*0.1*0.16=0.0096 0.1*0.0096=0.00096 [0,2]	**NP** → **NP**$_{[0,1]}$, **NP**$_{[1,3]}$ **VP** → **VP**$_{[0,1]}$, **NP**$_{[1,3]}$ **S** → **VP**$_{[0,3]}$ [0,3]	[0,4]	**VP** → **VP**$_{[0,1]}$, **NP**$_{[1,5]}$ **VP'** → **VP**$_{[0,3]}$, **PP**$_{[3,5]}$ **S** → **VP**$_{[0,5]}$ **S** → **VP'**$_{[0,5]}$ [0,5]
1.0	**Det** → **the**$_{[1,2]}$ [1,2]	**NP** → **Det**$_{[1,2]}$, **N**$_{[2,3]}$ 0.4*1.0*0.4=0.16 [1,3]	[1,4]	**NP** → **NP**$_{[1,3]}$, **PP**$_{[3,5]}$ [1,5]
	0.4 0.3*0.4=0.12	**N** → **flight**$_{[2,3]}$ **NP** → **N**$_{[2,3]}$ [2,3]	[2,4]	**NP** → **NP**$_{[2,3]}$, **PP**$_{[3,5]}$ [2,5]
			Prep → **through**$_{[3,4]}$ [3,4]	**PP** → **Prep**$_{[3,4]}$,**NP**$_{[4,5]}$ [3,5]
				N → **houston**$_{[4,5]}$ **NP** → **N**$_{[4,5]}$ [4,5]
Book	the	flight	through	Houston

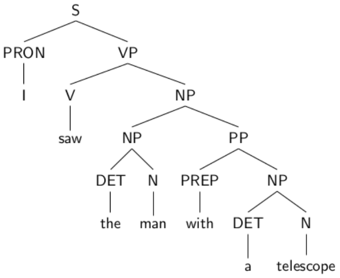
0 1 2 3 4 5

Parsing Ambiguity

- The main source of ambiguities in parsing
  - Finding the correct place of attachments

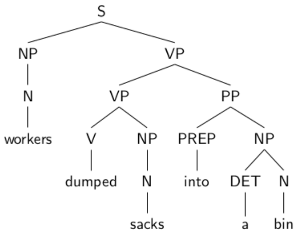
I saw the man with a telescope.

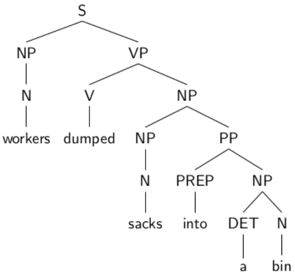




Problems

- PCFG considers no knowledge about the words
- Strong independent assumption in parse trees





Problems

(a)

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

(b)

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

Problems

- Knowing about lexicon help us to select a better option for merging constituents

Local Tree	*come*	*take*	*think*	*want*
VP → V	9.5%	2.6%	4.6%	5.7%
VP → V NP	1.1%	32.1%	0.2%	13.9%
VP → V PP	34.5%	3.1%	7.1%	0.3%
VP → V SBAR	6.6%	0.3%	73.0%	0.2%
VP → V S	2.2%	1.3%	4.8%	70.8%
VP → V NP S	0.1%	5.7%	0.0%	0.3%
VP → V PRT NP	0.3%	5.8%	0.0%	0.0%
VP → V PRT PP	6.1%	1.5%	0.2%	0.0%

Outline

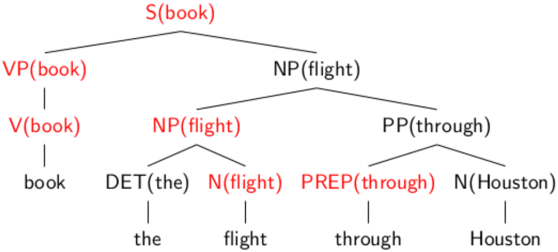
Headed CFG

- Each context-free rule has one special child that is the head of the rule

$S \rightarrow NP\ \ VP$
$S \rightarrow VP$
$NP \rightarrow N$
$NP \rightarrow Det\ N$
$NP \rightarrow NP\ \ NP$
$NP \rightarrow NP\ \ PP$
$VP \rightarrow V$
$VP \rightarrow VP\ \ PP$
$VP \rightarrow VP\ \ NP$
$PP \rightarrow Prep\ \ NP$

Lexicalization of a tree

- Trees with headwords
  - Assign a head word to each constituent and transfer it to the parents
  - Each constituent received its headword from its head child



Lexicalized PCFG

- Grammar G consists of
  - Terminals (T)
  - Non-terminals (N)
  - Start symbol (S)
  - Rules (R)

- Rules in CFG are written as follows:
  $N \rightarrow N_1 \;\; N_2$
  $N \rightarrow T$

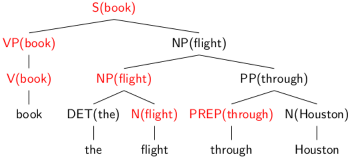
- Rules in lexicalized PCFG are written as follows:
  $N(X) \rightarrow N_1(X) \;\; N_2(Y)$
  $N(Y) \rightarrow N_1(X) \;\; N_2(Y)$
  $N(T) \rightarrow T$

Lexicalized PCFG

$S \rightarrow NP \;\; VP$
$S \rightarrow VP$
$NP \rightarrow N$
$NP(flight) \rightarrow Det(the) \; N(flight)$
$NP \rightarrow NP \;\; NP$
$NP(flight) \rightarrow NP(flight) \;\; PP(through)$
$VP(book) \rightarrow V(book)$
$VP \rightarrow VP \;\; PP$
$VP \rightarrow VP(book) \;\; NP(flight)$
$PP(through) \rightarrow Prep(through) \;\; NP(Hauston)$

Parameter Estimation in LPCFG

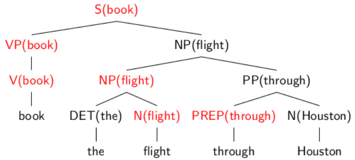
$$S(book) \rightarrow VP(book) \;\; NP(flight)$$



$$P(S \rightarrow VP \;\; NP)$$

$$P(S(book) \rightarrow_1 VP(book) \;\; NP(flight))$$

Parameter Estimation in LPCFG

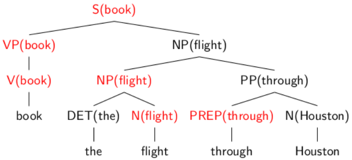


$$P(S(book) \rightarrow_1 VP(book)\ \ NP(flight))$$

$$S(book) \rightarrow_1 VP(book)\ \ NP$$

$$S(book) \rightarrow_1 VP(book)\ \ NP(\underset{flight}{- - -})$$

Parameter Estimation in LPCFG



$$P(S(book) \rightarrow_1 VP(book) \;\; NP(flight)) =$$

$$P(S \rightarrow_1 VP \;\; NP | S, book) \times P(flight | S \rightarrow_1 VP \;\; NP, book)$$

Parameter Estimation in LPCFG

- Using smoothing to estimate each probability

$$P(S(book) \rightarrow_1 VP(book) \;\; NP(flight)) =$$
$$P(S \rightarrow_1 VP \;\; NP|S, book) \times P(flight|S \rightarrow_1 VP \;\; NP, book)$$

$$P(S \rightarrow_1 VP \;\; NP|S, book) =$$
$$\lambda_1 P_{ML}(S \rightarrow_1 VP \;\; NP|S, book) + \lambda_2 P_{ML}(S \rightarrow_1 VP \;\; NP|S)$$

$$P_{ML}(S \rightarrow_1 VP \;\; NP|S, book) = \frac{\#(S(book) \rightarrow_1 VP \;\; NP)}{\#(S(book))}$$

$$P_{ML}(S \rightarrow_1 VP \;\; NP|S) = \frac{\#(S \rightarrow_1 VP \;\; NP)}{\#(S)}$$

Parameter Estimation in LPCFG

- Using smoothing to estimate each probability

$$P(S(book) \rightarrow_1 VP(book) \;\; NP(flight)) = \\ P(S \rightarrow_1 VP \;\; NP | S, book) \times P(flight | S \rightarrow_1 VP \;\; NP, book)$$

$$P(flight | S \rightarrow_1 VP \;\; NP, book) = \\ \lambda_3 P_{ML}(flight | S \rightarrow_1 VP \;\; NP, book) + \lambda_4 P_{ML}(flight | S \rightarrow_1 VP \;\; NP) + \lambda_5 P_{ML}(flight | NP)$$

$$P_{ML}(flight | S \rightarrow_1 VP \;\; NP, book) = \frac{\#(S(book) \rightarrow_1 VP(book) \;\; NP(flight))}{\#(S(book) \rightarrow_1 VP(book) \;\; NP)}$$

$$(flight | S \rightarrow_1 VP \;\; NP) = \frac{\#(S \rightarrow_1 VP \;\; NP(flight))}{\#(S \rightarrow_1 VP \;\; NP)}$$

$$P_{ML}(flight | NP) = \frac{\# NP(flight)}{\#(NP)}$$

Lexicalized PCFG

- Number of rules increases dramatically

- $\#$ rules
  - in CFG: $O(|N|^3)$
  - in LPCFG: $O(|\Sigma|^2 \times |N|^3)$

- Running time for parsing an *n* words sentence
  - in CFG: $O(n^3|N|^3)$
  - in LPCFG: $O(n^3|\Sigma|^2|N|^3)$

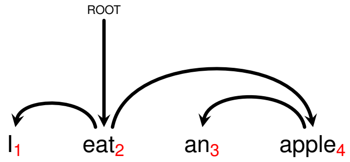
- But:
  Considering the observed words in the sentence, the number of rules will be reduced to $O(n^2 \times |N|^3)$ for parsing each sentence
  $\Rightarrow$ running time of LPCFG: $O(n^5|N|^3)$

Outline

1. Constituency vs. Dependency
2. Context Free Grammar
3. Probabilistic Context Free Grammar
4. Lexicalized PCFG
5. Dependency Parsing
6. Evaluation

Unlabeled Dependency Parses

- A special symbol, called root, is used to point the head of the sentence
- Each dependency connect a head word *h* to a modifier *m*

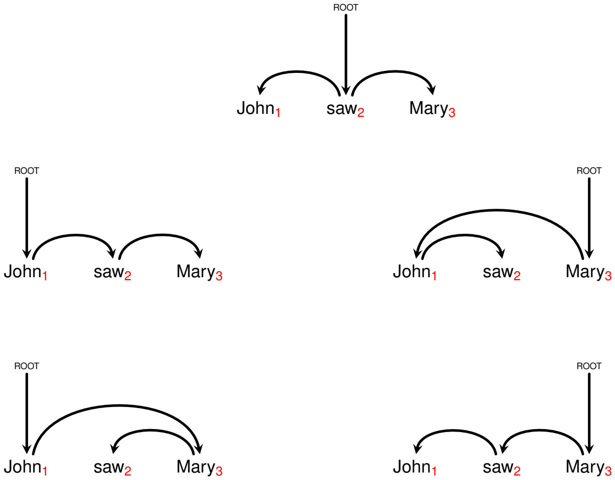


- Dependencies in the above example are as follows:
  (0,2)
  (2,1)
  (2,4)
  (4,3)

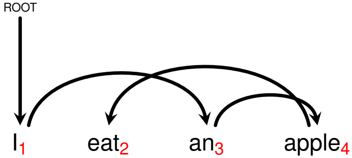
Dependency Structure

- The dependency arcs form a directed tree, with the root symbol at the root of the tree
- For each word there is a directed path from root
- There is no crossing dependencies (called projective structure)

Example: Possible Dependency Structures for a Sentence



Example: Non-projective Dependency Structures for a Sentence



Dependency Parsing with Global Linear Model

ALGORITHM:

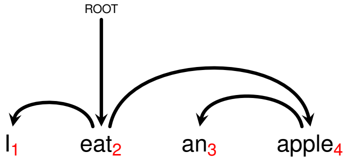
1. For an input sentence s, list all possible dependency parse trees $GEN(x)$
2. For each parse tree y, create a global feature vector $f(x, y)$ based on the local feature vectors $g(x, h, m)$

$$f(x, y) = \sum_{(h,m) \in y} g(x, h, m)$$

3. Select the parse tree with the maximum value for $f(x, y)$

$$F(x) = \text{arg max}_{y \in GEN(x)} w.f(x, y)$$

Example



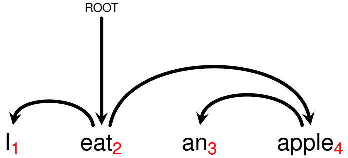
$$x = I, eat, an, apple$$

$$y = (0,2), (2,1), (2,4), (4,3)$$

$$f(x,y) = \sum_{(h,m) \in y} g(x,h,m)$$

$$f(x,y) = g(x,0,2) + g(x,2,1) + g(x,2,4) + g(x,4,3)$$

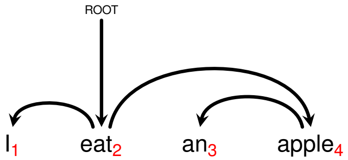
Example



$$g(x,h,m) = < g_1(x,h,m), g_2(x,h,m), g_3(x,h,m), ..., g_d(x,h,m) >$$

$$g_1(x,h,m) = \begin{cases} 1 & \text{if } x_h = \text{``}A\text{''} \;\; AND \;\; x_m = \text{``}B\text{''} \\ 0 & \text{otherwise} \end{cases}$$

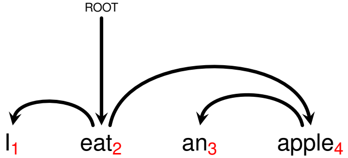
Example



$$g(x,h,m) =< g_1(x,h,m), g_2(x,h,m), g_3(x,h,m), ..., g_d(x,h,m) >$$

$$g_2(x,h,m) = \begin{cases} 1 & \text{if } POS(h) = \text{“}K\text{”} \;\; AND \;\; POS(m) = \text{“}L\text{”} \\ 0 & \text{otherwise} \end{cases}$$

Example



$$g(x, h, m) = < g_1(x, h, m), g_2(x, h, m), g_3(x, h, m), ..., g_d(x, h, m) >$$

$$g_3(x, h, m) = \begin{cases} 1 & \text{if } |h - m| > C \\ 0 & \text{otherwise} \end{cases}$$

Local Feature Vectors for Dependency Parsing

- Single features
  - The head word: w_h
  - The modifier word: w_m
  - The POS of head word: t_h
  - The POS of modifier word: t_m

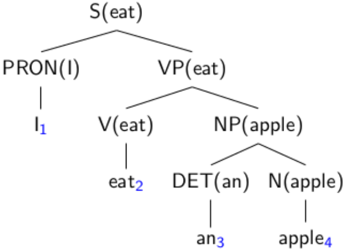
- The combination of the single features:
  $< w_h, w_m > < w_h, w_m, t_h, t_m >$

- Contextual features: t_{h-1} t_{h+1} t_{m-1} t_{m+1}

Dependency Parsing Resources

- For some languages, there exist “Dependency banks”; e.g., Czech
- If no dependency bank is available for a language, it is possible to extract it from a constituency-based treebank

Converting Lexicalized Constituency Trees to Dependency Trees



Unlabeled Dependencies:
(0,2) root $\rightarrow$ eat
(2,1) eat $\rightarrow$ I
(2,4) eat $\rightarrow$ apple
(4,3) apple $\rightarrow$ an

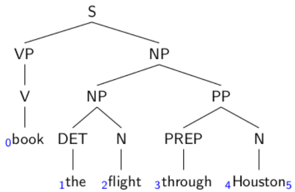
Efficiency of Dependency Parsing

- Running time for parsing an n words sentence
  - in CFG: $O(n^3|N|^3)$
  - in LPCFG: $O(n^5|N|^3)$
  - in Dependency: $O(n^3)$

$\Rightarrow$ Dependency parsing is very efficient and useful

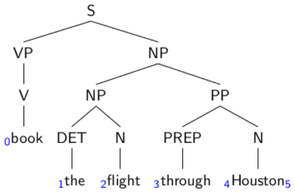
Outline

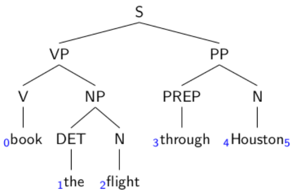
Constituents in Parse Tree



Label	Start	End
S	0	5
VP	0	1
NP	1	5
NP	1	3
PP	3	5

Constituents in Parse Tree





Label	Start	End
S	0	5
VP	0	1
NP	1	5
NP	1	3
PP	3	5

Label	Start	End
S	0	5
VP	0	3
NP	1	3
PP	3	5

Precision/Recall Evaluation

Label	Start	End
S	0	5
VP	0	1
NP	1	5
NP	1	3
PP	3	5

Label	Start	End
S	0	5
VP	0	3
NP	1	3
PP	3	5

$$\text{Labeled Precision} = \frac{3}{4} = 0.75$$

$$\text{Labeled Recall} = \frac{3}{5} = 0.6$$

$$F_1 = 0.66$$

Precision/Recall Evaluation

- Labeled precision and labeled recall consider the beginning and end of brackets as well as the label of detected constituencies

- Unlabeled precision and unlabeled recall only match the beginning and end of brackets regardless of their label

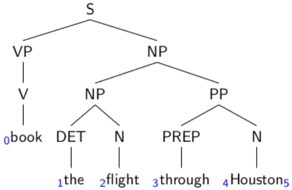
Results

- Available results on Penn treebank (Wall Street Journal)
  - # train sentences: 40,000
  - # test sentences: 2,400

  - PCFG: 70.6% Recall & 74.8% Precision
  - LPCFG: 88.1% Recall & 88.3% Precision

  - More results:
    - (Charniak & Johnson 2005): 91.2% Recall & 91.8% Precision
    - (Carreras et al. 2008): 90.7% Recall & 91.4% Precision
    - (Petrov 2010): 91.7% Recall & 92.0% Precision

Dependencies in Parse Tree



Head	Word	Rule
Root	book_1	ROOT
book_1	flight_3	$S \rightarrow_1 VP\ \ NP$
flight_3	through_4	$NP \rightarrow_1 NP\ \ PP$
flight_3	the_2	$NP \rightarrow_2 DET\ \ N$
through_4	Houston_5	$PP \rightarrow_1 PREP\ \ N$

Dependency Accuracy

- The number of dependencies in both gold and guess parse trees are equal to the number of words

- Dependency accuracy:
  The number of dependencies matches in both trees

Dependency Accuracy

Head	Word	Rule
Root	book_1	ROOT
book_1	flight_3	$S \rightarrow_1 VP\ \ NP$
flight_3	through_4	$NP \rightarrow_1 NP\ \ PP$
flight_3	the_2	$NP \rightarrow_2 DET\ \ N$
through_4	Houston_5	$PP \rightarrow_1 PREP\ \ N$

Head	Word	Rule
Root	book_1	ROOT
book_1	through_4	$S \rightarrow_1 VP\ \ PP$
book_1	flight_3	$VP \rightarrow_1 V\ \ NP$
flight_3	the_2	$NP \rightarrow_2 DET\ \ N$
through_4	Houston_5	$PP \rightarrow_1 PREP\ \ N$

$$\text{Dependency Accuracy} = \frac{3}{5} = 0.6$$

Results

- Comparing LPCFG with Dependency model
- Available results on Penn treebank (Wall Street Journal)

  - LPCFG (Collins 1997): 91.4% Dependency Accuracy
  - Dependency parsing (McDonald 2005): 90.7%
    - using dynamic programming

Behind Dependency Accuracy

- Precision and recall can still be used, but for a particular dependency type (e.g., $NP \rightarrow_1 NP \;\; PP$)

$\Rightarrow$ Can be used for error analysis

  - subject-verb: above 95% recall and precision
  - object-verb: above 92% recall and precision
  - PP attachments $\approx$ 82% recall and precision
  - Coordination $\approx$ 61% recall and precision

Further Reading

- Speech and Language Processing
  - Chapters 12, 13, 14, 15