Syntax - Introduction

Pawan Goyal

CSE, IIT Kharagpur

Week 5: Lecture 1

What is Syntax?

What is Syntax?

- Refers to the way words are arranged together, and the relationship between then.

What is Syntax?

- Refers to the way words are arranged together, and the relationship between then.
- **Language Models:** Importance of modeling word order

What is Syntax?

- Refers to the way words are arranged together, and the relationship between then.
- **Language Models:** Importance of modeling word order
- **POS categories:** An equivalence class for words

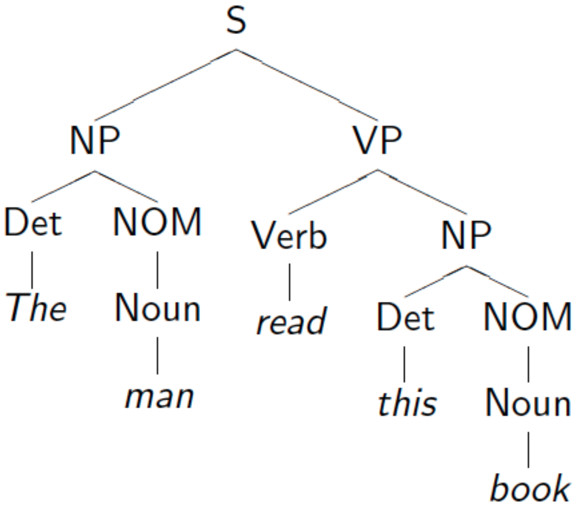
What is Syntax?

- Refers to the way words are arranged together, and the relationship between then.
- **Language Models:** Importance of modeling word order
- **POS categories:** An equivalence class for words
- More complex notions: constituency, grammatical relations, subcategorization etc.

What is Syntax?

- Refers to the way words are arranged together, and the relationship between then.
- **Language Models:** Importance of modeling word order
- **POS categories:** An equivalence class for words
- More complex notions: constituency, grammatical relations, subcategorization etc.

Syntax Tree: Example



Defining the notions: Constituency

Constituent

A group of words acts as a single unit - phrases, clauses etc.

Defining the notions: Constituency

Constituent

A group of words acts as a single unit - phrases, clauses etc.

Part of Speech - "Substitution Test"

The {sad, intelligent, green, fat, ...} one is in the corner.

Defining the notions: Constituency

Constituent

A group of words acts as a single unit - phrases, clauses etc.

Part of Speech - "Substitution Test"

The {sad, intelligent, green, fat, ...} one is in the corner.

Constituency: Noun Phrase

- *Kermit the frog*
- *they*
- *December twenty-sixth*
- *the reason he is running for president*

Constituent Phrases

Usually named based on the word that heads the constituent:

the man from Amherst	is a Noun Phrase (NP) because the head man is a noun
extremely clever	is an Adjective Phrase (AP) because the head clever is an adjective
down the river	is a Prepositional Phrase (PP) because the head down is a preposition
killed the rabbit	is a Verb Phrase (VP) because the head killed is a verb

Constituent Phrases

Usually named based on the word that heads the constituent:

the man from Amherst	is a Noun Phrase (NP) because the head man is a noun
extremely clever	is an Adjective Phrase (AP) because the head clever is an adjective
down the river	is a Prepositional Phrase (PP) because the head down is a preposition
killed the rabbit	is a Verb Phrase (VP) because the head killed is a verb

Words can also act as phrases

Joe grew potatoes

Constituent Phrases

Usually named based on the word that heads the constituent:

the man from Amherst	is a Noun Phrase (NP) because the head man is a noun
extremely clever	is an Adjective Phrase (AP) because the head clever is an adjective
down the river	is a Prepositional Phrase (PP) because the head down is a preposition
killed the rabbit	is a Verb Phrase (VP) because the head killed is a verb

Words can also act as phrases

Joe grew potatoes

Joe and *potatoes* are both nouns and noun phrases

Constituent Phrases

Usually named based on the word that heads the constituent:

the man from Amherst	is a Noun Phrase (NP) because the head man is a noun
extremely clever	is an Adjective Phrase (AP) because the head clever is an adjective
down the river	is a Prepositional Phrase (PP) because the head down is a preposition
killed the rabbit	is a Verb Phrase (VP) because the head killed is a verb

Words can also act as phrases

Joe grew potatoes

Joe and *potatoes* are both nouns and noun phrases

Compare with: *The man from Amherst grew beautiful russet potatoes*.

Constituent Phrases

Usually named based on the word that heads the constituent:

the man from Amherst	is a Noun Phrase (NP) because the head man is a noun
extremely clever	is an Adjective Phrase (AP) because the head clever is an adjective
down the river	is a Prepositional Phrase (PP) because the head down is a preposition
killed the rabbit	is a Verb Phrase (VP) because the head killed is a verb

Words can also act as phrases

Joe grew potatoes

Joe and *potatoes* are both nouns and noun phrases

Compare with: *The man from Amherst grew beautiful russet potatoes*.

Joe appears in a place that a larger noun phrase could have been.

Evidence that constituency exists

They appear in similar environments

Evidence that constituency exists

They appear in similar environments

Kermit the frog comes on stage
They come to Massachusetts every summer
December twenty-sixth comes after Christmas
The reason he is running for president comes out only now.
But not each individual word in the consituent
**The comes our... *is comes out... *for comes out...*

Evidence that constituency exists

They appear in similar environments

Kermit the frog comes on stage
They come to Massachusetts every summer
December twenty-sixth comes after Christmas
The reason he is running for president comes out only now.
But not each individual word in the consituent
**The comes our...* **is comes out...* **for comes out...*

Can be placed in a number of different locations

Evidence that constituency exists

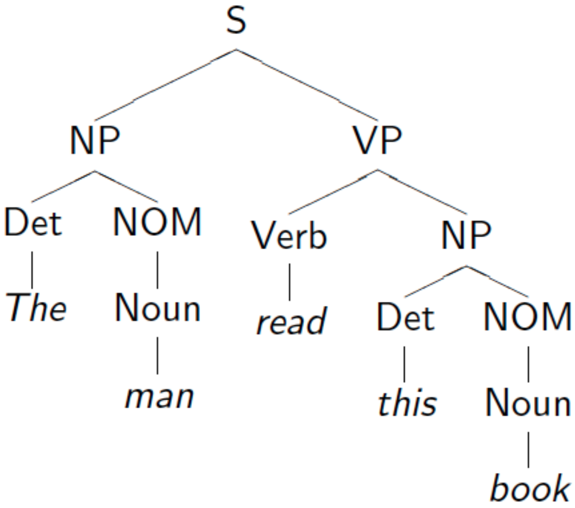
They appear in similar environments

Kermit the frog comes on stage
They come to Massachusetts every summer
December twenty-sixth comes after Christmas
The reason he is running for president comes out only now.
But not each individual word in the consituent
**The comes our... *is comes out... *for comes out...*

Can be placed in a number of different locations

Consituent = Prepositional phrase: *On December twenty-sixth*
On December twenty-sixth I'd like to fly to Florida.
I'd like to fly on December twenty-sixth to Florida.
I'd like to fly to Florida on December twenty-sixth.
But not split apart
**On December I'd like to fly twenty-sixth to Florida.*
**On I'd like to fly December twenty-sixth to Florida.*

Modeling Constituency: what tool do we need?



Modeling Constituency

Modeling Constituency

Context-free grammar

The most common way of modeling constituency

Modeling Constituency

Context-free grammar

The most common way of modeling constituency

Consists of production Rules

These rules express the ways in which the symbols of the language can be grouped and ordered together

Modeling Constituency

Context-free grammar

The most common way of modeling constituency

Consists of production Rules

These rules express the ways in which the symbols of the language can be grouped and ordered together

Example

Noun phrase can be composed of either a ProperNoun or a determiner (Det) followed by a Nominal; a Nominal can be more than one nouns

Modeling Constituency

Context-free grammar

The most common way of modeling constituency

Consists of production Rules

These rules express the ways in which the symbols of the language can be grouped and ordered together

Example

Noun phrase can be composed of either a ProperNoun or a determiner (Det) followed by a Nominal; a Nominal can be more than one nouns

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

CFG for Languages

CFG for Languages

CFG: $G = (T, N, S, R)$

- T: set of terminals
- N: set of non-terminals
  - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R: Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)*$

CFG for Languages

CFG: $G = (T, N, S, R)$

- T: set of terminals
- N: set of non-terminals
  - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R: Rules/productions of the form $X \to \gamma$, $X \in N$ and $\gamma \in (T \cup N)*$

Terminals and pre-terminals

Terminals mainly correspond to words in the language while pre-terminals mainly correspond to POS categories

CFG for Languages

CFG: $G = (T, N, S, R)$

- T: set of terminals
- N: set of non-terminals
  - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R: Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)*$

Terminals and pre-terminals

Terminals mainly correspond to words in the language while pre-terminals mainly correspond to POS categories

CFG for Languages

Example

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

CFG for Languages

Example

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

Now, these can be combined with other rules, that express facts about a lexicon.

CFG for Languages

Example

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

Now, these can be combined with other rules, that express facts about a lexicon.

$Det \rightarrow$ a

$Det \rightarrow$ the

$Noun \rightarrow$ flight

CFG for Languages

Example

$NP \rightarrow$ Det Nominal

$NP \rightarrow$ ProperNoun

$Nominal \rightarrow$ Noun | Noun Nominal

Now, these can be combined with other rules, that express facts about a lexicon.

$Det \rightarrow$ a

$Det \rightarrow$ the

$Noun \rightarrow$ flight

Can you identify the terminal, non-terminals and preterminals?

CFG as a generator

$NP \rightarrow$ Det Nominal
$NP \rightarrow$ ProperNoun
$Nominal \rightarrow$ Noun | Noun Nominal
$Det \rightarrow$ a
$Det \rightarrow$ the
$Noun \rightarrow$ flight

CFG as a generator

$NP \rightarrow$ Det Nominal
$NP \rightarrow$ ProperNoun
$Nominal \rightarrow$ Noun | Noun Nominal
$Det \rightarrow$ a
$Det \rightarrow$ the
$Noun \rightarrow$ flight
Generating ‘a flight’:

CFG as a generator

$NP \rightarrow$ Det Nominal
$NP \rightarrow$ ProperNoun
$Nominal \rightarrow$ Noun | Noun Nominal
$Det \rightarrow$ a
$Det \rightarrow$ the
$Noun \rightarrow$ flight
Generating ‘a flight’:
NP

CFG as a generator

$NP \rightarrow$ Det Nominal
$NP \rightarrow$ ProperNoun
$Nominal \rightarrow$ Noun | Noun Nominal
$Det \rightarrow$ a
$Det \rightarrow$ the
$Noun \rightarrow$ flight
Generating ‘a flight’:
$NP \rightarrow$ Det Nominal

CFG as a generator

$NP \rightarrow$ Det Nominal
$NP \rightarrow$ ProperNoun
$Nominal \rightarrow$ Noun | Noun Nominal
$Det \rightarrow$ a
$Det \rightarrow$ the
$Noun \rightarrow$ flight
Generating ‘a flight’:
$NP \rightarrow$ Det Nominal
$\rightarrow$ Det Noun

CFG as a generator

$NP \rightarrow$ Det Nominal
$NP \rightarrow$ ProperNoun
$Nominal \rightarrow$ Noun | Noun Nominal
$Det \rightarrow$ a
$Det \rightarrow$ the
$Noun \rightarrow$ flight
Generating ‘a flight’:
$NP \rightarrow$ Det Nominal
$\rightarrow$ Det Noun $\rightarrow$ a Noun

CFG as a generator

$NP \rightarrow$ Det Nominal
$NP \rightarrow$ ProperNoun
$Nominal \rightarrow$ Noun | Noun Nominal
$Det \rightarrow$ a
$Det \rightarrow$ the
$Noun \rightarrow$ flight
Generating ‘a flight’:
$NP \rightarrow$ Det Nominal
$\rightarrow$ Det Noun $\rightarrow$ a Noun $\rightarrow$ a flight

CFG as a generator

$NP \rightarrow$ Det Nominal
$NP \rightarrow$ ProperNoun
$Nominal \rightarrow$ Noun | Noun Nominal
$Det \rightarrow$ a
$Det \rightarrow$ the
$Noun \rightarrow$ flight
Generating 'a flight':
$NP \rightarrow$ Det Nominal
$\rightarrow$ Det Noun $\rightarrow$ a Noun $\rightarrow$ a flight

- Thus a CFG can be used to randomly generate a series of strings

CFG as a generator

$NP \rightarrow$ Det Nominal
$NP \rightarrow$ ProperNoun
$Nominal \rightarrow$ Noun | Noun Nominal
$Det \rightarrow$ a
$Det \rightarrow$ the
$Noun \rightarrow$ flight
Generating ‘a flight’:
$NP \rightarrow$ Det Nominal
$\rightarrow$ Det Noun $\rightarrow$ a Noun $\rightarrow$ a flight

- Thus a CFG can be used to randomly generate a series of strings
- This sequence of rule expansions is called a derivation of the string of words, usually represented as a tree

CFGs and Grammaticality

A CFG defines a formal language = set of all sentences (string of words) that can be derived by the grammar

CFGs and Grammaticality

A CFG defines a formal language = set of all sentences (string of words) that can be derived by the grammar

- Sentences in this set are said to be **grammatical**
- Sentences outside this set are said to be **ungrammatical**

CFGs and Recursion

Recursive Definition

- PP → Prep NP
- NP → Noun PP

CFGs and Recursion

Recursive Definition

- PP $\rightarrow$ Prep NP
- NP $\rightarrow$ Noun PP

Example Sentence

[$_S$The mailman ate his [$_{NP}$ lunch [$_{PP}$ with his friend [$_{PP}$ from the cleaning staff [$_{PP}$ of the building [$_{PP}$ at the intersection [$_{PP}$ on the north end [$_{PP}$ of town]]]]]]].

What does Context stand for in CFG?

What does Context stand for in CFG?

- The notion of *context* has nothing to do with the ordinary meaning of word context in language

What does Context stand for in CFG?

- The notion of *context* has nothing to do with the ordinary meaning of word context in language
- All it really means is that the non-terminal on the left-hand side of a rule is out there all by itself (free of context)

What does Context stand for in CFG?

- The notion of *context* has nothing to do with the ordinary meaning of word context in language
- All it really means is that the non-terminal on the left-hand side of a rule is out there all by itself (free of context)

$A \rightarrow BC$

- I can rewrite A as *B followed by C* regardless of the context in which A is found
- Or when I see a *B followed by a C*, I can infer an A regardless of the surrounding context

Syntax -Parsing I

Pawan Goyal

CSE, IIT Kharagpur

Week 5: Lecture 2

Grammar Rewrite Rules

S → NP VP
S → Aux NP VP
S → VP
NP → Det NOM
NOM → Noun
NOM → Noun NOM
VP → Verb
VP → Verb NP

Det → *that* | *this* | *a* | *the*
Noun → *book* | *flight* | *meal* | *man*
Verb → *book* | *include* | *read*
Aux → *does*

Grammar Rewrite Rules

S $\rightarrow$ NP VP
S $\rightarrow$ Aux NP VP
S $\rightarrow$ VP
NP $\rightarrow$ Det NOM
NOM $\rightarrow$ Noun
NOM $\rightarrow$ Noun NOM
VP $\rightarrow$ Verb
VP $\rightarrow$ Verb NP

Det $\rightarrow$ *that* | *this* | *a* | *the*
Noun $\rightarrow$ *book* | *flight* | *meal* | *man*
Verb $\rightarrow$ *book* | *include* | *read*
Aux $\rightarrow$ *does*

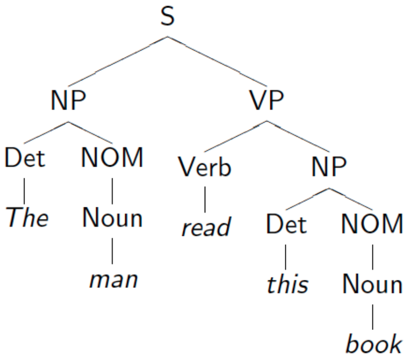
S $\rightarrow$ NP VP
$\rightarrow$ Det NOM VP
$\rightarrow$ *The* NOM VP
$\rightarrow$ *The* Noun VP
$\rightarrow$ *The man* VP
$\rightarrow$ *The man* Verb NP
$\rightarrow$ *The man read* NP
$\rightarrow$ *The man read* Det NOM
$\rightarrow$ *The man read this* NOM
$\rightarrow$ *The man read this* Noun
$\rightarrow$ *The man read this book*

Parse Tree

S $\rightarrow$ NP VP
$\rightarrow$ Det NOM VP
$\rightarrow$ *The* NOM VP
$\rightarrow$ *The* Noun VP
$\rightarrow$ *The man* VP
$\rightarrow$ *The man* Verb NP
$\rightarrow$ *The man read* NP
$\rightarrow$ *The man read* Det NOM
$\rightarrow$ *The man read this* NOM
$\rightarrow$ *The man read this* Noun
$\rightarrow$ *The man read this book*

Parse Tree

S → NP VP
→ Det NOM VP
→ *The* NOM VP
→ *The* Noun VP
→ *The man* VP
→ *The man* Verb NP
→ *The man read* NP
→ *The man read* Det NOM
→ *The man read this* NOM
→ *The man read this* Noun
→ *The man read this book*



What is Parsing?

- The process of taking a string and a grammar and returning all possible parse trees for that string

What is Parsing?

- The process of taking a string and a grammar and returning all possible parse trees for that string
- That is, find all trees, whose root is the start symbol S, which cover exactly the words in the input

What is Parsing?

- The process of taking a string and a grammar and returning all possible parse trees for that string
- That is, find all trees, whose root is the start symbol S, which cover exactly the words in the input

What are the constraints? "book that flight"

- There must be three leaves, *book*, *that* and *flight*
- The tree must have one root, the start symbol S

What is Parsing?

- The process of taking a string and a grammar and returning all possible parse trees for that string
- That is, find all trees, whose root is the start symbol S, which cover exactly the words in the input

What are the constraints? "book that flight"

- There must be three leaves, *book*, *that* and *flight*
- The tree must have one root, the start symbol S
- Give rise to two search strategies: *top-down* (goal-oriented) and *bottom-up* (data-directed)

Parsing

Grammar

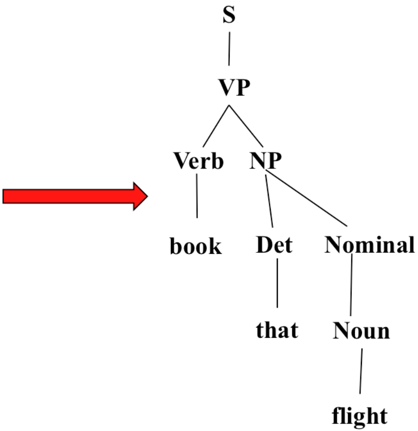
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

Lexicon

Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | he | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

Parsing

book that flight



Top-Down Parsing

- Searches for a parse tree by trying to build upon the root node S down to the leaves

Top-Down Parsing

- Searches for a parse tree by trying to build upon the root node S down to the leaves
- Start by assuming that the input can be derived by the designated start symbol S

Top-Down Parsing

- Searches for a parse tree by trying to build upon the root node S down to the leaves
- Start by assuming that the input can be derived by the designated start symbol S
- Find all trees that can start with S, by looking at the grammar rules with S on the left-hand side

Top-Down Parsing

- Searches for a parse tree by trying to build upon the root node S down to the leaves
- Start by assuming that the input can be derived by the designated start symbol S
- Find all trees that can start with S, by looking at the grammar rules with S on the left-hand side
- Trees are grown downward until they eventually reach the POS categories at the bottom

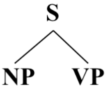
Top-Down Parsing

- Searches for a parse tree by trying to build upon the root node S down to the leaves
- Start by assuming that the input can be derived by the designated start symbol S
- Find all trees that can start with S, by looking at the grammar rules with S on the left-hand side
- Trees are grown downward until they eventually reach the POS categories at the bottom
- Trees whose leaves fail to match the words in the input can be rejected

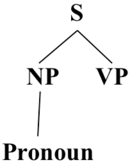
Top-Down Parsing

S

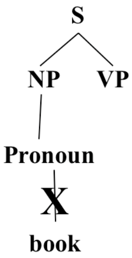
Top-Down Parsing



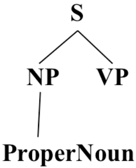
Top-Down Parsing



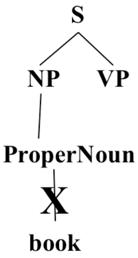
Top-Down Parsing



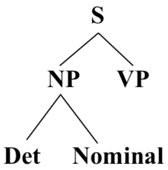
Top-Down Parsing



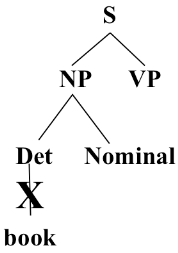
Top-Down Parsing



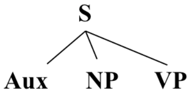
Top-Down Parsing



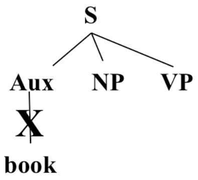
Top-Down Parsing



Top-Down Parsing

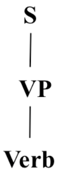


Top-Down Parsing



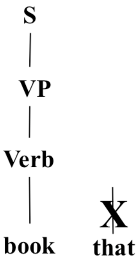
Top-Down Parsing

Top-Down Parsing

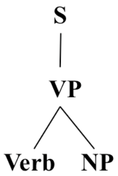


Top-Down Parsing

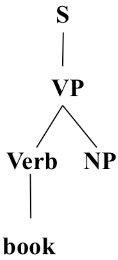
Top-Down Parsing



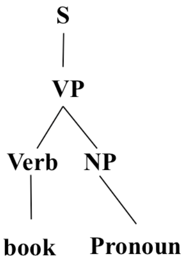
Top-Down Parsing



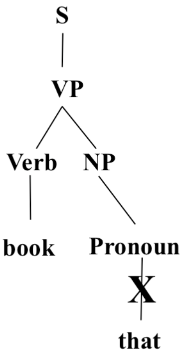
Top-Down Parsing



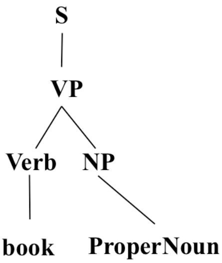
Top-Down Parsing



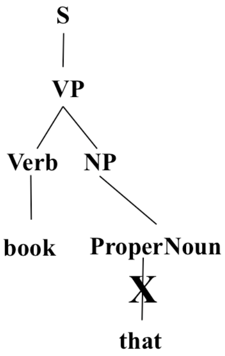
Top-Down Parsing



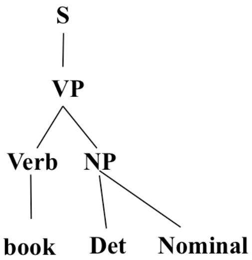
Top-Down Parsing



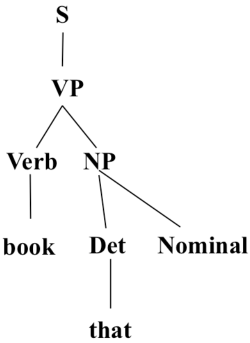
Top-Down Parsing



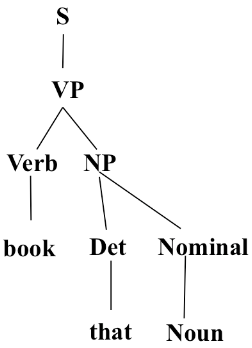
Top-Down Parsing



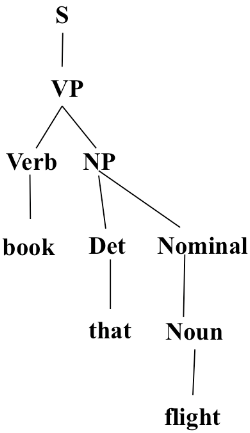
Top-Down Parsing



Top-Down Parsing



Top-Down Parsing



Bottom-Up Parsing

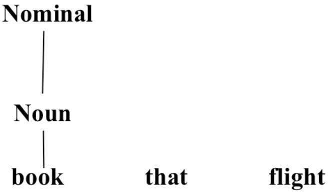
- The parser starts with the words of the input, and tries to build trees from the words up, by applying rules from the grammar one at a time
- Parser looks for the places in the parse-in-progress where the right-hand-side of some rule might fit.

Bottom-Up Parsing

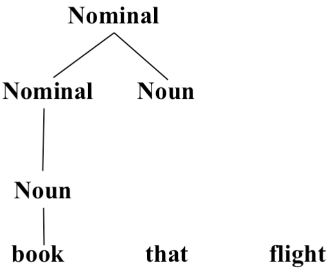
book **that** **flight**


Noun
book
that
flight

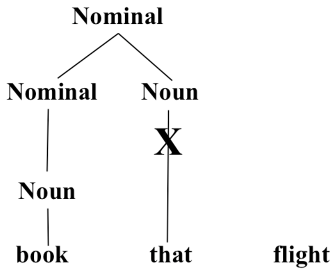
Bottom-Up Parsing



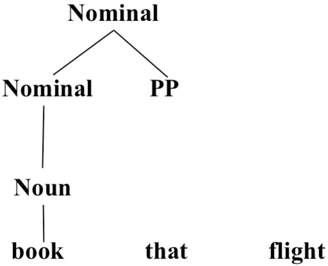
Bottom-Up Parsing



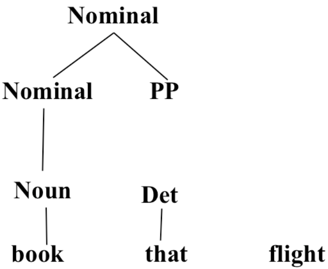
Bottom-Up Parsing



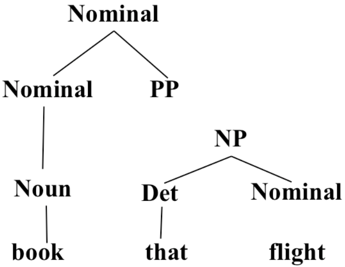
Bottom-Up Parsing



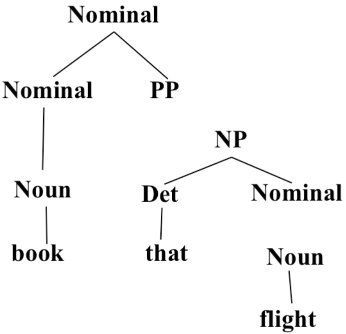
Bottom-Up Parsing



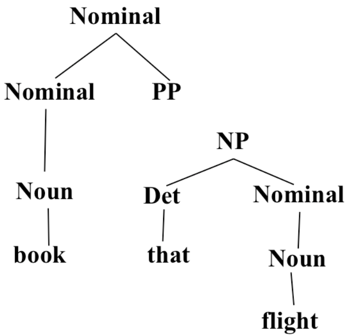
Bottom-Up Parsing



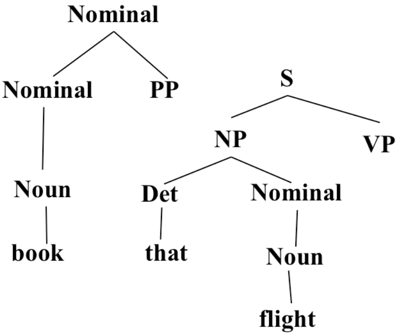
Bottom-Up Parsing



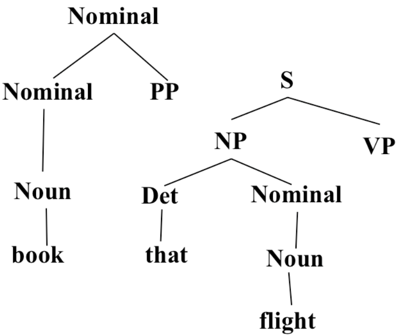
Bottom-Up Parsing



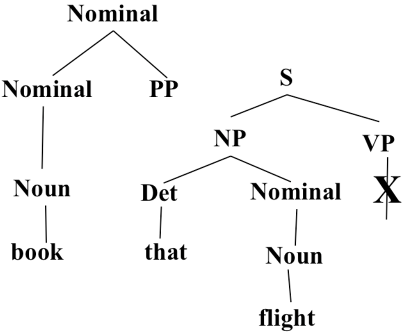
Bottom-Up Parsing



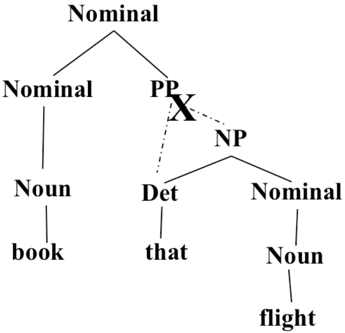
Bottom-Up Parsing



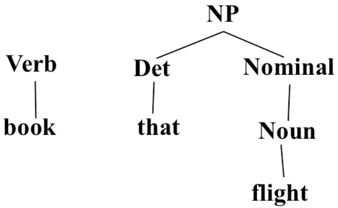
Bottom-Up Parsing



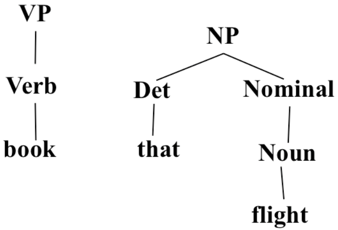
Bottom-Up Parsing



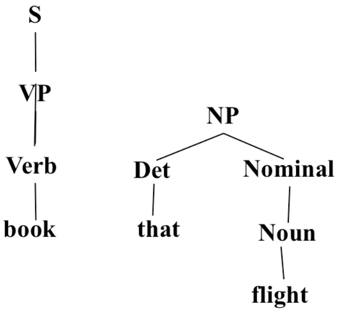
Bottom-Up Parsing



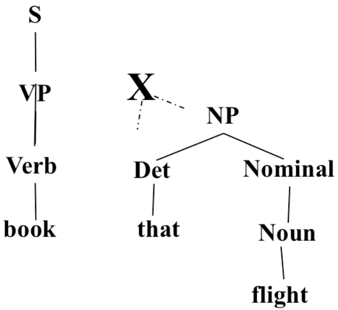
Bottom-Up Parsing



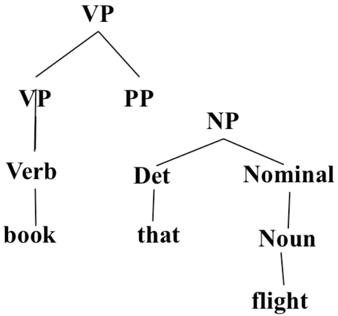
Bottom-Up Parsing



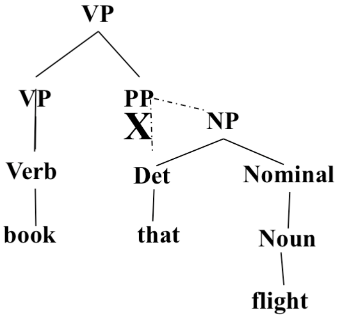
Bottom-Up Parsing



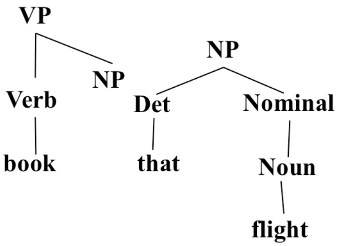
Bottom-Up Parsing



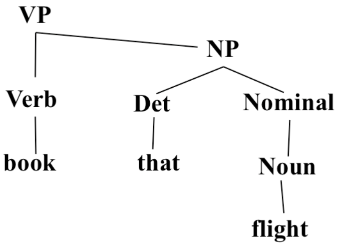
Bottom-Up Parsing



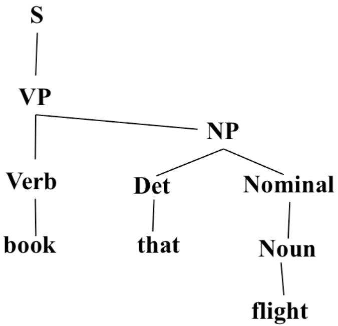
Bottom-Up Parsing



Bottom-Up Parsing



Bottom-Up Parsing



Top-Down vs. Bottom-Up

Top-Down vs. Bottom-Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.

Top-Down vs. Bottom-Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.
- Bottom up never explores options that do not connect to the actual sentence but can explore options that can never lead to a full parse.

Top-Down vs. Bottom-Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.
- Bottom up never explores options that do not connect to the actual sentence but can explore options that can never lead to a full parse.
- Relative amounts of wasted search depend on how much the grammar branches in each direction.

Dynamic Programming Parsing

Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.

Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.
- Caching (memoizing) critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs.

Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.
- Caching (memoizing) critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs.
- Dynamic programming algorithms based on both top-down and bottom-up search can achieve $O(n^3)$ recognition time where n is the length of the input string.

Dynamic Programming Parsing Methods

- CKY (Cocke-Kasami-Younger) algorithm: bottom-up, requires normalizing the grammar

Dynamic Programming Parsing Methods

- CKY (Cocke-Kasami-Younger) algorithm: bottom-up, requires normalizing the grammar
- Earley Parser - top-down, does not require normalizing grammar, more complex

Dynamic Programming Parsing Methods

- CKY (Cocke-Kasami-Younger) algorithm: bottom-up, requires normalizing the grammar
- Earley Parser - top-down, does not require normalizing grammar, more complex
- More generally, *chart parsers* retain completed phrases in a chart and can combine top-down and bottom-up searches.

CKY Algorithm

- Grammar must be converted to Chomsky normal form (CNF) in which all productions must have
  - Either, exactly two non-terminals on the RHS
  - Or, 1 terminal symbol on the RHS

CKY Algorithm

- Grammar must be converted to Chomsky normal form (CNF) in which all productions must have
  - Either, exactly two non-terminals on the RHS
  - Or, 1 terminal symbol on the RHS
- Parse bottom-up storing phrases formed from all substrings in a triangular table (chart)

Original Grammar

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP
Pronoun → I | he | she | me
Noun → book | flight | meal | money
Verb → book | include | prefer
Proper-Noun → Houston | NWA

Converting to CNF

Original Grammar

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP
Pronoun → I | he | she | me
Noun → book | flight | meal | money
Verb → book | include | prefer
Proper-Noun → Houston | NWA

Chomsky Normal Form

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → VP PP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → VP PP
PP → Prep NP
Pronoun → I | he | she | me
Noun → book | flight | meal | money
Verb → book | include | prefer
Proper-Noun → Houston | NWA

Syntax -CKY, PCFGs

Pawan Goyal

CSE, IIT Kharagpur

Week 5: Lecture 3

CKY Algorithm

- Let n be the number of words in the input. Think about $n+1$ lines separating them, numbered 0 to n.
- x_{ij} will denote the words between line i and j
- We build a table so that x_{ij} contains all the possible non-terminal spanning for words between line i and j.
- We build the Table bottom-up.

CKY Algorithm

- Let n be the number of words in the input. Think about $n+1$ lines separating them, numbered 0 to n.
- x_{ij} will denote the words between line i and j
- We build a table so that x_{ij} contains all the possible non-terminal spanning for words between line i and j.
- We build the Table bottom-up.

Home Exercise

Use CKY algorithm to find the parse tree for "Book the flight through Houston" using the CNF form shown in the previous slide.

CKY for CFG

a 1	pilot 2	likes 3	flying 4	planes 5

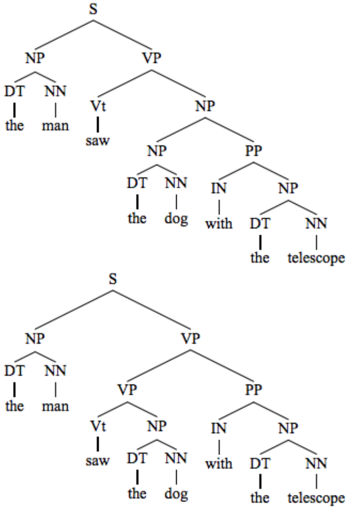
$S \longrightarrow NP\ VP$
$VP \longrightarrow VBG\ NNS$
$VP \longrightarrow VBZ\ VP$
$VP \longrightarrow VBZ\ NP$
$NP \longrightarrow DT\ NN$
$NP \longrightarrow JJ\ NNS$
$DT \longrightarrow a$
$NN \longrightarrow pilot$
$VBZ \longrightarrow likes$
$VBG \longrightarrow flying$
$JJ \longrightarrow flying$
$NNS \longrightarrow planes$

CKY for CFG

a 1	pilot 2	likes 3	flying 4	planes 5
DT	NP	-	-	S S
	NN	-	-	-
		VBZ	-	VP VP
			JJ VBG	NP VP
				NNS

$S \longrightarrow NP\ VP$
$VP \longrightarrow VBG\ NNS$
$VP \longrightarrow VBZ\ VP$
$VP \longrightarrow VBZ\ NP$
$NP \longrightarrow DT\ NN$
$NP \longrightarrow JJ\ NNS$
$DT \longrightarrow a$
$NN \longrightarrow pilot$
$VBZ \longrightarrow likes$
$VBG \longrightarrow flying$
$JJ \longrightarrow flying$
$NNS \longrightarrow planes$

What about Ambiguities?



Probabilistic Context-free grammars (PCFGs)

PCFG: $G = (T, N, S, R, P)$

- T: set of terminals
- N: set of non-terminals
  - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R: Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)*$

Probabilistic Context-free grammars (PCFGs)

PCFG: $G = (T, N, S, R, P)$

- T: set of terminals
- N: set of non-terminals
  - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R: Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)*$
- $P(R)$ gives the probability of each rule.

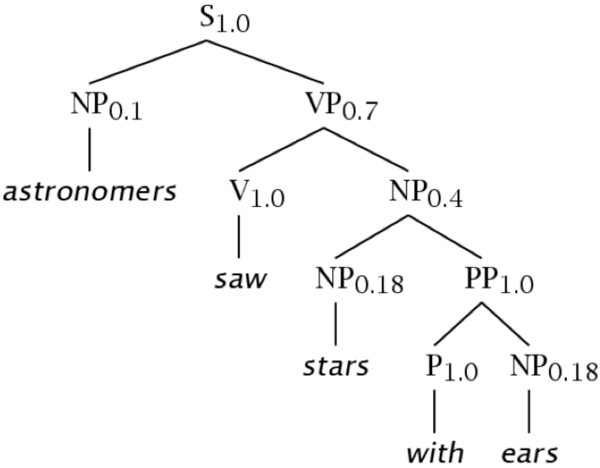
$$\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$$

A Simple PCFG (in CNF)

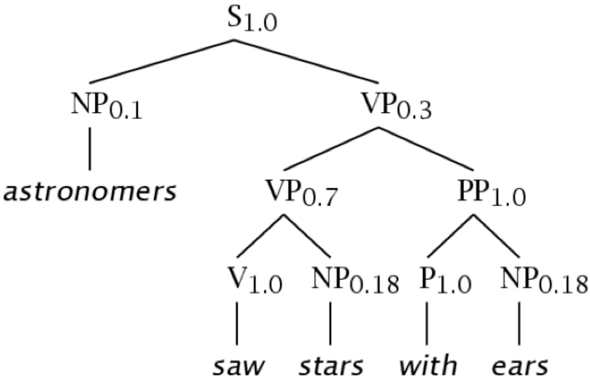
S	$\rightarrow$	NP VP	1.0	NP	$\rightarrow$	NP PP	0.4
VP	$\rightarrow$	V NP	0.7	NP	$\rightarrow$	*astronomers*	0.1
VP	$\rightarrow$	VP PP	0.3	NP	$\rightarrow$	*ears*	0.18
PP	$\rightarrow$	P NP	1.0	NP	$\rightarrow$	*saw*	0.04
P	$\rightarrow$	*with*	1.0	NP	$\rightarrow$	*stars*	0.18
V	$\rightarrow$	*saw*	1.0	NP	$\rightarrow$	*telescope*	0.1

Example Trees

t_1:



t_2:


S1.0
NP0.1
VP0.3
astronomers
VP0.7
PP1.0
V1.0
NP0.18
P1.0
NP0.18
saw
stars
with
ears

Probability of trees and strings

- $P(t)$: The probability of tree is the product of the probabilities of the rules used to generate it
- $P(w_{1n})$: The probability of the string is the sum of the probabilities of the trees which have that string as their yield

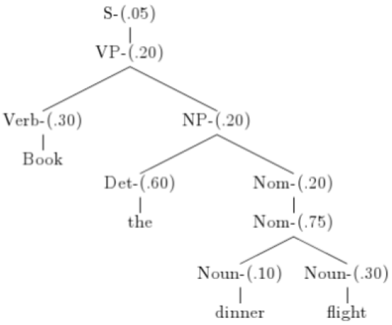
Tree and String probabilities

Tree and String probabilities

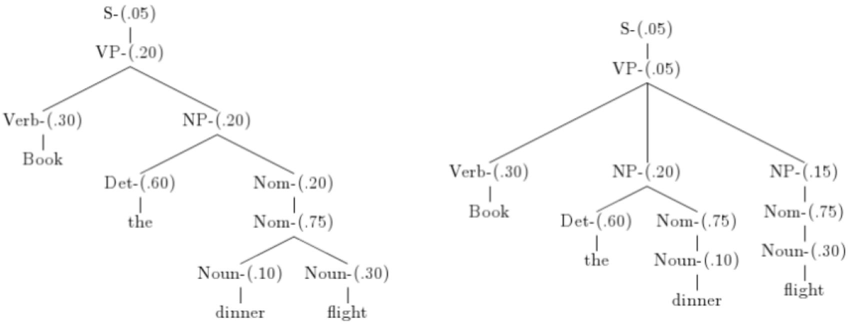
$$
\begin{aligned}
w_{15} &= \textit{astronomers saw stars with ears} \\
P(t_1) &= 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 \\
&\quad * 1.0 * 1.0 * 0.18 \\
&= 0.0009072 \\
P(t_2) &= 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 \\
&\quad * 1.0 * 1.0 * 0.18 \\
&= 0.0006804 \\
P(w_{15}) &= P(t_1) + P(t_2) \\
&= 0.0009072 + 0.0006804 \\
&= 0.0015876
\end{aligned}
$$

"Book the dinner flight"

"Book the dinner flight"



"Book the dinner flight"



Probabilities

- Parse tree 1: $.05 \times .20 \times .30 \times .20 \times .60 \times .20 \times .75 \times .10 \times .30 = 1.62 \times 10^{-6}$
- Parse tree 2: $.05 \times .05 \times .30 \times .20 \times .60 \times .75 \times .10 \times .15 \times .75 \times .30 = 2.28 \times 10^{-7}$

Features of PCFGs

- As the number of possible trees for a given input grows, a PCFG gives some idea of the plausibility of a particular parse

Features of PCFGs

- As the number of possible trees for a given input grows, a PCFG gives some idea of the plausibility of a particular parse
- *But* the probability estimates are based purely on structural factors, and do not factor in lexical co-occurrence. Thus, PCFG does not give a very good idea of the plausibility of the sentence.

Features of PCFGs

- As the number of possible trees for a given input grows, a PCFG gives some idea of the plausibility of a particular parse
- *But* the probability estimates are based purely on structural factors, and do not factor in lexical co-occurrence. Thus, PCFG does not give a very good idea of the plausibility of the sentence.
- Real text tends to have grammatical mistakes. PCFG avoids this problem by ruling out nothing, but by giving implausible sentences a low probability

Features of PCFGs

- As the number of possible trees for a given input grows, a PCFG gives some idea of the plausibility of a particular parse
- *But* the probability estimates are based purely on structural factors, and do not factor in lexical co-occurrence. Thus, PCFG does not give a very good idea of the plausibility of the sentence.
- Real text tends to have grammatical mistakes. PCFG avoids this problem by ruling out nothing, but by giving implausible sentences a low probability
- In practice, a PCFG is a worse language model for English than an n-gram model

Features of PCFGs

- As the number of possible trees for a given input grows, a PCFG gives some idea of the plausibility of a particular parse
- *But* the probability estimates are based purely on structural factors, and do not factor in lexical co-occurrence. Thus, PCFG does not give a very good idea of the plausibility of the sentence.
- Real text tends to have grammatical mistakes. PCFG avoids this problem by ruling out nothing, but by giving implausible sentences a low probability
- In practice, a PCFG is a worse language model for English than an n-gram model
- All else being equal, the probability of a smaller tree is greater than a larger tree

Important Questions?

Let W_{1m} be a sentence, G a grammar, t a parse tree

Important Questions?

Let W_{1m} be a sentence, G a grammar, t a parse tree

- What is the most likely parse of sentence?

$$argmax_t P(t|w_{1m}, G)$$

Important Questions?

Let W_{1m} be a sentence, G a grammar, t a parse tree

- What is the most likely parse of sentence?

$$argmax_t P(t|w_{1m}, G)$$

- What is the probability of a sentence?

$$P(w_{1m}|G)$$

Important Questions?

Let W_{1m} be a sentence, G a grammar, t a parse tree

- What is the most likely parse of sentence?

$$argmax_t P(t|w_{1m}, G)$$

- What is the probability of a sentence?

$$P(w_{1m}|G)$$

- How to learn the rule probabilities in the grammar G?

PCFGs - Inside-outside probabilities

Pawan Goyal

CSE, IIT Kharagpur

Week 5: Lecture 4

How to find the most likely parse?: CKY for PCFG

How to find the most likely parse?: CKY for PCFG

a 1	pilot 2	likes 3	flying 4	planes 5

$S \longrightarrow NP\ VP$	[1.0]
$VP \longrightarrow VBG\ NNS$	[0.1]
$VP \longrightarrow VBZ\ VP$	[0.1]
$VP \longrightarrow VBZ\ NP$	[0.3]
$NP \longrightarrow DT\ NN$	[0.3]
$NP \longrightarrow JJ\ NNS$	[0.4]
$DT \longrightarrow a$	[0.3]
$NN \longrightarrow pilot$	[0.1]
$VBZ \longrightarrow likes$	[0.4]
$VBG \longrightarrow flying$	[0.5]
$JJ \longrightarrow flying$	[0.1]
$NNS \longrightarrow planes$	[.34]

CKY for PCFG

a 1	pilot 2	likes 3	flying 4	planes 5
DT [0.3]	NP [.009]	-	-	S [$1.4688 x 10^{-5}$] S [$6.12 x 10^{-6}$]
	NN [0.1]	-	-	-
		VBZ [0.4]	-	VP [.001632] VP [.00068]
			JJ [0.1] VBG [0.5]	NP [.0136] VP [.017]
				NNS [.34]

$S \longrightarrow NP\ VP$		[1.0]
$VP \longrightarrow VBG\ NNS$		[0.1]
$VP \longrightarrow VBZ\ VP$		[0.1]
$VP \longrightarrow VBZ\ NP$		[0.3]
$NP \longrightarrow DT\ NN$		[0.3]
$NP \longrightarrow JJ\ NNS$		[0.4]
$DT \longrightarrow a$		[0.3]
$NN \longrightarrow pilot$		[0.1]
$VBZ \longrightarrow likes$		[0.4]
$VBG \longrightarrow flying$		[0.5]
$JJ \longrightarrow flying$		[0.1]
$NNS \longrightarrow planes$		[.34]

$0.009 \times 0.00068 \times 1.0 = 6.12 x 10^{-6}$

Probability of a String

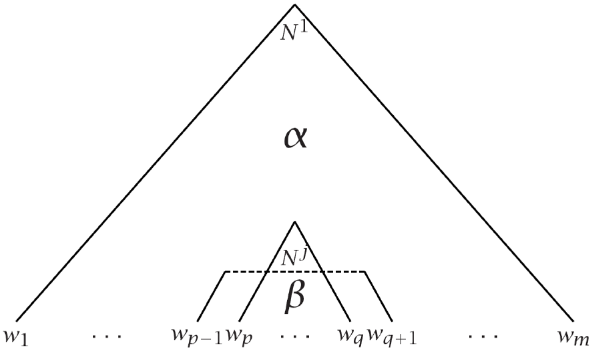
$$P(w_{1m}|G)$$

Probability of a String

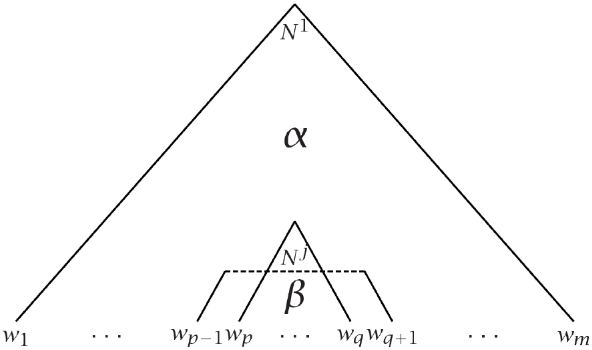
$$P(w_{1m}|G)$$

- In general, simply summing the probabilities of all possible parse trees is not an efficient way to calculate the string probability
- We use *inside algorithm*, a dynamic programming algorithm based on inside probabilities.

Inside and Outside Probabilities

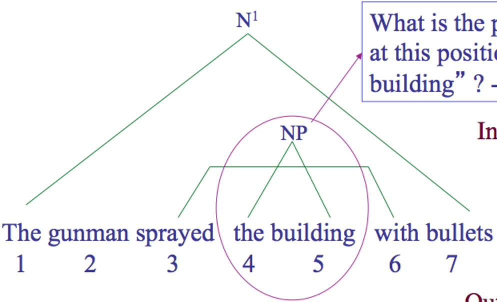


Inside and Outside Probabilities



Outside: $\alpha_j(p,q) = P(w_{1(p-1)}, N^j{}_{pq}, w_{(q+1)m}|G)$
Inside: $\beta_j(p,q) = P(w_{pq}|N^j{}_{pq}, G)$

Inside-outside probabilities



What is the probability that an NP at this position will derive "the building" ? - $\beta_{NP}(4,5)$

Inside Probabilities

Outside Probabilities

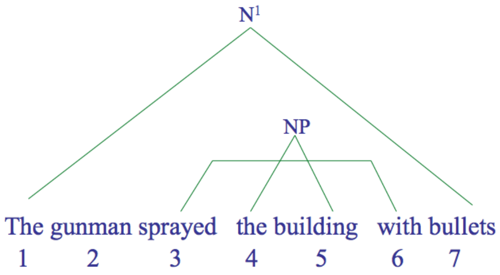
What is the probability of starting from N^1 and deriving "The gunman sprayed", a NP and "with bullets" ? - $\alpha_{NP}(4,5)$

Inside-outside probabilities

$\alpha_{NP}(4,5)$ for "the building"

$= P(\text{The gunman sprayed}, NP_{4,5}, \text{with bullets} \mid G)$

$\beta_{NP}(4,5)$ for "the building" $= P(\text{the building} \mid NP_{4,5}, G)$



Inside Probabilities: Base Step

$$\beta_j(p,q) = P(w_{pq}|N^j{}_{pq}, G)$$

Inside Probabilities: Base Step

$$\beta_j(p,q) = P(w_{pq}|N^j{}_{pq}, G)$$

Base case

$$\beta_j(k,k) = P(w_{kk}|N^j{}_{kk}, G)$$
$$= P(N^j \rightarrow w_k|G)$$

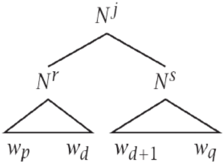
Base case for pre-terminals only

E.g., suppose $N^j = NN$ is being considered and $NN \rightarrow building$ is one of the rules with probability 0.5
$\beta_{NN}(5,5) = P(building|NN_{5,5}, G) = P(NN_{5,5} \rightarrow building|G)$

Inside Probabilities: Induction Step

Assuming Chomsky Normal Form, the first rule must be of the form $N^j \rightarrow N^r N^s$

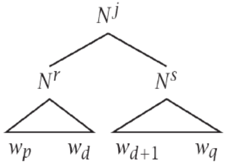
$$\beta_j(p,q) = \sum_{r,s} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)$$



Inside Probabilities: Induction Step

Assuming Chomsky Normal Form, the first rule must be of the form $N^j \rightarrow N^r N^s$

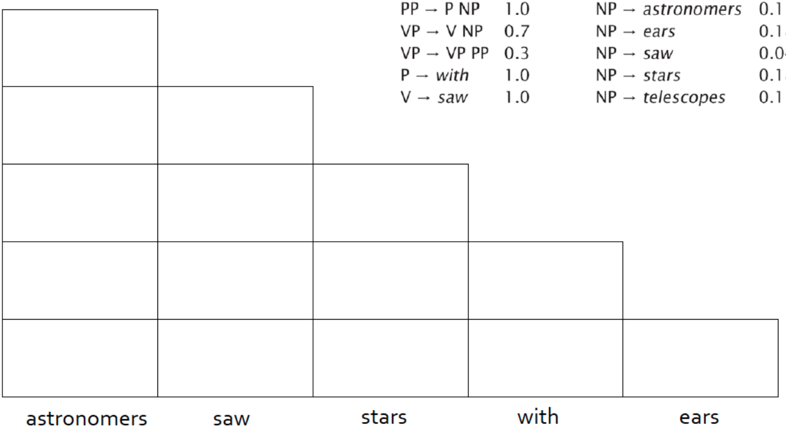
$$\beta_j(p,q) = \sum_{r,s} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)$$



- Consider different splits of the words - indicated by d
  E.g., *the huge building*
- Consider different non-terminals to be used in the rule:
  E.g., $NP \rightarrow DT\ NN$, $NP \rightarrow DT\ NNS$

Calculation of inside probabilities

S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → *astronomers*	0.1
VP → V NP	0.7	NP → *ears*	0.18
VP → VP PP	0.3	NP → *saw*	0.04
P → *with*	1.0	NP → *stars*	0.18
V → *saw*	1.0	NP → *telescopes*	0.1



Calculation of inside probabilities

	1	2	3	4	5
1	$\beta_{NP} = 0.1$		$\beta_S = 0.0126$		$\beta_S = 0.0015876$
2		$\beta_{NP} = 0.04$ $\beta_V = 1.0$	$\beta_{VP} = 0.126$		$\beta_{VP} = 0.015876$
3			$\beta_{NP} = 0.18$		$\beta_{NP} = 0.01296$
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	astronomers	*saw*	*stars*	*with*	*ears*

Outside Probabilities

Compute top-down (after inside probabilities)

Outside Probabilities

Compute top-down (after inside probabilities)

Base Case

Outside Probabilities

Compute top-down (after inside probabilities)

Base Case

$$\alpha_1(1,m) = 1$$

$$\alpha_j(1,m) = 0, j \neq 1$$

Outside Probabilities

Compute top-down (after inside probabilities)

Base Case

$$\alpha_1(1,m) = 1$$

$$\alpha_j(1,m) = 0, j \neq 1$$

Induction

Outside Probabilities

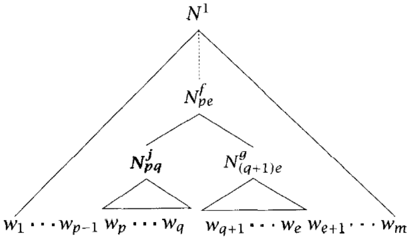
Compute top-down (after inside probabilities)

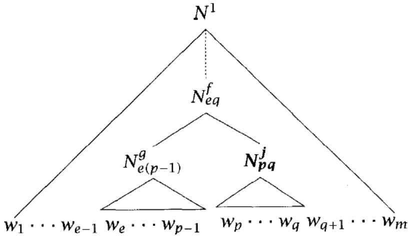
Base Case

$$\alpha_1(1,m) = 1$$

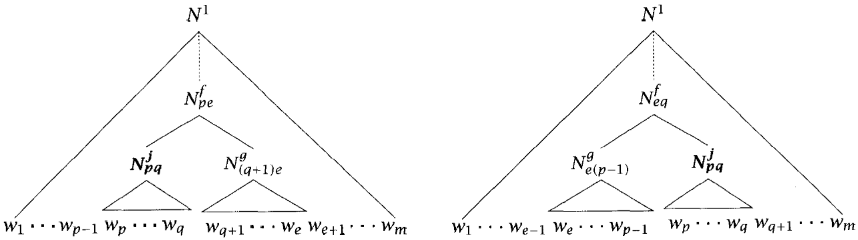
$$\alpha_j(1,m) = 0, j \neq 1$$

Induction





Outside Probabilities: Induction



$$\alpha_j(p,q) = \sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p,e) P(N^f \rightarrow N^j N^g) \beta_g(q+1,e) + \sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e,q) P(N^f \rightarrow N^g N^j) \beta_g(e,p-1)$$

Product of inside-outside probabilities

$$\alpha_j(p,q)\beta_j(p,q) = P(w_{1(p-1)}, N^j{}_{pq}, w_{(q+1)m}|G)P(w_{pq}|N^j{}_{pq}, G) = P(w_{1m}, N^j{}_{pq}|G)$$

Product of inside-outside probabilities

$$\alpha_j(p,q)\beta_j(p,q) = P(w_{1(p-1)}, N^j{}_{pq}, w_{(q+1)m}|G)P(w_{pq}|N^j{}_{pq}, G) = P(w_{1m}, N^j{}_{pq}|G)$$

The probability of the sentence and that there is some consistent spanning from word p to q is given by:

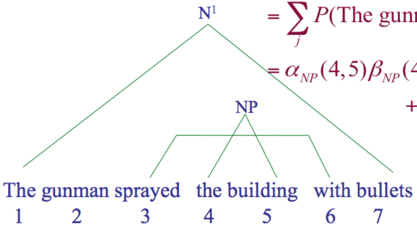
$$P(w_{1m}, N_{pq}|G) = \sum \alpha_j(p,q)\beta_j(p,q) = P(N_1 \rightarrow w_{1m}, N_{pq} \rightarrow w_{pq}|G)$$

Product of inside-outside probabilities

$$\alpha_j(p,q)\beta_j(p,q) = P(w_{1(p-1)}, N^j{}_{pq}, w_{(q+1)m}|G)P(w_{pq}|N^j{}_{pq}, G) = P(w_{1m}, N^j{}_{pq}|G)$$

The probability of the sentence and that there is some consistent spanning from word p to q is given by:

$$P(w_{1m}, N_{pq}|G) = \sum \alpha_j(p,q)\beta_j(p,q) = P(N_1 \rightarrow w_{1m}, N_{pq} \rightarrow w_{pq}|G)$$



$$P(\text{The gunman....bullets}, N_{4,5} \mid G)$$
$$= \sum_j P(\text{The gunman...bullets} \mid N^j_{4,5}, G)$$
$$= \alpha_{NP}(4,5)\beta_{NP}(4,5)$$
$$+ \alpha_{VP}(4,5)\beta_{VP}(4,5) + \ldots$$

Inside-outside probabilities

Pawan Goyal

CSE, IIT Kharagpur

Week 5: Lecture 5

How to get the rule probabilities

Parsed Training Data

You can count!

$$\hat{P}(N^j \to \delta) = \frac{C(N^j \to \delta)}{\sum_\gamma C(N^j \to \gamma)}$$

How to get the rule probabilities

Parsed Training Data

You can count!

$$\hat{P}(N^j \to \delta) = \frac{C(N^j \to \delta)}{\sum_\gamma C(N^j \to \gamma)}$$

But what if the training data is not available?

i.e. gold standard parse is not known.

How to get the rule probabilities

Parsed Training Data

You can count!

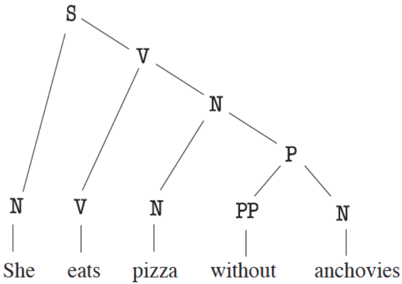
$$\hat{P}(N^j \to \delta) = \frac{C(N^j \to \delta)}{\sum_\gamma C(N^j \to \gamma)}$$

But what if the training data is not available?

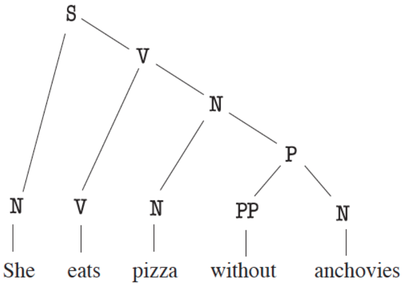
i.e. gold standard parse is not known.

- Underlying CFG is known and we are given a set of sentences
- For each sentence, we can find out all the possible parses
- *Maximize the likelihood of the sentences in the data under the PCFG constraints*

Example data



Example data



Rules of the form $A \rightarrow BC$

S → N V

V → V N

N → N P

P → PP N .

Rules of the form $A \rightarrow w$

N → She

V → eats

N → pizza

PP → without

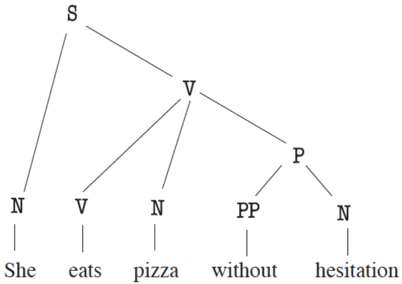
N → anchovies.

Example data

Is any other parse possible for *She eats pizza without anchovies* syntactically?

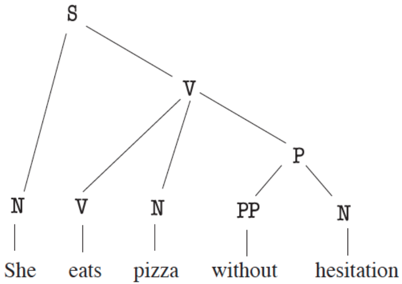
Example data

Is any other parse possible for *She eats pizza without anchovies* syntactically?
Consider *She eats pizza without hesitation*



Example data

Is any other parse possible for *She eats pizza without anchovies* syntactically?
Consider *She eats pizza without hesitation*



New Context-free rules:

$$V \rightarrow V\ N\ P$$
$$N \rightarrow \text{hesitation}\ .$$

Estimating the model parameters

We need to find probabilities such as

- $\phi(S \rightarrow N\ V)$
- $\phi(N \rightarrow pizza)$

Estimating the model parameters

We need to find probabilities such as

- $\phi(S \rightarrow N\ V)$
- $\phi(N \rightarrow pizza)$

Requirements

For each non-terminal A, the derivation probabilities sum up to 1

$$\sum_{\alpha} \phi(A \rightarrow \alpha) = 1$$

Estimating the model parameters

We need to find probabilities such as

- $\phi(S \rightarrow N\ V)$
- $\phi(N \rightarrow pizza)$

Requirements

For each non-terminal A, the derivation probabilities sum up to 1

$$\sum_{\alpha} \phi(A \rightarrow \alpha) = 1$$

For the example grammar:

$$\begin{aligned}
\phi(\mathtt{N} \rightarrow \mathtt{N\ P}) + \phi(\mathtt{N} \rightarrow \text{pizza}) + \phi(\mathtt{N} \rightarrow \text{anchovies}) &\quad + \\
+\ \phi(\mathtt{N} \rightarrow \text{hesitation}) + \phi(\mathtt{N} \rightarrow \text{She}) &= 1 \\
\phi(\mathtt{V} \rightarrow \mathtt{V\ N}) + \phi(\mathtt{V} \rightarrow \mathtt{V\ N\ P}) + \phi(\mathtt{V} \rightarrow \text{eats}) &= 1
\end{aligned}$$

$$\begin{aligned}
\phi(\mathtt{S} \rightarrow \mathtt{N\ V}) &= 1 \\
\phi(\mathtt{P} \rightarrow \mathtt{PP\ N}) &= 1 \\
\phi(\mathtt{PP} \rightarrow \text{without}) &= 1
\end{aligned}$$

Likelihood computation

W_1 = "She eats pizza without anchovies"

W_2 = "She eats pizza without hesitation".

Likelihood computation

$W_1 =$ "She eats pizza without anchovies"

$W_2 =$ "She eats pizza without hesitation".

$$\begin{aligned} P_\phi(W_1, T_1) &= \phi(\mathtt{S} \to \mathtt{N\ V})\, \phi(\mathtt{V} \to \mathtt{V\ N})\, \phi(\mathtt{N} \to \mathtt{N\ P}) \times \\ &\times \phi(\mathtt{P} \to \mathtt{PP\ N})\, \phi(\mathtt{N} \to \text{She})\, \phi(\mathtt{V} \to \text{eats}) \times \\ &\times \phi(\mathtt{N} \to \text{pizza})\, \phi(\mathtt{PP} \to \text{without})\, \phi(\mathtt{N} \to \text{anchovies}) \end{aligned}$$

$$\begin{aligned} P_\phi(W_2, T_1) &= \phi(\mathtt{S} \to \mathtt{N\ V})\, \phi(\mathtt{V} \to \mathtt{V\ N\ P})\, \phi(\mathtt{P} \to \mathtt{P\ PP}) \times \\ &\times \phi(\mathtt{N} \to \text{She})\, \phi(\mathtt{V} \to \text{eats})\, \phi(\mathtt{N} \to \text{pizza}) \times \\ &\times \phi(\mathtt{PP} \to \text{without})\, \phi(\mathtt{N} \to \text{hesitation}) \end{aligned}$$

Likelihood computation

$$
\begin{aligned}
P_\phi(W_1, T_2) &= \phi(\mathtt{S} \rightarrow \mathtt{N\ V})\,\phi(\mathtt{V} \rightarrow \mathtt{V\ N\ P})\,\phi(\mathtt{P} \rightarrow \mathtt{P\ PP}) \times \\
&\times \phi(\mathtt{N} \rightarrow \text{She})\,\phi(\mathtt{V} \rightarrow \text{eats})\,\phi(\mathtt{N} \rightarrow \text{pizza}) \times \\
&\times \phi(\mathtt{PP} \rightarrow \text{without})\,\phi(\mathtt{N} \rightarrow \text{anchovies})
\end{aligned}
$$

$$
\begin{aligned}
P_\phi(W_2, T_1) &= \phi(\mathtt{S} \rightarrow \mathtt{N\ V})\,\phi(\mathtt{V} \rightarrow \mathtt{V\ N})\,\phi(\mathtt{N} \rightarrow \mathtt{N\ P}) \times \\
&\times \phi(\mathtt{P} \rightarrow \mathtt{PP\ N})\,\phi(\mathtt{N} \rightarrow \text{She})\,\phi(\mathtt{V} \rightarrow \text{eats}) \times \\
&\times \phi(\mathtt{N} \rightarrow \text{pizza})\,\phi(\mathtt{PP} \rightarrow \text{without})\,\phi(\mathtt{N} \rightarrow \text{hesitation})
\end{aligned}
$$

Likelihood computation

$$\begin{aligned} P_\phi(W_1, T_2) &= \phi(\mathtt{S} \to \mathtt{N\ V})\,\phi(\mathtt{V} \to \mathtt{V\ N\ P})\,\phi(\mathtt{P} \to \mathtt{P\ PP}) \times \\ &\times \phi(\mathtt{N} \to \text{She})\,\phi(\mathtt{V} \to \text{eats})\,\phi(\mathtt{N} \to \text{pizza}) \times \\ &\times \phi(\mathtt{PP} \to \text{without})\,\phi(\mathtt{N} \to \text{anchovies}) \end{aligned}$$

$$\begin{aligned} P_\phi(W_2, T_1) &= \phi(\mathtt{S} \to \mathtt{N\ V})\,\phi(\mathtt{V} \to \mathtt{V\ N})\,\phi(\mathtt{N} \to \mathtt{N\ P}) \times \\ &\times \phi(\mathtt{P} \to \mathtt{PP\ N})\,\phi(\mathtt{N} \to \text{She})\,\phi(\mathtt{V} \to \text{eats}) \times \\ &\times \phi(\mathtt{N} \to \text{pizza})\,\phi(\mathtt{PP} \to \text{without})\,\phi(\mathtt{N} \to \text{hesitation}) \end{aligned}$$

Likelihood of the corpus

Probability of a sentence W : $P_\phi(W) = \sum_T P_\phi(W,T)$

Likelihood computation

$$\begin{aligned} P_\phi(W_1, T_2) &= \phi(\mathtt{S} \rightarrow \mathtt{N\ V})\, \phi(\mathtt{V} \rightarrow \mathtt{V\ N\ P})\, \phi(\mathtt{P} \rightarrow \mathtt{P\ PP}) \times \\ &\times \quad \phi(\mathtt{N} \rightarrow \text{She})\, \phi(\mathtt{V} \rightarrow \text{eats})\, \phi(\mathtt{N} \rightarrow \text{pizza}) \times \\ &\times \quad \phi(\mathtt{PP} \rightarrow \text{without})\, \phi(\mathtt{N} \rightarrow \text{anchovies}) \end{aligned}$$

$$\begin{aligned} P_\phi(W_2, T_1) &= \phi(\mathtt{S} \rightarrow \mathtt{N\ V})\, \phi(\mathtt{V} \rightarrow \mathtt{V\ N})\, \phi(\mathtt{N} \rightarrow \mathtt{N\ P}) \times \\ &\times \quad \phi(\mathtt{P} \rightarrow \mathtt{PP\ N})\, \phi(\mathtt{N} \rightarrow \text{She})\, \phi(\mathtt{V} \rightarrow \text{eats}) \times \\ &\times \quad \phi(\mathtt{N} \rightarrow \text{pizza})\, \phi(\mathtt{PP} \rightarrow \text{without})\, \phi(\mathtt{N} \rightarrow \text{hesitation}) \end{aligned}$$

Likelihood of the corpus

Probability of a sentence W : $P_\phi(W) = \sum_T P_\phi(W,T)$

If the training data comprises of sentences $W_1, W_2, \ldots, W_N$, then the likelihood is

$$L(\phi) = P_\phi(W_1) P_\phi(W_2) \cdots P_\phi(W_N)$$

Likelihood maximization

Approach

Starting at some initial parameters ϕ, re-estimate to obtain new parameters ϕ' for which $L(\phi') \geq L(\phi)$. *Repeat until convergence*

Parameter Estimation

Given some rule probabilities ϕ and training corpus $W_1, W_2 \dots W_n$, the new parameters are obtained as:

$$\phi'(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C}) = \frac{count(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C})}{\sum_\alpha count(\mathtt{A} \rightarrow \alpha)}$$

$$\phi'(\mathtt{A} \rightarrow \mathrm{w}) = \frac{count(\mathtt{A} \rightarrow \mathrm{w})}{\sum_\alpha count(\mathtt{A} \rightarrow \alpha)}$$

What is $count(.)$?

Parameter Estimation

Given some rule probabilities ϕ and training corpus $W_1, W_2 \ldots W_n$, the new parameters are obtained as:

$$\phi'(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C}) = \frac{count(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C})}{\sum_\alpha count(\mathtt{A} \rightarrow \alpha)}$$

$$\phi'(\mathtt{A} \rightarrow \mathrm{w}) = \frac{count(\mathtt{A} \rightarrow \mathrm{w})}{\sum_\alpha count(\mathtt{A} \rightarrow \alpha)}$$

What is $count(.)$?

$$count(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C}) = \sum_{i=1}^{N} c_\phi(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C}, W_i)$$

$$count(\mathtt{A} \rightarrow \mathrm{w}) = \sum_{i=1}^{N} c_\phi(\mathtt{A} \rightarrow \mathrm{w}, W_i)$$

Parameter Estimation

Given some rule probabilities ϕ and training corpus $W_1, W_2 \ldots W_n$, the new parameters are obtained as:

$$\phi'(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C}) = \frac{count(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C})}{\sum_\alpha count(\mathtt{A} \rightarrow \alpha)}$$

$$\phi'(\mathtt{A} \rightarrow \mathrm{w}) = \frac{count(\mathtt{A} \rightarrow \mathrm{w})}{\sum_\alpha count(\mathtt{A} \rightarrow \alpha)}$$

What is $count(.)$?

$$count(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C}) = \sum_{i=1}^{N} c_\phi(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C}, W_i)$$

$$count(\mathtt{A} \rightarrow \mathrm{w}) = \sum_{i=1}^{N} c_\phi(\mathtt{A} \rightarrow \mathrm{w}, W_i)$$

$c_\phi(A \rightarrow \alpha, W_i)$ is the expected number of times $(A \rightarrow \alpha)$ is used in generating the sentence W_i, when the rule probabilities are given by ϕ

Computing Expected counts

Inside probabilities

The nonterminal A derives the string of words $w_i, \ldots w_j$ in the sentence :
$\beta_{ij}(A) = P_\phi(A \Rightarrow^* w_i \ldots w_j)$

Computing Expected counts

Inside probabilities

The nonterminal A derives the string of words $w_i, \ldots w_j$ in the sentence :
$\beta_{ij}(A) = P_\phi(A \Rightarrow^* w_i \ldots w_j)$

Outside probabilities

Beginning with the start symbol S we can derive the string
$w_1 \ldots w_{i-1} A w_{j+1} \ldots w_n : \alpha_{ij}(A) = P_\phi(S \Rightarrow^* w_1 \ldots w_{i-1} A w_{j+1} \ldots w_n)$

Computing Expected counts

Inside probabilities

The nonterminal A derives the string of words $w_i, \dots w_j$ in the sentence :
$\beta_{ij}(A) = P_\phi(A \Rightarrow^* w_i \dots w_j)$

Outside probabilities

Beginning with the start symbol S we can derive the string
$w_1 \dots w_{i-1} A w_{j+1} \dots w_n$: $\alpha_{ij}(A) = P_\phi(S \Rightarrow^* w_1 \dots w_{i-1} A w_{j+1} \dots w_n)$

Expected count

$$c_\phi(A \rightarrow BC, W) = \frac{\phi(A \rightarrow BC)}{P_\phi(W)} \sum_{1 \le i \le j \le k \le n} \alpha_{ik}(A) \beta_{ij}(B) \beta_{j+1,k}(C)$$

$$c_\phi(A \rightarrow w, W) = \frac{\phi(A \rightarrow w)}{P_\phi(W)} \sum_{1 \le i \le n} \alpha_{ii}(A)$$

And how to compute inside-outside probabilities

Inductively, as discussed earlier

$$\beta_{ii}(A) = \phi(A \rightarrow w_i)$$

$$\alpha_{1n}(S) = 1$$