Lexicalized and Probabilistic Parsing – Part 2

ICS 482 Natural Language Processing

Lecture 15: Lexicalized and Probabilistic Parsing – Part 2

Husni Al-Muhtaseb

بسم الله الرحمن الرحيم

ICS 482 Natural Language Processing

Lecture 15: Lexicalized and Probabilistic Parsing – Part 2

Husni Al-Muhtaseb

NLP Credits and Acknowledgment

These slides were adapted from presentations of the Authors of the book

SPEECH and LANGUAGE PROCESSING:

An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition

and some modifications from presentations found in the WEB by several scholars including the following

NLP Credits and Acknowledgment

If your name is missing please contact me
muhtaseb
At
Kfupm.
Edu.
sa

NLP Credits and Acknowledgment

Husni Al-Muhtaseb
James Martin
Jim Martin
Dan Jurafsky
Sandiway Fong
Song young in
Paula Matuszek
Mary-Angela Papalaskari
Dick Crouch
Tracy Kin
L. Venkata Subramaniam
Martin Volk
Bruce R. Maxim
Jan Hajič
Srinath Srinivasa
Simeon Ntafos
Paolo Pirjanian
Ricardo Vilalta
Tom Lenaerts

Heshaam Feili
Björn Gambäck
Christian Korthals
Thomas G. Dietterich
Devika Subramanian
Duminda Wijesekera
Lee McCluskey
David J. Kriegman
Kathleen McKeown
Michael J. Ciaraldi
David Finkel
Min-Yen Kan
Andreas Geyer-Schulz
Franz J. Kurfess
Tim Finin
Nadjet Bouayad
Kathy McCoy
Hans Uszkoreit
Azadeh Maghsoodi

Khurshid Ahmad
Staffan Larsson
Robert Wilensky
Feiyu Xu
Jakub Piskorski
Rohini Srihari
Mark Sanderson
Andrew Elks
Marc Davis
Ray Larson
Jimmy Lin
Marti Hearst
Andrew McCallum
Nick Kushmerick
Mark Craven
Chia-Hui Chang
Diana Maynard
James Allan

Martha Palmer
julia hirschberg
Elaine Rich
Christof Monz
Bonnie J. Dorr
Nizar Habash
Massimo Poesio
David Goss-Grubbs
Thomas K Harris
John Hutchins
Alexandros Potamianos
Mike Rosner
Latifa Al-Sulaiti
Giorgio Satta
Jerry R. Hobbs
Christopher Manning
Hinrich Schütze
Alexander Gelbukh
Gina-Anne Levow
Guitao Gao
Qing Ma
Zeynep Altan

Previous Lectures

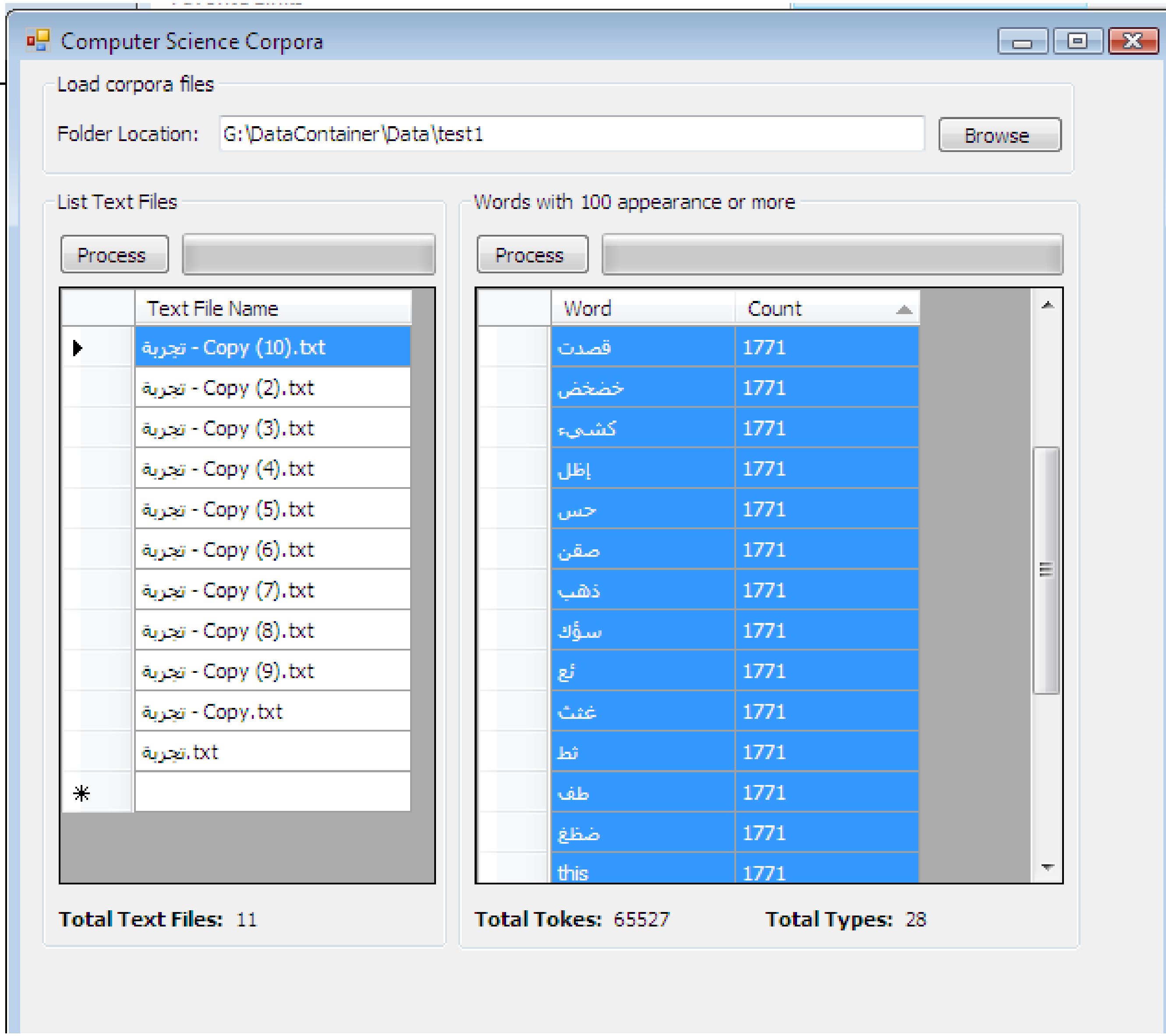
- Introduction and Phases of an NLP system
- NLP Applications - Chatting with Alice
- Finite State Automata & Regular Expressions & languages
- Morphology: Inflectional & Derivational
- Parsing and Finite State Transducers
- Stemming & Porter Stemmer
- Statistical NLP – Language Modeling
- N Grams
- Smoothing and NGram: Add-one & Witten-Bell
- Parts of Speech - Arabic Parts of Speech
- Syntax: Context Free Grammar (CFG) & Parsing
- Parsing: Earley's Algorithm
- Probabilistic Parsing

Today's Lecture

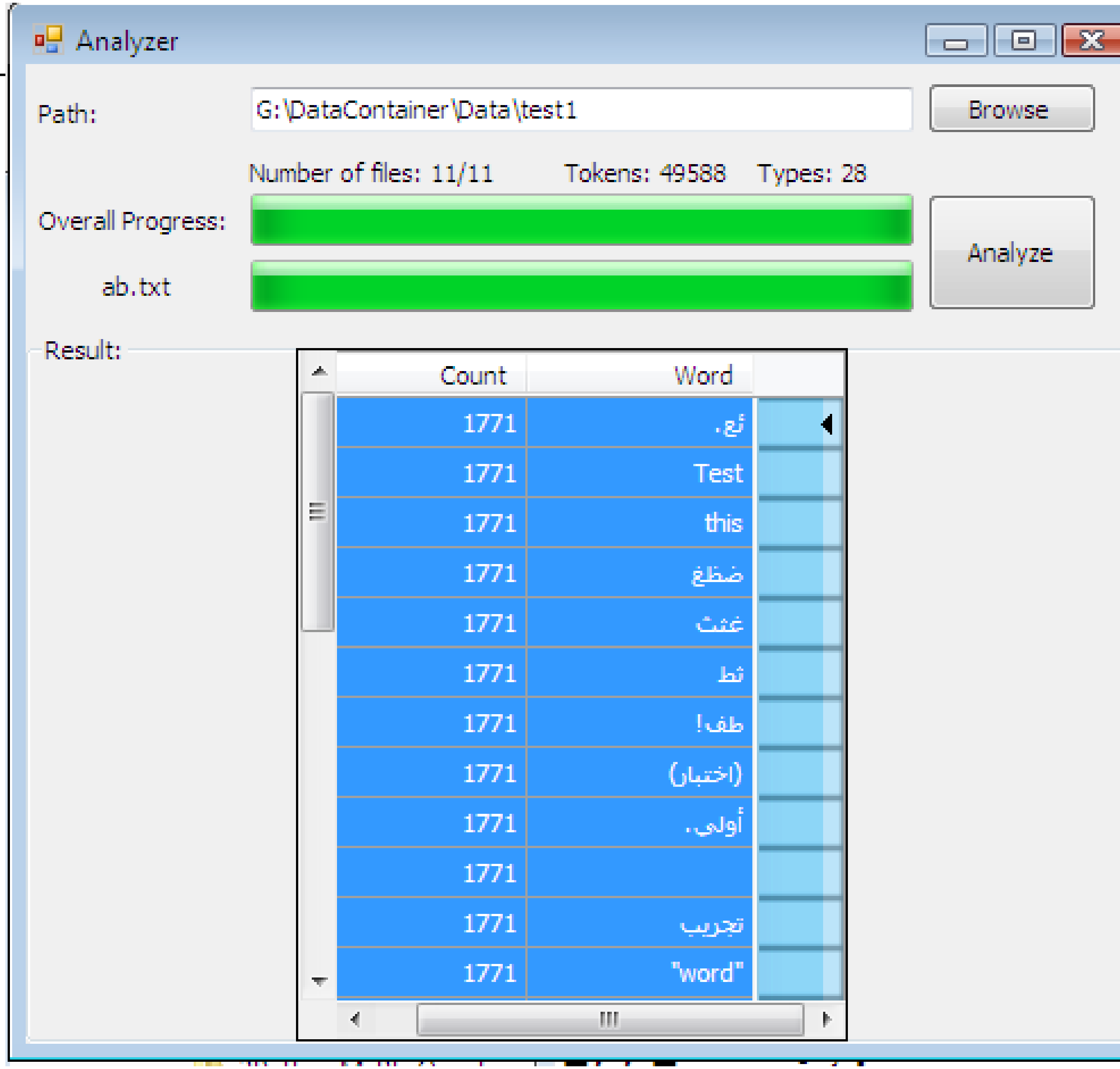
- Lexicalized and Probabilistic Parsing
  - Administration: Previous Assignments
  - Probabilistic CYK (Cocke-Younger-Kasami)
  - Dependency Grammar

Administration: Previous Assignments

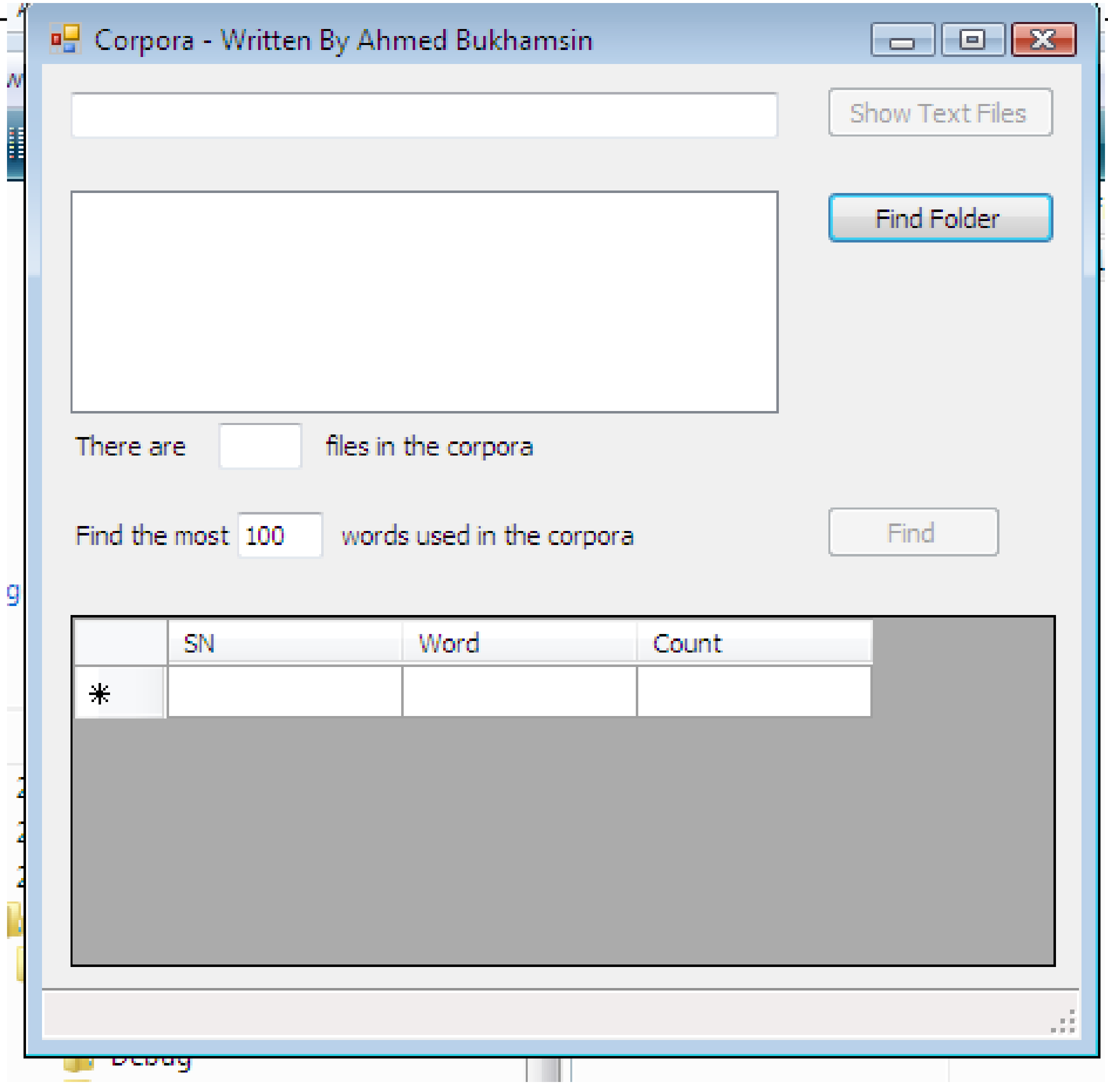
- WebCt visit

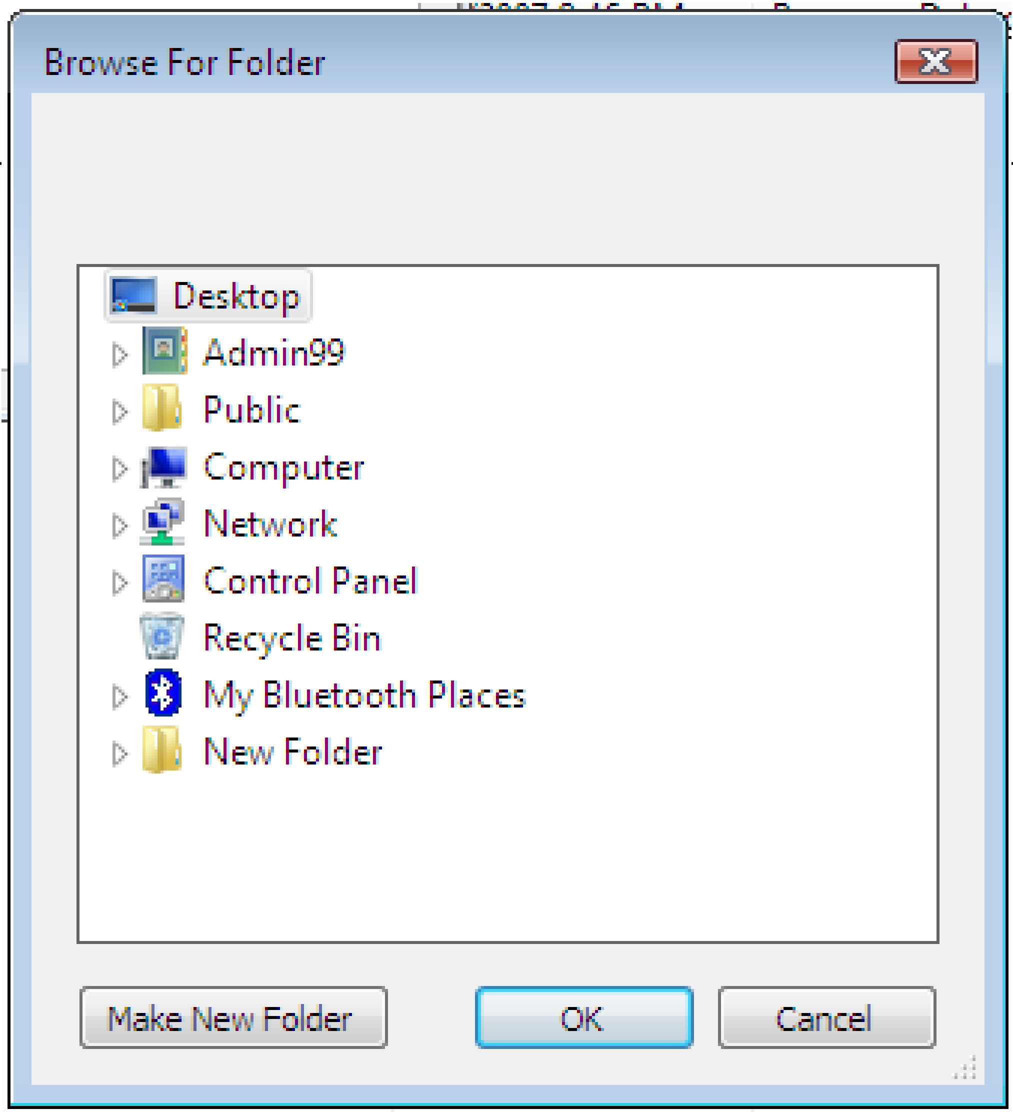

Computer Science Corpora
Load corpora files
Folder Location:
G:\DataContainer\Data\test1
Browse
List Text Files
Process
Text File Name
تجربة - Copy (10).txt
تجربة - Copy (2).txt
تجربة - Copy (3).txt
تجربة - Copy (4).txt
تجربة - Copy (5).txt
تجربة - Copy (6).txt
تجربة - Copy (7).txt
تجربة - Copy (8).txt
تجربة - Copy (9).txt
تجربة - Copy.txt
تجربة.txt
Total Text Files: 11
Words with 100 appearance or more
Process
Word
Count
قصدت 1771
خضخض 1771
كشيء 1771
إظل 1771
حس 1771
صقن 1771
ذهب 1771
سؤك 1771
ئع 1771
غثث 1771
ثط 1771
طف 1771
ضظغ 1771
this 1771
Total Tokes: 65527
Total Types: 28


عرض الملفات
ملف
ضع رقم هنا :
100
تم البحث عن الكلمات المتكررة.
أنظر أيضا إلى الملف في ملف التخزين.
الكلمة
التعداد
1
24794
2
يستعملونه
1771
3
أشكي،
1771
4
سؤك
1771
5
الفيروزآبادي
1771
6
إظل
1771
7
تجريب
1771


Analyzer
Path:
G:\DataContainer\Data\test1
Browse
Number of files: 11/11
Tokens: 49588
Types: 28
Overall Progress:
Analyze
ab.txt
Result:
Count
Word
1771 ئع.
1771 Test
1771 this
1771 ضظغ
1771 غثث
1771 ثط
1771 طف!
1771 (اختبار)
1771 أولي..
1771
1771 تجريب
1771 "word"

???


Corpora - Written By Ahmed Bukhamsin
Show Text Files
Find Folder
There are
files in the corpora
Find the most
100
words used in the corpora
Find
SN
Word
Count


Browse For Folder
Desktop
Admin99
Public
Computer
Network
Control Panel
Recycle Bin
My Bluetooth Places
New Folder
Make New Folder
OK
Cancel

الفيروزآبادي يستعملونه منبطحة فأعجزتهم أشكى، جذر قصدت خضخض كشيء. حس صقن ذهب سؤك؟ إظل غثث ثط طف! ضظغ نع. " .Test this "word" please!تجربة" تجريب (اختبار) أولي.

- الفيروزآبادي ١٧٧١
- يستعملونه ١٧٧١
- منبطحة ١٧٧١
- فأعجزتهم ١٧٧١
- أشكى، ١٧٧١
- جذر ١٧٧١
- قصدت ١٧٧١
- خضخض ١٧٧١
- كشيء ١٧٧١
- إظل ١٧٧١
- حس ١٧٧١
- صقن ١٧٧١
- ذهب ١٧٧١
- سؤك ١٧٧١
- نع ١٧٧١
- غثث ١٧٧١
- ثط ١٧٧١
- طف ١٧٧١
- ضظغ ١٧٧١
- this 1771
- Test 1771
- ""تجربة ١٧٧١
- "word" 1771
- please 1771
- تجريب ١٧٧١
- اختبار ١٧٧١
- أولي ١٧٧١
- ١٧٧١٠
-

	الكلمة	التعداد
١		٢٤٧٩٤
٢	يستعملونه	١٧٧١
٣	أشكى،	١٧٧١
٤	سؤك	١٧٧١
٥	الفيروزآبادي	١٧٧١
٦	إظل	١٧٧١
٧	تجريب	١٧٧١
٨	اختبار	١٧٧١
٩	حس	١٧٧١
١٠	طف	١٧٧١
11	this	١٧٧١
12	please	1771
١٣	أولي	1771
١٤	ثط	١٧٧١
١٥	غثث	١٧٧١
١٦	تجربة	١٧٧١
17	word	١٧٧١
١٨	ضظغ	1771
١٩	خضخض	١٧٧١
٢٠	قصدت	١٧٧١
21	Test	١٧٧١
٢٢	نع	1771
٢٣	صقن	١٧٧١
٢٤	كشيء	١٧٧١
٢٥	فأعجزتهم	١٧٧١
٢٦	منبطحة	١٧٧١
٢٧	جذر	١٧٧١

- نع. ١٧٧١
- Test ١٧٧١
- this 1771
- ضظغ 1771
- غثث ١٧٧١
- ثط ١٧٧١
- طف! ١٧٧١
- (اختبار) ١٧٧١
- أولي. ١٧٧١
- ١٧٧١
- تجريب ١٧٧١
- "word" ١٧٧١
- please!. 1771
- ""تجربة 1771
- أشكى، ١٧٧١
- جذر ١٧٧١
- قصدت ١٧٧١
- فأعجزتهم ١٧٧١
- الفيروزآبادي ١٧٧١
- يستعملونه ١٧٧١
- منبطحة ١٧٧١
- ذهب ١٧٧١
- سؤك؟ ١٧٧١
- إظل ١٧٧١
- صقن ١٧٧١
- خضخض ١٧٧١
- كشيء. ١٧٧١
- حس ١٧٧١

What should we do?

- Suggestions

Probabilistic CFGs

- The probabilistic model
  - Assigning probabilities to parse trees
- Getting the probabilities for the model
- Parsing with probabilities
  - Slight modification to dynamic programming approach
  - Task is to find the max probability tree for an input

Getting the Probabilities

- From an annotated database (a treebank)
- Learned from a corpus

Assumptions

- We're assuming that there is a grammar to be used to parse with.
- We're assuming the existence of a large robust dictionary with parts of speech
- We're assuming the ability to parse (i.e. a parser)
- Given all that… we can parse probabilistically

Typical Approach

- Bottom-up dynamic programming approach
- Assign probabilities to constituents as they are completed and placed in the table
- Use the max probability for each constituent going up

Max probability

- Say we're talking about a final part of a parse
  - $S_0 \rightarrow NP_i VP_j$

The probability of the S is…

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

The green stuff is already known. We're doing bottom-up parsing

Max

- The P(NP) is known.
- What if there are multiple NPs for the span of text in question (*0* to *i*)?
- Take the max (Why?)
- Does not mean that other kinds of constituents for the same span are ignored (i.e. they might be in the solution)

Probabilistic Parsing

- Probabilistic CYK (Cocke-Younger-Kasami) algorithm for parsing PCFG
- Bottom-up dynamic programming algorithm
- Assume PCFG is in Chomsky Normal Form (production is either $A \rightarrow B\ C$ or $A \rightarrow a$)

Chomsky Normal Form (CNF)

All rules have form:

$$A \rightarrow BC \qquad \text{and} \qquad A \rightarrow a$$

Non-Terminal Non-Terminal terminal

Examples:

$$S \to AS$$

$$S \to a$$

$$A \to SA$$

$$A \to b$$

Chomsky
Normal Form

$$S \to AS$$

$$S \to AAS$$

$$A \to SA$$

$$A \to aa$$

Not Chomsky
Normal Form

Observations

- Chomsky normal forms are good for parsing and proving theorems
- It is possible to find the Chomsky normal form of any context-free grammar

Probabilistic CYK Parsing of PCFGs

- CYK Algorithm: bottom-up parser
- Input:
  - A Chomsky normal form PCFG, G= (N, Σ, P, S, D) Assume that the N non-terminals have indices 1, 2, …, |N|, and the start symbol S has index 1
  - n words $w_1, \ldots, w_n$
- Data Structure:
  - A dynamic programming array $\pi[i,j,a]$ holds the maximum probability for a constituent with non-terminal index a spanning words $i..j$.
- Output:
  - The maximum probability parse $\pi[1,n,1]$

Base Case

- CYK fills out $\pi[i,j,a]$ by induction
- Base case
  - Input strings with length = 1 (individual words w_i)
  - In CNF, the probability of a given non-terminal A expanding to a single word w_i must come only from the rule $A \rightarrow w_i$ i.e., $P(A \rightarrow w_i)$

Probabilistic CYK Algorithm [Corrected]

```
Function CYK(words, grammar)
    return the most probable parse and its probability
For i ←1 to num_words
    for a ←1 to num_nonterminals
        If (A →w_i) is in grammar then π[i, i, a] ←P(A →w_i)
For span ←2 to num_words
    For begin ←1 to num_words – span + 1
        end ←begin + span – 1
        For m ←begin to end – 1
          For a ←1 to num_nonterminals
            For b ←1 to num_nonterminals
              For c ←1 to num_nonterminals
                prob ←π[begin, m, b] × π[m+1, end, c] × P(A →BC)
                If (prob > π[begin, end, a]) then
                    π[begin, end, a] = prob
                    back[begin, end, a] = {m, b, c}
Return build_tree(back[1, num_words, 1]), π[1, num_words, 1]
```

The CYK Membership Algorithm

Input:

- Grammar G in Chomsky Normal Form
- String w

Output:

find if $w \in L(G)$

The Algorithm

Input example:

- Grammar G:

$$S \rightarrow AB$$
$$A \rightarrow BB$$
$$A \rightarrow a$$
$$B \rightarrow AB$$
$$B \rightarrow b$$

- String : w $aabbb$

aabbb

All substrings of length 1	a	a	b	b	b
All substrings of length 2	aa	ab	bb	bb	
All substrings of length 3	aab	abb	bbb		
All substrings of length 4	aabb	abbb			
All substrings of length 5	aabbb				

$$S \to AB$$

$$A \to BB$$

$$A \to a$$

$$B \to AB$$

$$B \to b$$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
aab	abb	bbb		
aabb	abbb			
aabbb				

$$S \rightarrow AB$$

$$A \rightarrow BB$$

$$A \rightarrow a$$

$$B \rightarrow AB$$

$$B \rightarrow b$$

a A	a A	b B	b B	b B
aa	ab S,B	bb A	bb A	
aab	abb	bbb		
aabb	abbb			
aabbb				

$$S \to AB$$
$$A \to BB$$
$$A \to a$$
$$B \to AB$$
$$B \to b$$

a A	a A	b B	b B	b B
aa	ab S,B	bb A	bb A	
aab S,B	abb A	bbb S,B		
aabb A	abbb S,B			
aabbb S,B				

Therefore: $aabbb \in L(G)$

CYK Algorithm for Parsing CFG

IDEA: For each substring of a given input x, find all variables which can derive the substring. Once these have been found, telling which variables generate x becomes a simple matter of looking at the grammar, since it's in Chomsky normal form

CYK Example

- S → NP VP
- VP → V NP
- NP → NP PP
- VP → VP PP
- PP → P NP
- NP → Ahmad | Ali | Hail
- V → called
- P → from

Example: Ahmad called Ali from Hail

CYK Example

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	Ahmad	Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		called	called Ali	called Ali from	called Ali from Hail
2:			Ali	Ali from	Ali from Hail
3:				from	From Hail
4:					Hail

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	Ali from	Ali from Hail
3:				**P** (From)	From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP

NP → Ahmad | Ali | Hail V → called P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X** Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	Ali from	Ali from Hail
3:				**P** (From)	From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP

NP → Ahmad | Ali | Hail V → called P → from

0 Ahmad 1 called 2 Ali 3 from 4 Hail 5

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X** Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	Ali from	Ali from Hail
3:				**P** (From)	From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X** Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	**X** Ali from	Ali from Hail
3:				**P** (From)	From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP
NP → Ahmad | Ali | Hail V → called P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	X Ali from	Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP

NP → Ahmad | Ali | Hail V → called P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

start at \ end at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X Ahmad called	**S** Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	X Ali from	Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP　VP → V NP　NP → NP PP　VP → VP PP　PP → P NP

NP → Ahmad | Ali | Hail　V → called　P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	called Ali from Hail
2:			**NP** (Ali)	X Ali from	Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP

NP → Ahmad | Ali | Hail V → called P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X**	**S** Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	**X** called Ali from	called Ali from Hail
2:			**NP** (Ali)	**X** Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X**	**S** Ahmad called Ali	**X** Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	**X** called Ali from	called Ali from Hail
2:			**NP** (Ali)	**X** Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP

NP → Ahmad | Ali | Hail V → called P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	**VP** called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP

NP → Ahmad | Ali | Hail V → called P → from

0 Ahmad 1 called 2 Ali 3 from 4 Hail 5

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	**VP_1** called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	**VP_2** **VP_1** called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	**S** Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	**VP_2** **VP_1** called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X**	**S** Ahmad called Ali	**X** Ahmad called Ali from	**S_1** Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	**X** called Ali from	**VP_2** **VP_1** called Ali from Hail
2:			**NP** (Ali)	**X** Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	$\mathbf{S_1}$ $\mathbf{S_2}$ Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	$\mathbf{VP_2}$ $\mathbf{VP_1}$ called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

S → NP VP

VP → V NP

NP → NP PP

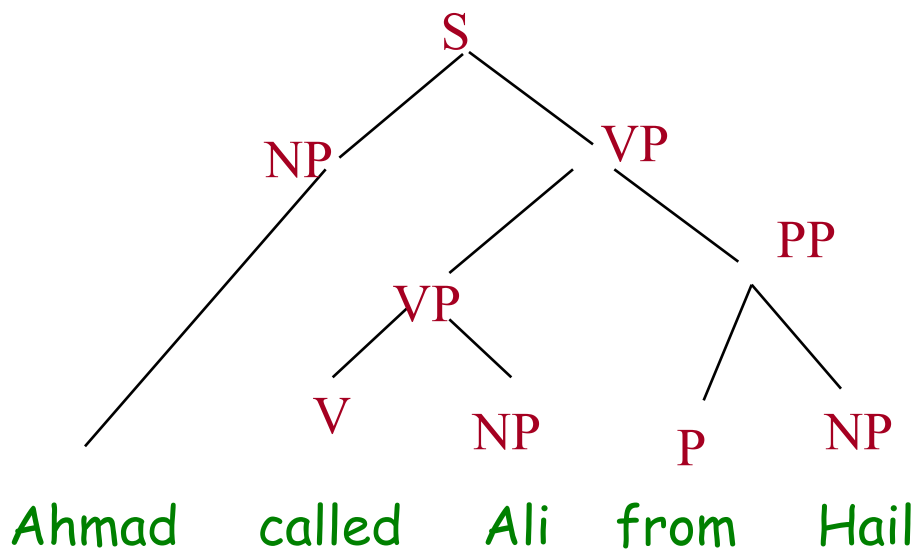
VP → VP PP

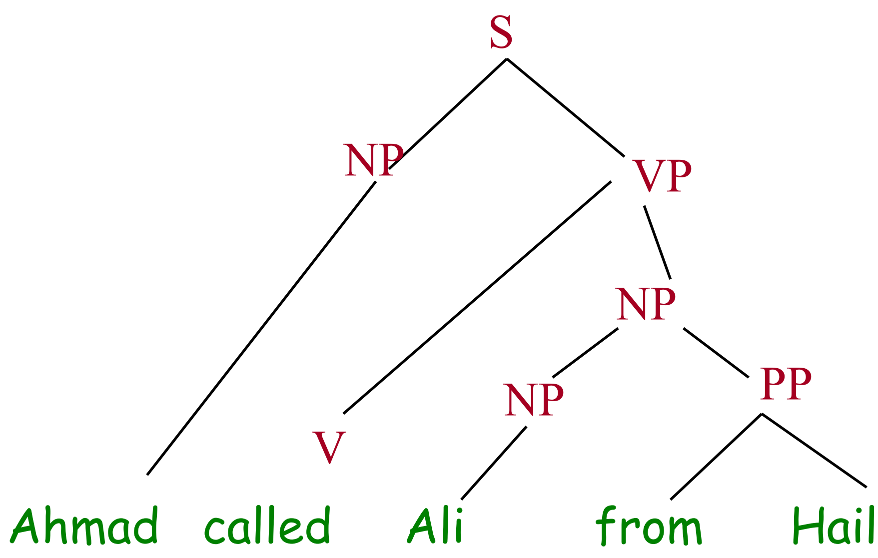
PP → P NP

NP → Ahmad | Ali | Hail

V → called

P → from





Same Example: We might see it in different format

				NP
			P	Hail
		NP	from	
	V	Ali		
NP	called			
Ahmad				

S → NP VP

VP → V NP

NP → NP PP

VP → VP PP

PP → P NP

NP → Ahmad | Ali | Hail

V → called

P → from

Example

S_1 S_2	VP_1 VP_2	NP	PP	NP
X	X	X	P	Hail
S	VP	NP	from	
X	V	Ali		
NP	called			
Ahmad				

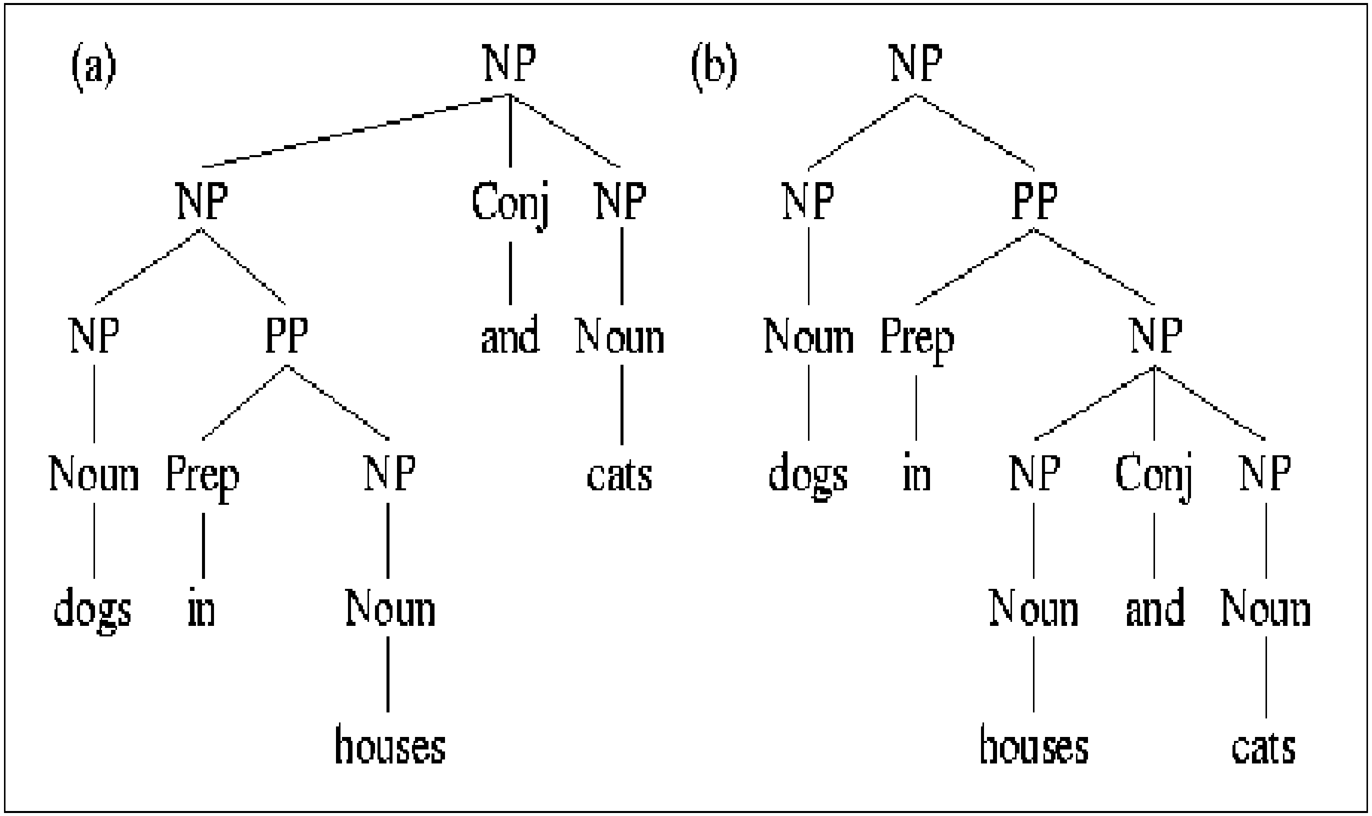
Problems with PCFGs

- The probability model we're using is just based on the rules in the derivation…
  - Doesn't take into account where in the derivation a rule is used
  - Doesn't use the words in any real way
- In PCFGs we make a number of independence assumptions.
- **Context**: Humans make wide use of context
  - Context of who we are talking to, where we are, prior context of the conversation.
  - Prior discourse context
- We need to incorporate these sources of information to build better parsers than PCFGs.

Problems with PCFG

- Lack of sensitivity to words
- Attachment ambiguity
- Coordination ambiguity
  - [[*dogs in houses*] *and*[*cats*]]
  - *dogs in* [[*houses*] *and* [*cats*]]

Problems with PCFG



Same set of rules used and hence the same probability without considering individual words

Structural context

- Assumption
  - Probabilities are context-free

    Ex: P(NP) is independent of where the NP is in the tree

Expansion	% as Subj	% as Obj
NP $\rightarrow$ PRP	13.7%	2.1%
NP $\rightarrow$ DT NN	5.6%	4.6%
NP $\rightarrow$ NP PP	5.6%	14.1%

    - Pronouns, proper names and definite NPs : Subj
    - NPs containing post-head modifiers and subcategorizes nouns : Obj
  - Need better probabilistic parser!

Lexicalization

- Frequency of common Sub-categorization frames

Local tree	*come*	*take*	*think*	*want*
VP $\rightarrow$ V	9.5%	2.6%	4.6%	5.7%
VP $\rightarrow$ V NP	1.1%	32.1%	0.2%	13.9%
VP $\rightarrow$ V PP	34.5%	3.1%	7.1%	0.3%

Solution

- Add lexical dependencies to the scheme…
  - Infiltrate the influence of particular words into the probabilities in the derivation
  - I.e. Condition on the actual words in the right way
  - All the words? No, only the right ones.
    - **Structural Context**: Certain types have locational preferences in the parse tree.

Heads

- To do that we're going to make use of the notion of the head of a phrase
  - The head of an NP is its noun
  - The head of a VP is its verb
  - The head of a PP is its preposition

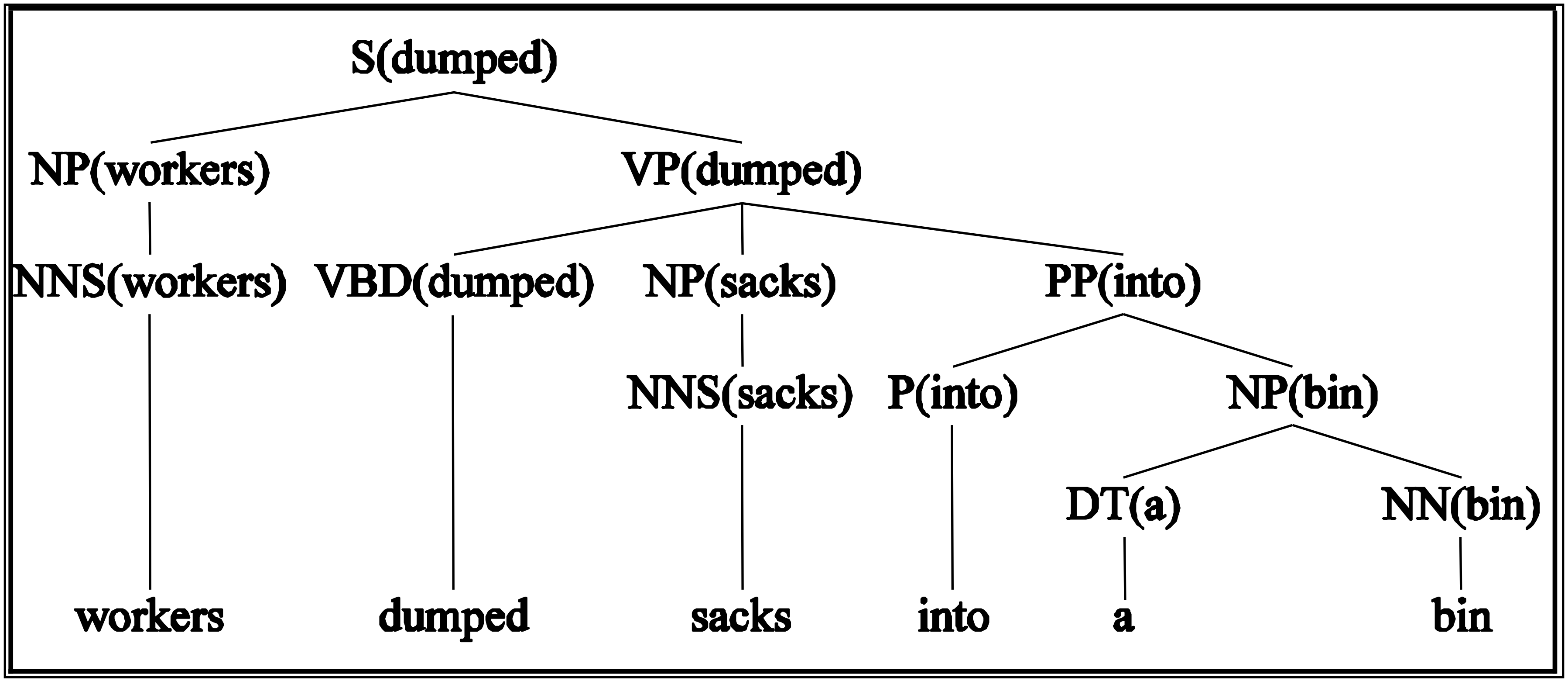
  (its really more complicated than that)

Probabilistic Lexicalized CFGs

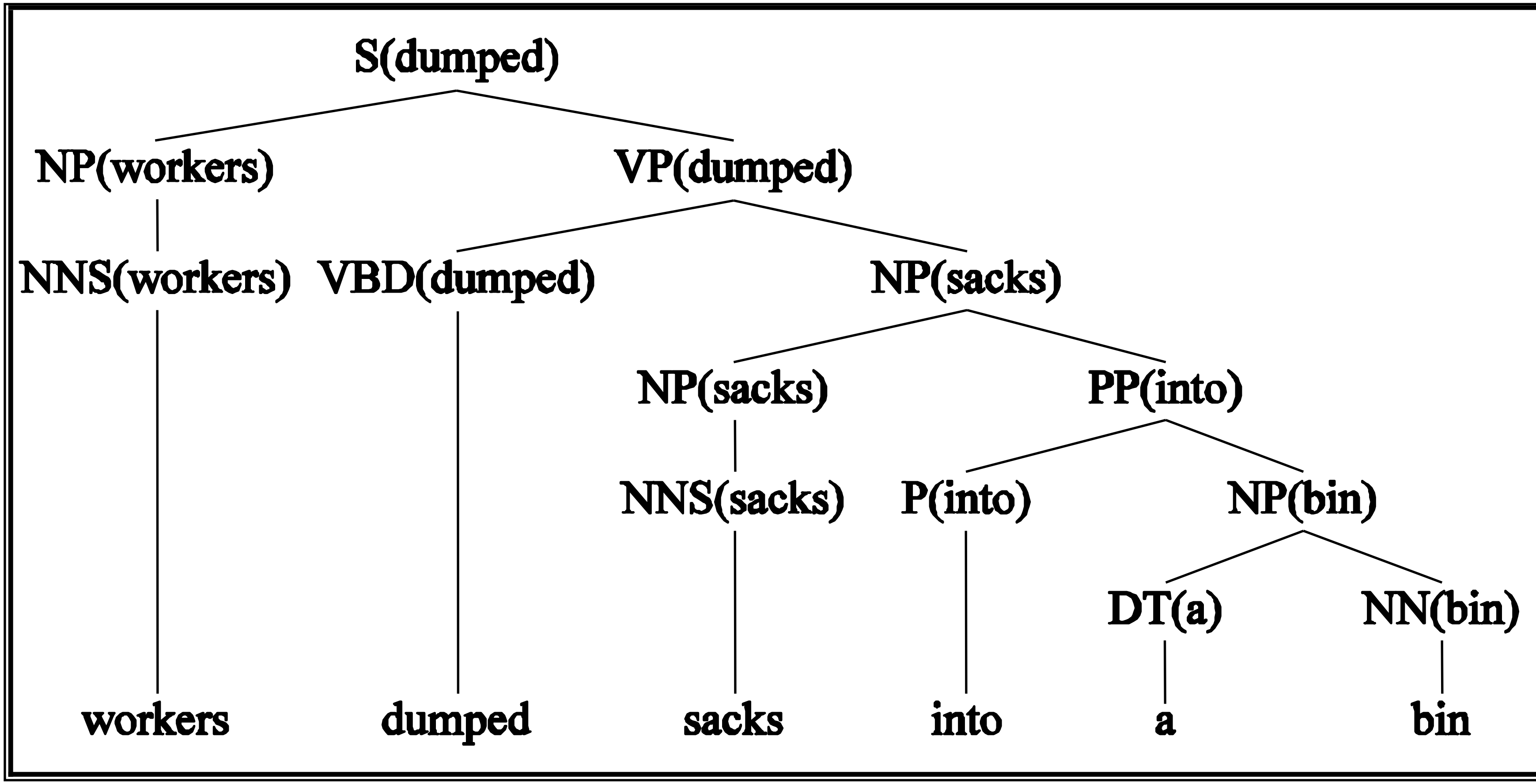
- Head child (underlined):
- S $\rightarrow$ NP VP
- VP $\rightarrow$ VBD NP
- VP $\rightarrow$ VBD NP PP
- PP $\rightarrow$ P NP
- NP $\rightarrow$ NNS
- NP $\rightarrow$ DT NN
- NP $\rightarrow$ NP PP

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

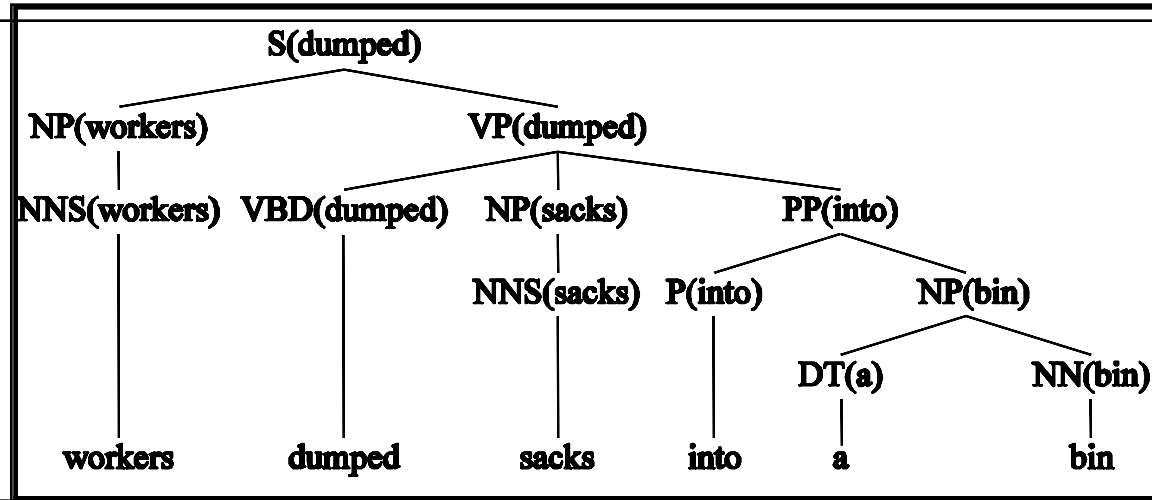
Example (right): Attribute grammar

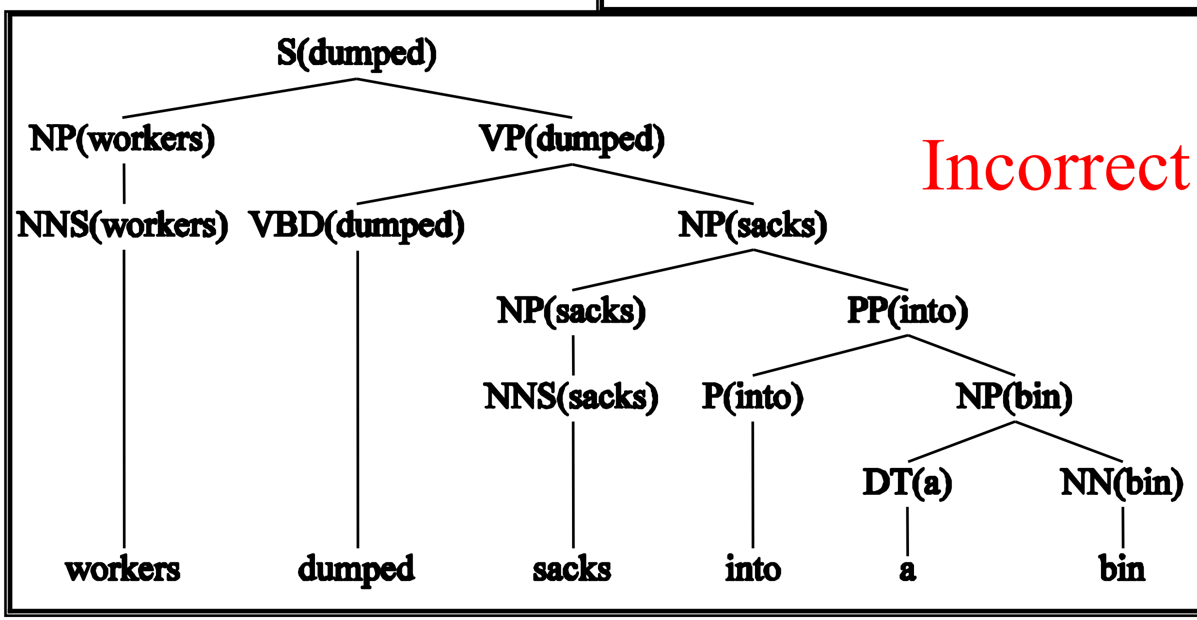


Example (wrong): Attribute grammar



Attribute grammar





Probabilities?

- We used to have
  - VP → V NP PP $\qquad\qquad p(r|\text{VP})$
    - That's the count of this rule VP → V NP PP divided by the number of VPs in a treebank
- Now we have
  - VP(dumped) → V(dumped) NP(sacks) PP(in)
  - $p(r|\text{VP}$ ^ dumped is the verb ^ sacks is the head of the NP ^ in is the head of the PP$)$
  - Not likely to have significant counts in any treebank

Sub-categorization

- Condition particular VP rules on their head… so

  r: VP → V NP PP $p(r|\text{VP})$

  Becomes

  $p(r|\text{ VP } \wedge \text{ dumped})$

  What's the count?

  How many times was this rule used with dump, divided by the number of VPs that dump appears in total

Preferences

- The issue here is the attachment of the PP. So the affinities we care about are the ones between dumped and into vs. sacks and into.
- So count the places where dumped is the head of a constituent that has a PP daughter with into as its head and normalize
- Vs. the situation where sacks is a constituent with into as the head of a PP daughter.

So We Can Solve the Dumped Sacks Problem

From the Brown corpus:

$p(VP \rightarrow VBD\ NP\ PP \mid VP, \textit{dumped}) = .67$

$p(VP \rightarrow VBD\ NP \mid VP, \textit{dumped}) = 0$

$p(\textit{into} \mid PP, \textit{dumped}) = .22$

$p(\textit{into} \mid PP, \textit{sacks}) = 0$

So, the contribution of this part of the parse to the total scores for the two candidates is:

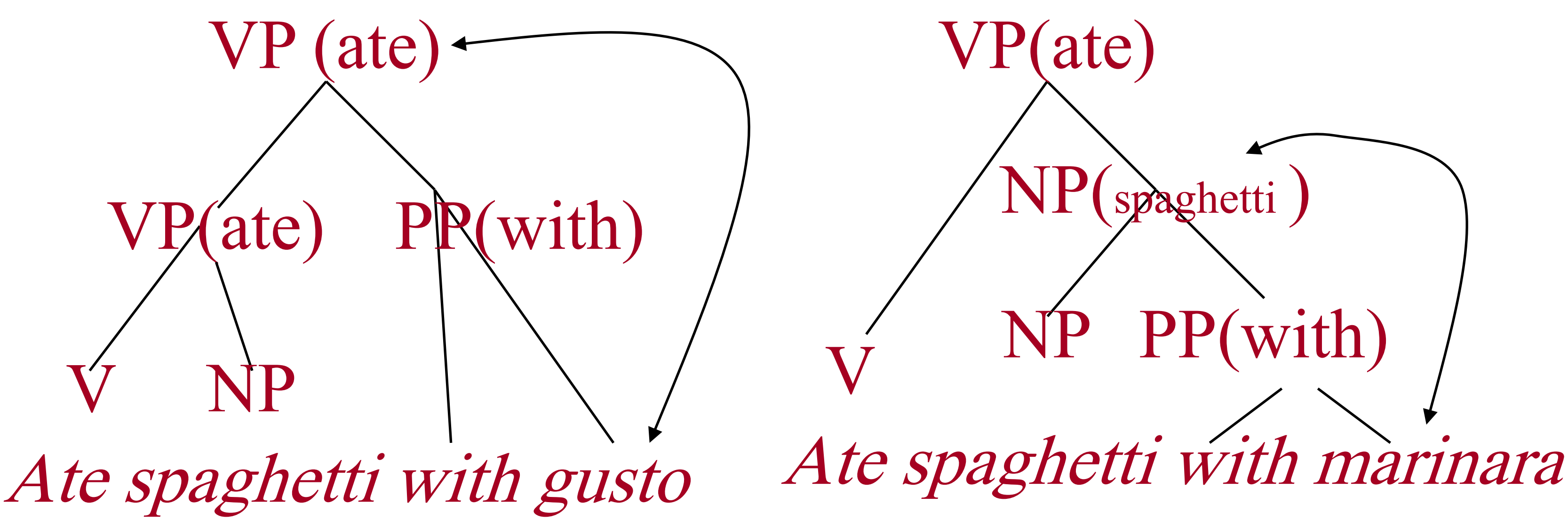
[dumped into]	$.67 \times .22$	$= .147$
[sacks into]	0×0	$= 0$

Preferences (2)

- Consider the VPs
  - Ate spaghetti with gusto ذوق
  - Ate spaghetti with marinara صلصة
- The affinity of gusto for eat is much larger than its affinity for spaghetti
- On the other hand, the affinity of marinara for spaghetti is much higher than its affinity for ate

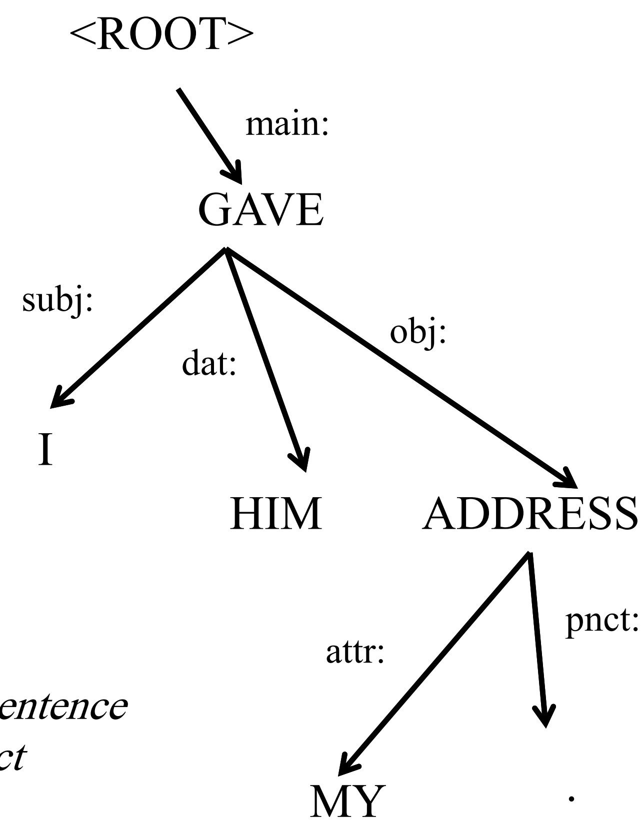
Preferences (2)

- Note the relationship here is more distant and doesn't involve a headword since gusto and marinara aren't the heads of the PPs.



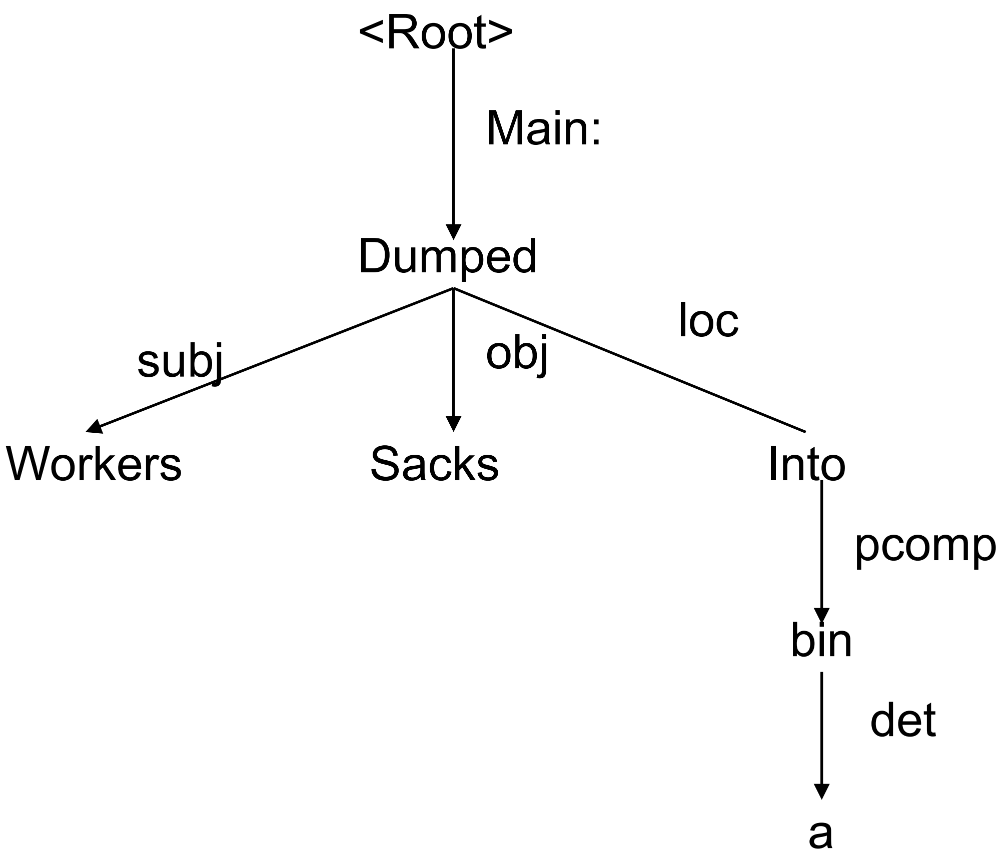
Dependency Grammars

- Based purely on lexical dependency (binary relations between words)
- Constituents and phrase-structure rules have no fundamental role



Key
Main: beginning of sentence
Subj: syntactic subject
Dat: indirect object
Obj: direct object
Attr: pre-modifying nominal
Pnct: punctuation mark

Dependency Grammar Example



Dependency	Description
subj	syntactic subject
obj	direct object
dat	indirect object
tmp	temporal adverbials
loc	location adverbials
attr	Pre-modifying nominal (possessives, etc.)
mod	nominal post-modifiers (prepositional phrases, etc.)
pcomp	Complement of a preposition
comp	Predicate nominal

Grammars Dependency

Dependency	Description
subj	syntactic subject
obj	direct object
dat	indirect object
tmp	temporal adverbials
loc	location adverbials
attr	Pre-modifying nominal (possessives, etc.)
mod	nominal post-modifiers (prepositional phrases, etc.)
pcomp	Complement of a preposition
comp	Predicate nominal

Thank you

- السلام عليكم ورحمة الله