CS 489: Introduction to Natural Language Processing

Lecture 11: Syntax and Context-Free Grammars

Instructor: Freda Shi

October 27, 2026



UNIVERSITY OF WATERLOO

Outline

We have trained language models on token sequences. Today we describe the **structure** those sequences can have.

1. Constituents: how words group into phrases.
2. CFGs and PCFGs: generating trees and assigning probabilities.
3. CKY and inside: the best tree versus all trees.
4. Dependencies: relations between the words themselves.

Why Study Syntax?

The man walked to the park.

Words form **nested phrases**, not just a flat sequence:

[the man] [walked [to [the park]]].

noun phrase — noun phrase

Why Study Syntax?

The man walked to the park.

Words form **nested phrases**, not just a flat sequence:

[the man] [walked [to [the park]]].

noun phrase — noun phrase

Structure helps us describe **who did what** and **which words belong together**.

- A sentence can have more than one structure, with different meanings.
- We need an explicit representation before we can ask whether an LM learns it.

What Is a Constituent?

A **constituent** is a word or group of words that functions as a unit in a syntactic structure.

[The man] [walked [to [the park]]].

- **the man**, **the park**: noun phrases (NP).
- **to the park**: a prepositional phrase (PP).
- **walked to the park**: a verb phrase (VP).

What Is a Constituent?

A **constituent** is a word or group of words that functions as a unit in a syntactic structure.

[The man] [walked [to [the park]]].

- **the man**, **the park**: noun phrases (NP).
- **to the park**: a prepositional phrase (PP).
- **walked to the park**: a verb phrase (VP).

How can we identify these groups? Try replacing, moving, coordinating, or deleting a candidate span. These tests provide evidence, not a perfect decision procedure.

Constituency Test: Coordination

The man walked to the park.

Join a candidate phrase to another expression using *and* or *or*.

Constituency Test: Coordination

The man walked to the park.

Join a candidate phrase to another expression using *and* or *or*.

- [**The man**] and [**the woman**] walked to the park.
- The man [**walked to the park**] and [**sat on a bench**].

Constituency Test: Coordination

The man walked to the park.

Join a candidate phrase to another expression using *and* or *or*.

- [**The man**] and [**the woman**] walked to the park.
- The man [**walked to the park**] and [**sat on a bench**].

These examples support an NP analysis and a VP analysis.

Constituency Test: Movement

A phrase can sometimes move as a unit while leaving the rest of the sentence intact.

The man walked [**to the park**].

Constituency Test: Movement

A phrase can sometimes move as a unit while leaving the rest of the sentence intact.

The man walked [**to the park**].

[**To the park**], the man walked; to the station, he drove.

The contrast makes the fronted phrase more natural.

Constituency Test: Movement

A phrase can sometimes move as a unit while leaving the rest of the sentence intact.

The man walked [**to the park**].

[**To the park**], the man walked; to the station, he drove.

The contrast makes the fronted phrase more natural.

A failed test does not prove that a span is not a constituent. Movement also depends on the construction and discourse context.

Constituency Tests: Substitution and Deletion

Substitution: replace a phrase with an expression that fills the same role.

- [The man] walked to the park. → He walked to the park.
- The man [walked to the park]. → The man did so.

Constituency Tests: Substitution and Deletion

Substitution: replace a phrase with an expression that fills the same role.

- [The man] walked to the park. → He walked to the park.
- The man [walked to the park]. → The man did so.

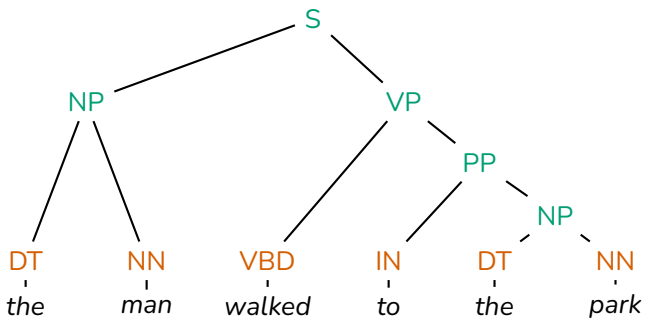
Deletion: an optional phrase can be omitted.

The man walked ~~to the park~~.

Not every constituent is optional. Use converging evidence from multiple tests.

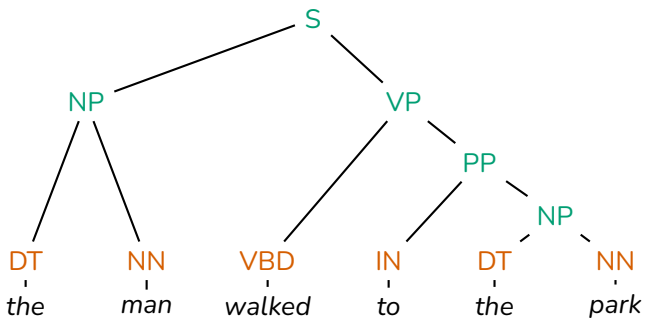
From Brackets to a Parse Tree

Bracketing: *((the man) (walked (to (the park))))*



From Brackets to a Parse Tree

Bracketing: *((the man) (walked (to (the park))))*

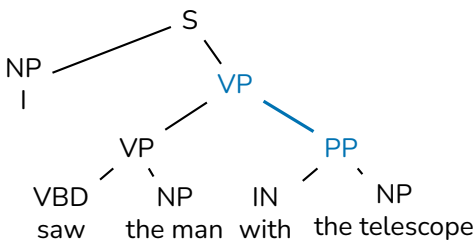


Phrasal categories and POS tags are nonterminals. A POS tag directly above a word is a **preterminal**; the words are terminals.

One Sentence, Two Structures

I saw the man with the telescope.

VP attachment

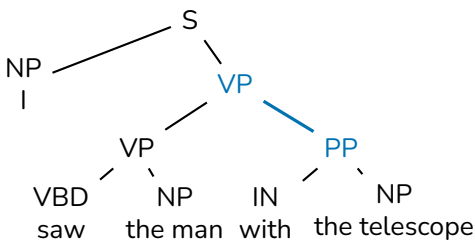


I used the telescope to see.

One Sentence, Two Structures

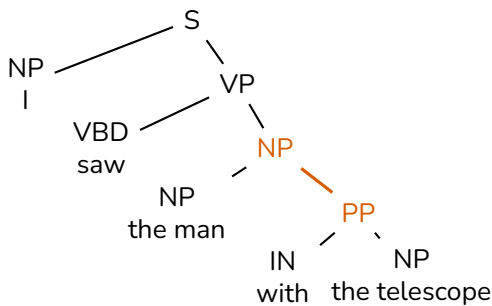
I saw the man with the telescope.

VP attachment



I used the telescope to see.

NP attachment

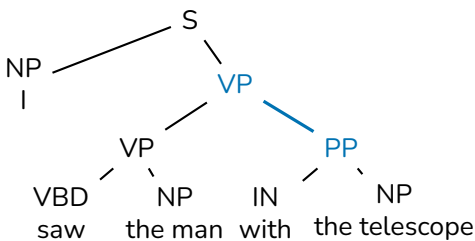


The man had the telescope.

One Sentence, Two Structures

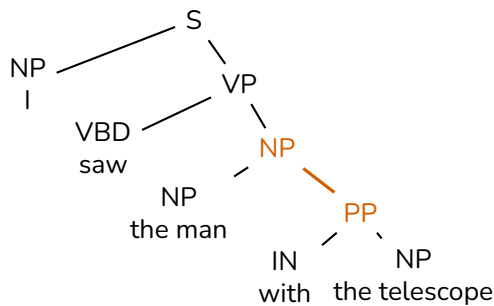
I saw the man with the telescope.

VP attachment



I used the telescope to see.

NP attachment



The man had the telescope.

Syntactic ambiguity: the same words have different parse trees.

Learning from Annotated Trees

A **treebank** is a corpus annotated with syntactic trees. The **Penn Treebank** includes the widely used Wall Street Journal constituency dataset.

POS tags in our examples

DT	Determiner
NN	Singular noun
VBD	Past-tense verb
IN	Preposition / subordinator

Phrasal categories

S	Sentence or clause
NP	Noun phrase
VP	Verb phrase
PP	Prepositional phrase

Learning from Annotated Trees

A **treebank** is a corpus annotated with syntactic trees. The **Penn Treebank** includes the widely used Wall Street Journal constituency dataset.

POS tags in our examples

DT	Determiner
NN	Singular noun
VBD	Past-tense verb
IN	Preposition / subordinator

Phrasal categories

S	Sentence or clause
NP	Noun phrase
VP	Verb phrase
PP	Prepositional phrase

We can extract grammar rules from these trees and estimate how often each rule is used.

[Marcus, Santorini, and Marcinkiewicz, 1993.]

Context-Free Grammar: Definition

A **context-free grammar** is a tuple $G = (N, \Sigma, R, S)$:

- N: nonterminal symbols, such as S, NP, VP, and NN.
- Σ: terminal symbols (words), disjoint from N.
- R: production rules $A \rightarrow \alpha$, with a single $A \in N$ on the left.
- $S \in N$: the start symbol.

Here we use finite grammars without empty productions: $\alpha \in (N \cup \Sigma)^+$.

Context-Free Grammar: Definition

A **context-free grammar** is a tuple $G = (N, \Sigma, R, S)$:

- N: nonterminal symbols, such as S, NP, VP, and NN.
- Σ: terminal symbols (words), disjoint from N.
- R: production rules $A \to \alpha$, with a single $A \in N$ on the left.
- $S \in N$: the start symbol.

Here we use finite grammars without empty productions: $\alpha \in (N \cup \Sigma)^+$.

Structural rules

$$\begin{aligned} \text{S} &\to \text{NP VP} \\ \text{NP} &\to \text{DT NN} \\ \text{VP} &\to \text{VBD PP} \\ \text{PP} &\to \text{IN NP} \end{aligned}$$

Lexical rules

$$\begin{aligned} \text{DT} &\to \text{the} \\ \text{NN} &\to \text{man} \mid \text{park} \\ \text{VBD} &\to \text{walked} \\ \text{IN} &\to \text{to} \end{aligned}$$

Deriving a Sentence

A **derivation** applies rules until only terminal symbols remain.

$$
\begin{aligned}
S &\Rightarrow NP\ VP && (S \rightarrow NP\ VP) \\
&\Rightarrow DT\ NN\ VP && (NP \rightarrow DT\ NN)
\end{aligned}
$$

$\Rightarrow^*$ groups several rule applications. Each rule application supplies the children of one node in the parse tree.

Deriving a Sentence

A **derivation** applies rules until only terminal symbols remain.

$$
\begin{aligned}
S &\Rightarrow NP\ VP && (S \to NP\ VP)\\
&\Rightarrow DT\ NN\ VP && (NP \to DT\ NN)\\
&\Rightarrow^{*} \text{the man } VP && (DT \to \text{the},\ NN \to \text{man})
\end{aligned}
$$

$\Rightarrow^{*}$ groups several rule applications. Each rule application supplies the children of one node in the parse tree.

Deriving a Sentence

A **derivation** applies rules until only terminal symbols remain.

$$
\begin{aligned}
S &\Rightarrow NP\ VP && (S \rightarrow NP\ VP) \\
&\Rightarrow DT\ NN\ VP && (NP \rightarrow DT\ NN) \\
&\Rightarrow^{*} \text{the man } VP && (DT \rightarrow \text{the},\ NN \rightarrow \text{man}) \\
&\Rightarrow \text{the man } VBD\ PP && (VP \rightarrow VBD\ PP) \\
&\Rightarrow^{*} \text{the man walked } IN\ NP && (VBD \rightarrow \text{walked},\ PP \rightarrow IN\ NP)
\end{aligned}
$$

$\Rightarrow^{*}$ groups several rule applications. Each rule application supplies the children of one node in the parse tree.

Deriving a Sentence

A **derivation** applies rules until only terminal symbols remain.

$$
\begin{aligned}
S &\Rightarrow \text{NP VP} && (S \to \text{NP VP}) \\
&\Rightarrow \text{DT NN VP} && (\text{NP} \to \text{DT NN}) \\
&\Rightarrow^* \text{the man VP} && (\text{DT} \to \text{the},\ \text{NN} \to \text{man}) \\
&\Rightarrow \text{the man VBD PP} && (\text{VP} \to \text{VBD PP}) \\
&\Rightarrow^* \text{the man walked IN NP} && (\text{VBD} \to \text{walked},\ \text{PP} \to \text{IN NP}) \\
&\Rightarrow^* \text{the man walked to DT NN} && (\text{IN} \to \text{to},\ \text{NP} \to \text{DT NN}) \\
&\Rightarrow^* \text{the man walked to the park}
\end{aligned}
$$

$\Rightarrow^*$ groups several rule applications. Each rule application supplies the children of one node in the parse tree.

Why “Context-Free”?

The rule

$$\text{NP} \rightarrow \text{DT NN}$$

can expand an NP **without inspecting its surroundings**.

- Subject: [**the man**] walked to the park.
- Inside a PP: the man walked to [**the park**].

Why “Context-Free”?

The rule

$$\text{NP} \rightarrow \text{DT NN}$$

can expand an NP **without inspecting its surroundings**.

- Subject: [**the man**] walked to the park.
- Inside a PP: the man walked to [**the park**].

The nonterminal label controls which rules are available. More detailed labels can encode distinctions such as singular and plural NPs to help enforce agreement.

Recursive Rules Build Larger Phrases

A rule can expand an NP into a structure containing another NP. For example, add subject relative clauses:

$$\text{NP} \rightarrow \text{DT NN} \mid \text{NP RC} \qquad \text{RC} \rightarrow \text{that VP}$$
$$\text{VP} \rightarrow \text{VBD NP} \qquad \text{DT} \rightarrow \text{the}$$
$$\text{NN} \rightarrow \text{cat} \mid \text{mouse} \mid \text{cheese} \qquad \text{VBD} \rightarrow \text{chased} \mid \text{ate}$$

RC denotes a relative clause; *that* is a terminal in this toy grammar.

Recursive Rules Build Larger Phrases

A rule can expand an NP into a structure containing another NP. For example, add subject relative clauses:

$$\begin{array}{ll} \text{NP} \rightarrow \text{DT NN} \mid \text{NP RC} & \text{RC} \rightarrow \text{that VP} \\ \text{VP} \rightarrow \text{VBD NP} & \text{DT} \rightarrow \text{the} \\ \text{NN} \rightarrow \text{cat} \mid \text{mouse} \mid \text{cheese} & \text{VBD} \rightarrow \text{chased} \mid \text{ate} \end{array}$$

RC denotes a relative clause; *that* is a terminal in this toy grammar.

Starting from NP, these rules generate:

- the cat
- the cat [that chased the mouse]
- the cat [that chased [the mouse [that ate the cheese]]]

The grammar has finitely many rules, but puts no fixed bound on the depth of the tree.

From CFGs to Probabilistic CFGs

A CFG specifies possible trees under a grammar. It does not tell us which tree to prefer for an ambiguous sentence.

From CFGs to Probabilistic CFGs

A CFG specifies possible trees under a grammar. It does not tell us which tree to prefer for an ambiguous sentence.

A **probabilistic CFG (PCFG)** assigns a probability to each rule:

$$p(A \to \alpha) \geq 0, \qquad \sum_{\alpha:\,(A \to \alpha) \in R} p(A \to \alpha) = 1.$$

Normalize separately for each left-hand-side nonterminal A.

Example: expanding NP might choose a pronoun, a determiner–noun phrase, or an NP with an attached PP.

A Toy PCFG for Attachment Ambiguity

Sentence: *I saw the man with the telescope.*
We omit punctuation and use the following complete grammar throughout the probability examples.

Rule			p
S	$\rightarrow$	NP VP	1.0
NP	$\rightarrow$	I	0.2
NP	$\rightarrow$	DT NN	0.4
NP	$\rightarrow$	NP PP	0.4
VP	$\rightarrow$	VBD NP	0.5
VP	$\rightarrow$	VP PP	0.5
PP	$\rightarrow$	IN NP	1.0

Rule			p
DT	$\rightarrow$	the	1.0
NN	$\rightarrow$	man	0.5
NN	$\rightarrow$	telescope	0.5
VBD	$\rightarrow$	saw	1.0
IN	$\rightarrow$	with	1.0

A Toy PCFG for Attachment Ambiguity

Sentence: *I saw the man with the telescope.*
We omit punctuation and use the following complete grammar throughout the probability examples.

Rule			p
S	$\rightarrow$	NP VP	1.0
NP	$\rightarrow$	I	0.2
NP	$\rightarrow$	DT NN	0.4
NP	$\rightarrow$	NP PP	0.4
VP	$\rightarrow$	VBD NP	0.5
VP	$\rightarrow$	VP PP	0.5
PP	$\rightarrow$	IN NP	1.0

Rule			p
DT	$\rightarrow$	the	1.0
NN	$\rightarrow$	man	0.5
NN	$\rightarrow$	telescope	0.5
VBD	$\rightarrow$	saw	1.0
IN	$\rightarrow$	with	1.0

NP: $0.2 + 0.4 + 0.4 = 1$; VP: $0.5 + 0.5 = 1$; NN: $0.5 + 0.5 = 1$.

Generating a Tree and Its Yield

1. Begin with S.
2. At each nonterminal A, sample a rule from $p(A \to \alpha)$.
3. Expand its children until only terminals remain.

The words at the leaves, read left to right, are the tree's **yield**.

Generating a Tree and Its Yield

1. Begin with S.
2. At each nonterminal A, sample a rule from $p(A \to \alpha)$.
3. Expand its children until only terminals remain.

The words at the leaves, read left to right, are the tree's **yield**.

For a tree T whose yield is $\mathbf{x}$,

$$P(\mathbf{x}, T) = \prod_{v \in \text{internal}(T)} p(r_v).$$

Multiply once for **each rule application**, including repeated uses of the same rule.

Generating a Tree and Its Yield

1. Begin with S.
2. At each nonterminal A, sample a rule from $p(A \to \alpha)$.
3. Expand its children until only terminals remain.

The words at the leaves, read left to right, are the tree's **yield**.

For a tree T whose yield is $\mathbf{x}$,

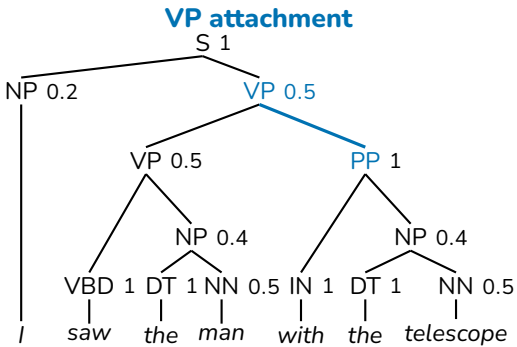
$$P(\mathbf{x}, T) = \prod_{v \in \text{internal}(T)} p(r_v).$$

Multiply once for **each rule application**, including repeated uses of the same rule.

Assume generation terminates with probability 1, as it does for our toy grammar. Rule-by-rule normalization alone does not guarantee termination for every recursive PCFG.

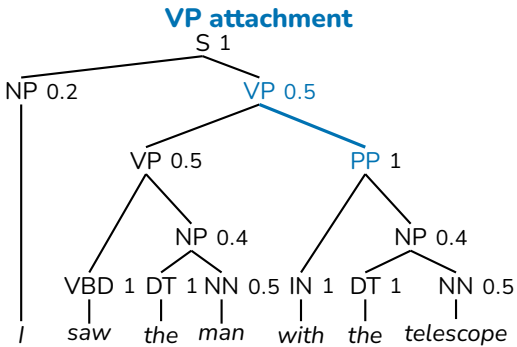
Two Trees, Two Probabilities

Multiply the rule probabilities beside the parents; factors of 1 are omitted below.



Two Trees, Two Probabilities

Multiply the rule probabilities beside the parents; factors of 1 are omitted below.

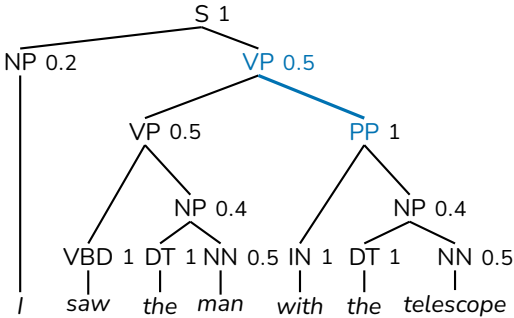


$$P(\mathbf{x}, T_{\text{VP}})$$
$$= 0.2 \times 0.4^2 \times 0.5^2 \times 0.5 \times 0.5$$
$$= \boxed{0.002}.$$

Two Trees, Two Probabilities

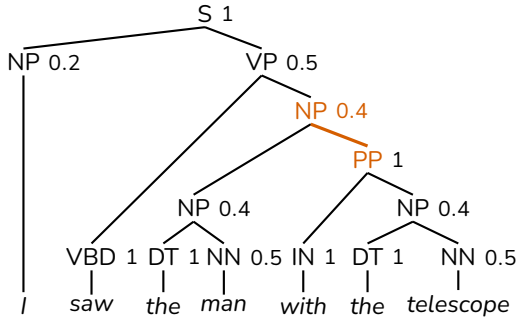
Multiply the rule probabilities beside the parents; factors of 1 are omitted below.

VP attachment



$$P(\mathbf{x}, T_{\text{VP}}) = 0.2 \times 0.4^2 \times 0.5^2 \times 0.5 \times 0.5 = \boxed{0.002}.$$

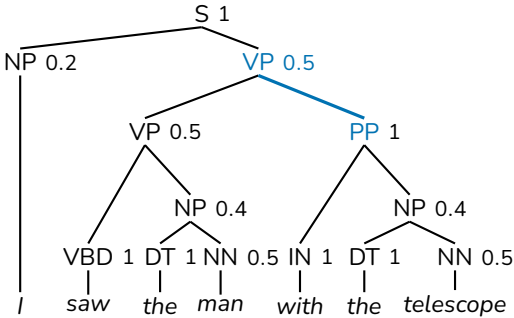
NP attachment



Two Trees, Two Probabilities

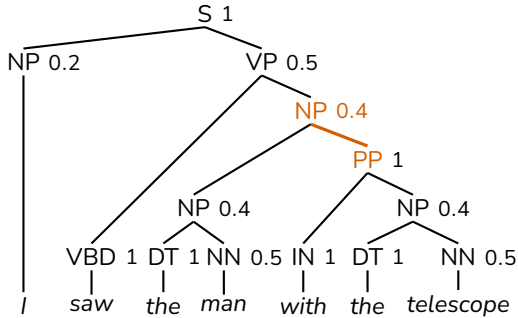
Multiply the rule probabilities beside the parents; factors of 1 are omitted below.

VP attachment



$$P(\mathbf{x}, T_{\text{VP}}) = 0.2 \times 0.4^2 \times 0.5^2 \times 0.5 \times 0.5 = \boxed{0.002}.$$

NP attachment



$$P(\mathbf{x}, T_{\text{NP}}) = 0.2 \times 0.4^2 \times 0.5^2 \times 0.5 \times 0.4 = \boxed{0.0016}.$$

Estimating Rule Probabilities

Given annotated trees, count rule applications and normalize by their left-hand side:

$$\widehat{p}(A \to \alpha) = \frac{\mathsf{count}(A \to \alpha)}{\sum_{\alpha'} \mathsf{count}(A \to \alpha')}.$$

The same formula covers structural and lexical rules.

Estimating Rule Probabilities

Given annotated trees, count rule applications and normalize by their left-hand side:

$$\widehat{p}(A \to \alpha) = \frac{\text{count}(A \to \alpha)}{\sum_{\alpha'} \text{count}(A \to \alpha')}.$$

The same formula covers structural and lexical rules.

Illustrative counts for NP:

Rule	Count	Estimated probability
NP $\to$ I	20	$20/100 = 0.2$
NP $\to$ DT NN	40	$40/100 = 0.4$
NP $\to$ NP PP	40	$40/100 = 0.4$

Count and normalize, just as in N-gram maximum likelihood estimation.

Chomsky Normal Form

In **Chomsky normal form (CNF)**, rules have one of two forms:

$$A \to B\ C \qquad \text{or} \qquad A \to w.$$

A, B, C are nonterminals and w is one terminal. Our attachment grammar already has this form, including NP $\to$ I.

Chomsky Normal Form

In **Chomsky normal form (CNF)**, rules have one of two forms:

$$A \to B\ C \qquad \text{or} \qquad A \to w.$$

A, B, C are nonterminals and w is one terminal. Our attachment grammar already has this form, including NP $\to$ I.

Binarization: introduce an auxiliary category to split a longer rule.

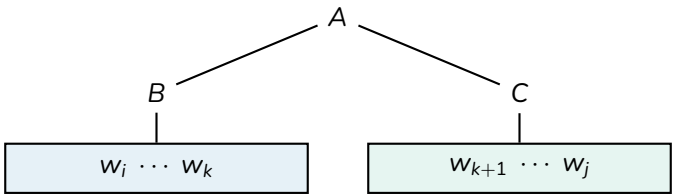
Original		Binary rules	
$A \to B\ C\ D$	[0.5]	$A \to B\ X$	[0.5]
		$X \to C\ D$	[1.0]

The new category X is specific to this rule, so the probability is unchanged.

General conversion also isolates terminals in mixed rules and eliminates unary nonterminal rules. For CFGs without the empty string, an equivalent CNF grammar exists.

CKY: Find the Tree with the Highest Probability

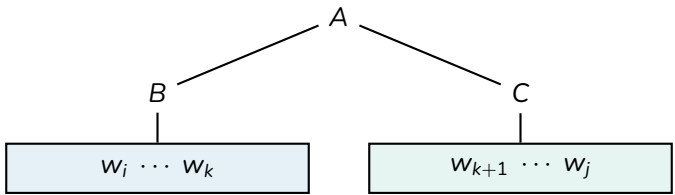
Goal: find the highest-probability tree for a fixed sentence. CKY (Cocke–Younger–Kasami) reuses solutions for smaller spans.



Split at k, with a rule $A \to B\ C$.

CKY: Find the Tree with the Highest Probability

Goal: find the highest-probability tree for a fixed sentence. CKY (Cocke–Younger–Kasami) reuses solutions for smaller spans.



Split at k, with a rule $A \to B\ C$.

- Store a best subtree for **each span and each root label**.
- Try all split points and compatible binary rules.
- Fill short spans before longer ones, ending with the full sentence.

CKY: The Recurrence

Let $V_{i,j}(A)$ be the probability of the **best subtree** rooted at A with yield $w_i \dots w_j$.

Single words:

$$V_{i,i}(A) = p(A \to w_i) \qquad \text{(0 if the rule is absent).}$$

CKY: The Recurrence

Let $V_{i,j}(A)$ be the probability of the **best subtree** rooted at A with yield $w_i \dots w_j$.

Single words:

$$V_{i,i}(A) = p(A \to w_i) \qquad \text{(0 if the rule is absent).}$$

Longer spans:

$$\boxed{V_{i,j}(A) = \max_{\substack{(A \to B\ C) \in R \\ i \le k < j}} p(A \to B\ C)\, V_{i,k}(B)\, V_{k+1,j}(C).}$$

Unreachable entries have value 0.

CKY: The Recurrence

Let $V_{i,j}(A)$ be the probability of the **best subtree** rooted at A with yield $w_i \dots w_j$.

Single words:

$$V_{i,i}(A) = p(A \to w_i) \qquad \text{(0 if the rule is absent).}$$

Longer spans:

$$\boxed{V_{i,j}(A) = \max_{\substack{(A \to B\ C) \in R \\ i \le k < j}} p(A \to B\ C)\, V_{i,k}(B)\, V_{k+1,j}(C).}$$

Unreachable entries have value 0.

Answer: $V_{1,n}(S)$ is the best tree's probability. Store the maximizing rule and split as a **backpointer** to recover the tree.

CKY: The Recurrence

Let $V_{i,j}(A)$ be the probability of the **best subtree** rooted at A with yield $w_i \dots w_j$.

Single words:

$$V_{i,i}(A) = p(A \to w_i) \qquad \text{(0 if the rule is absent).}$$

Longer spans:

$$\boxed{V_{i,j}(A) = \max_{\substack{(A \to B\ C) \in R \\ i \le k < j}} p(A \to B\ C)\, V_{i,k}(B)\, V_{k+1,j}(C).}$$

Unreachable entries have value 0.

Answer: $V_{1,n}(S)$ is the best tree's probability. Store the maximizing rule and split as a **backpointer** to recover the tree.

Time: $O(n^3|R|)$; space: $O(n^2|N|)$, storing one best derivation per entry.
Implement with log probabilities to avoid underflow.

CKY: Filling the Chart

Input: I_1 saw_2 the_3 man_4 with_5 the_6 telescope_7

$i \backslash j$	1	2	3	4	5	6	7
1	NP: .2						
2		VBD: 1					
3			DT: 1				
4				NN: .5			
5					IN: 1		
6						DT: 1	
7							NN: .5

Best probabilities per span and label; unreachable entries are omitted.

Initialize lexical entries using $p(A \rightarrow w_i)$.

Rule			p
S	$\rightarrow$	NP VP	1.0
NP	$\rightarrow$	DT NN	0.4
NP	$\rightarrow$	NP PP	0.4
VP	$\rightarrow$	VBD NP	0.5
VP	$\rightarrow$	VP PP	0.5
PP	$\rightarrow$	IN NP	1.0
NP	$\rightarrow$	I	0.2
DT	$\rightarrow$	the	1.0
NN	$\rightarrow$	man	0.5
NN	$\rightarrow$	telescope	0.5
VBD	$\rightarrow$	saw	1.0
IN	$\rightarrow$	with	1.0

CKY: Filling the Chart

Input: I_1 saw_2 the_3 man_4 with_5 the_6 telescope_7

$i \backslash j$	1	2	3	4	5	6	7
1	NP: .2						
2		VBD: 1					
3			DT: 1	NP: .2			
4				NN: .5			
5					IN: 1		
6						DT: 1	NP: .2
7							NN: .5

Best probabilities per span and label; unreachable entries are omitted.

Length 2: $V_{3,4}(\text{NP}) = 0.4 \times 1 \times 0.5 = 0.2$.
The same calculation gives $V_{6,7}(\text{NP}) = 0.2$.

Rule			p
S	$\rightarrow$	NP VP	1.0
NP	$\rightarrow$	DT NN	0.4
NP	$\rightarrow$	NP PP	0.4
VP	$\rightarrow$	VBD NP	0.5
VP	$\rightarrow$	VP PP	0.5
PP	$\rightarrow$	IN NP	1.0
NP	$\rightarrow$	I	0.2
DT	$\rightarrow$	the	1.0
NN	$\rightarrow$	man	0.5
NN	$\rightarrow$	telescope	0.5
VBD	$\rightarrow$	saw	1.0
IN	$\rightarrow$	with	1.0

CKY: Filling the Chart

Input: I_1 saw_2 the_3 man_4 with_5 the_6 telescope_7

$i \backslash j$	1	2	3	4	5	6	7
1	NP: .2			S: .02			
2		VBD: 1		VP: .1			
3			DT: 1	NP: .2			
4				NN: .5			
5					IN: 1		PP: .2
6						DT: 1	NP: .2
7							NN: .5

Best probabilities per span and label; unreachable entries are omitted.

Length 3: $V_{2,4}(\text{VP}) = 0.5 \times 1 \times 0.2 = 0.1$.
Length 3: $V_{5,7}(\text{PP}) = 1 \times 1 \times 0.2 = 0.2$.
Length 4: $V_{1,4}(\text{S}) = 1 \times 0.2 \times 0.1 = 0.02$.

Rule			p
S	$\rightarrow$	NP VP	1.0
NP	$\rightarrow$	DT NN	0.4
NP	$\rightarrow$	NP PP	0.4
VP	$\rightarrow$	VBD NP	0.5
VP	$\rightarrow$	VP PP	0.5
PP	$\rightarrow$	IN NP	1.0
NP	$\rightarrow$	I	0.2
DT	$\rightarrow$	the	1.0
NN	$\rightarrow$	man	0.5
NN	$\rightarrow$	telescope	0.5
VBD	$\rightarrow$	saw	1.0
IN	$\rightarrow$	with	1.0

CKY: Filling the Chart

Input: I_1 saw_2 the_3 man_4 with_5 the_6 telescope_7

$i \backslash j$	1	2	3	4	5	6	7
1	NP: .2			S: .02			
2		VBD: 1		VP: .1			
3			DT: 1	NP: .2			NP: .016
4				NN: .5			
5					IN: 1		PP: .2
6						DT: 1	NP: .2
7							NN: .5

Best probabilities per span and label; unreachable entries are omitted.

Length 5: $V_{3,7}(\text{NP}) = 0.4 \times 0.2 \times 0.2 = 0.016$.
This is the NP *the man with the telescope*.

Rule			p
S	$\rightarrow$	NP VP	1.0
NP	$\rightarrow$	DT NN	0.4
NP	$\rightarrow$	NP PP	0.4
VP	$\rightarrow$	VBD NP	0.5
VP	$\rightarrow$	VP PP	0.5
PP	$\rightarrow$	IN NP	1.0
NP	$\rightarrow$	I	0.2
DT	$\rightarrow$	the	1.0
NN	$\rightarrow$	man	0.5
NN	$\rightarrow$	telescope	0.5
VBD	$\rightarrow$	saw	1.0
IN	$\rightarrow$	with	1.0

CKY: Filling the Chart

Input: I_1 saw_2 the_3 man_4 with_5 the_6 telescope_7

$i \backslash j$	1	2	3	4	5	6	7
1	NP: .2			S: .02			
2		VBD: 1		VP: .1			VP: .01
3			DT: 1	NP: .2			NP: .016
4				NN: .5			
5					IN: 1		PP: .2
6						DT: 1	NP: .2
7							NN: .5

Best probabilities per span and label; unreachable entries are omitted.

Length 6: two ways to build VP over words 2–7.
Keep the larger probability: $V_{2,7}(\text{VP}) = 0.01$.

Rule			p
S	$\rightarrow$	NP VP	1.0
NP	$\rightarrow$	DT NN	0.4
NP	$\rightarrow$	NP PP	0.4
VP	$\rightarrow$	VBD NP	0.5
VP	$\rightarrow$	VP PP	0.5
PP	$\rightarrow$	IN NP	1.0
NP	$\rightarrow$	I	0.2
DT	$\rightarrow$	the	1.0
NN	$\rightarrow$	man	0.5
NN	$\rightarrow$	telescope	0.5
VBD	$\rightarrow$	saw	1.0
IN	$\rightarrow$	with	1.0

CKY: Filling the Chart

Input: I_1 saw_2 the_3 man_4 with_5 the_6 telescope_7

$i \backslash j$	1	2	3	4	5	6	7
1	NP: .2			S: .02			**S: .002**
2		VBD: 1		VP: .1			VP: .01
3			DT: 1	NP: .2			NP: .016
4				NN: .5			
5					IN: 1		PP: .2
6						DT: 1	NP: .2
7							NN: .5

Best probabilities per span and label; unreachable entries are omitted.

Length 7: $V_{1,7}(\text{S}) = 1 \times 0.2 \times 0.01 = \boxed{0.002}$.

Rule			p
S	$\rightarrow$	NP VP	1.0
NP	$\rightarrow$	DT NN	0.4
NP	$\rightarrow$	NP PP	0.4
VP	$\rightarrow$	VBD NP	0.5
VP	$\rightarrow$	VP PP	0.5
PP	$\rightarrow$	IN NP	1.0
NP	$\rightarrow$	I	0.2
DT	$\rightarrow$	the	1.0
NN	$\rightarrow$	man	0.5
NN	$\rightarrow$	telescope	0.5
VBD	$\rightarrow$	saw	1.0
IN	$\rightarrow$	with	1.0

CKY: Comparing Two Ways to Build the VP

The span $[2, 7]$ is *saw the man with the telescope*.

Rule and split	Child spans	Candidate probability
VP $\rightarrow$ VBD NP, $k = 2$	[saw] [the man with the telescope]	$0.5 \times 1 \times 0.016$ $= 0.008$

CKY: Comparing Two Ways to Build the VP

The span $[2, 7]$ is *saw the man with the telescope*.

Rule and split	Child spans	Candidate probability
VP $\rightarrow$ VBD NP, $k = 2$	[saw] [the man with the telescope]	$0.5 \times 1 \times 0.016 = 0.008$
VP $\rightarrow$ VP PP, $k = 4$	[saw the man] [with the telescope]	$0.5 \times 0.1 \times 0.2 = \mathbf{0.01}$

CKY: Comparing Two Ways to Build the VP

The span $[2, 7]$ is *saw the man with the telescope*.

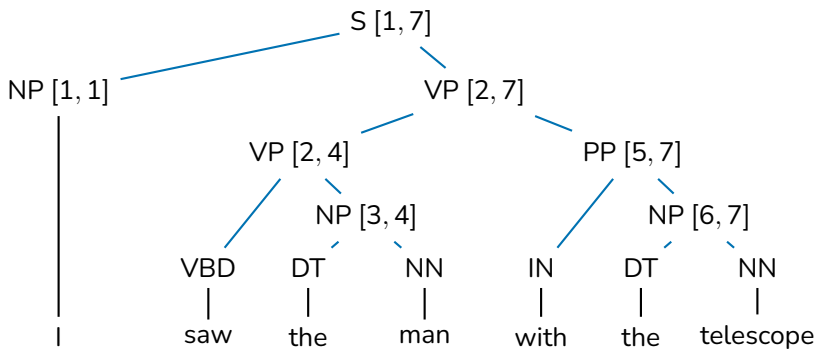
Rule and split	Child spans	Candidate probability
VP $\rightarrow$ VBD NP, $k = 2$	[saw] [the man with the telescope]	$0.5 \times 1 \times 0.016 = 0.008$
VP $\rightarrow$ VP PP, $k = 4$	[saw the man] [with the telescope]	$0.5 \times 0.1 \times 0.2 = \mathbf{0.01}$

Choose VP $\rightarrow$ VP PP at $k = 4$. Store that choice at $(2, 7, \text{VP})$, not just the value 0.01.

Alternative splits have an unreachable child or no compatible rule.

Recovering the Best Tree

Follow backpointers from $(1, 7, S)$ to the lexical entries.



The Best Tree Is Not the Sentence Probability

The grammar gives our sentence two parses:

$$P(\mathbf{x}, T_{\text{VP}}) = 0.002, \qquad P(\mathbf{x}, T_{\text{NP}}) = 0.0016.$$

The Best Tree Is Not the Sentence Probability

The grammar gives our sentence two parses:

$$P(\mathbf{x}, T_{\text{VP}}) = 0.002, \qquad P(\mathbf{x}, T_{\text{NP}}) = 0.0016.$$

For parsing, choose the best tree: $\max_T P(\mathbf{x}, T) = 0.002$.

For language modeling, sum over all trees with the same yield:

$$P(\mathbf{x}) = \sum_{T:\, \text{yield}(T)=\mathbf{x}} P(\mathbf{x}, T) = 0.002 + 0.0016 = \boxed{0.0036}.$$

The Best Tree Is Not the Sentence Probability

The grammar gives our sentence two parses:

$$P(\mathbf{x}, T_{\text{VP}}) = 0.002, \qquad P(\mathbf{x}, T_{\text{NP}}) = 0.0016.$$

For parsing, choose the best tree: $\max_T P(\mathbf{x}, T) = 0.002$.

For language modeling, sum over all trees with the same yield:

$$P(\mathbf{x}) = \sum_{T:\, \text{yield}(T)=\mathbf{x}} P(\mathbf{x}, T) = 0.002 + 0.0016 = \boxed{0.0036}.$$

A sentence may have exponentially many parses. Can we compute this sum without enumerating every tree?

The Inside Algorithm

Let $I_{i,j}(A)$ be the **total probability** of generating $w_i \dots w_j$ from A, summed over its subtrees.

Single words:

$$I_{i,i}(A) = p(A \to w_i) \qquad \text{(0 if absent)}.$$

The Inside Algorithm

Let $I_{i,j}(A)$ be the **total probability** of generating $w_i \dots w_j$ from A, summed over its subtrees.

Single words:

$$I_{i,i}(A) = p(A \to w_i) \qquad \text{(0 if absent)}.$$

Longer spans:

$$\boxed{I_{i,j}(A) = \sum_{\substack{(A \to B\ C) \in R \\ i \le k < j}} p(A \to B\ C)\, I_{i,k}(B)\, I_{k+1,j}(C).}$$

The Inside Algorithm

Let $I_{i,j}(A)$ be the **total probability** of generating $w_i \dots w_j$ from A, summed over its subtrees.

Single words:

$$I_{i,i}(A) = p(A \to w_i) \qquad \text{(0 if absent)}.$$

Longer spans:

$$\boxed{I_{i,j}(A) = \sum_{\substack{(A \to B\ C) \in R \\ i \le k < j}} p(A \to B\ C)\, I_{i,k}(B)\, I_{k+1,j}(C).}$$

Same chart and fill order as CKY, but **sum over alternatives instead of taking their maximum**. The result is $I_{1,n}(S) = P(\mathbf{x})$.

Time: $O(n^3|R|)$; space: $O(n^2|N|)$. For a fixed grammar, these become $O(n^3)$ and $O(n^2)$.

Inside: Reuse the Same Two VP Candidates

In our example, each smaller reachable span has only one tree. Its inside value therefore equals its CKY value.

$$
\begin{aligned}
I_{2,7}(\mathrm{VP}) &= \underbrace{0.5 \times 1 \times 0.016}_{\text{NP attachment}} + \underbrace{0.5 \times 0.1 \times 0.2}_{\text{VP attachment}} \\
&= 0.008 + 0.01 = 0.018.
\end{aligned}
$$

Inside: Reuse the Same Two VP Candidates

In our example, each smaller reachable span has only one tree. Its inside value therefore equals its CKY value.

$$I_{2,7}(\mathrm{VP}) = \underbrace{0.5 \times 1 \times 0.016}_{\text{NP attachment}} + \underbrace{0.5 \times 0.1 \times 0.2}_{\text{VP attachment}}$$
$$= 0.008 + 0.01 = 0.018.$$

$$I_{1,7}(\mathrm{S}) = 1 \times 0.2 \times 0.018 = \boxed{0.0036}.$$

In general, each child inside value may already sum over many trees. Multiplying the child sums accounts for every compatible pair of subtrees.

What the PCFG Does Not Condition On

In a plain PCFG, the rule distribution depends only on the nonterminal being expanded:

$$p(\text{VP} \rightarrow \text{VP PP}) = 0.5, \qquad p(\text{NP} \rightarrow \text{NP PP}) = 0.4.$$

These attachment preferences do not inspect the surrounding words.

What the PCFG Does Not Condition On

In a plain PCFG, the rule distribution depends only on the nonterminal being expanded:

$$p(\text{VP} \rightarrow \text{VP PP}) = 0.5, \qquad p(\text{NP} \rightarrow \text{NP PP}) = 0.4.$$

These attachment preferences do not inspect the surrounding words.

Compare:

- I saw the man **with the telescope**.
- I saw the man **with the hat**.

Even after adding *hat* to the noun vocabulary, the structural rule probabilities stay the same. Word meanings and context affect which attachment is plausible.

What the PCFG Does Not Condition On

In a plain PCFG, the rule distribution depends only on the nonterminal being expanded:

$$p(\text{VP} \rightarrow \text{VP PP}) = 0.5, \qquad p(\text{NP} \rightarrow \text{NP PP}) = 0.4.$$

These attachment preferences do not inspect the surrounding words.

Compare:

- I saw the man **with the telescope**.
- I saw the man **with the hat**.

Even after adding *hat* to the noun vocabulary, the structural rule probabilities stay the same. Word meanings and context affect which attachment is plausible.

More detailed grammar states or lexical conditioning can help. Neural parsers can instead score spans using contextual representations.

[Collins, 2003; Kitaev and Klein, 2018.]

Heads of Phrases

A phrase's **head** determines its central syntactic properties.

Phrase	Category	Head
the man	NP	
saw the man	VP	
with the telescope	PP	

Heads of Phrases

A phrase's **head** determines its central syntactic properties.

Phrase	Category	Head
the man	NP	**man** (noun)
saw the man	VP	**saw** (verb)
with the telescope	PP	**with** (preposition)

Heads of Phrases

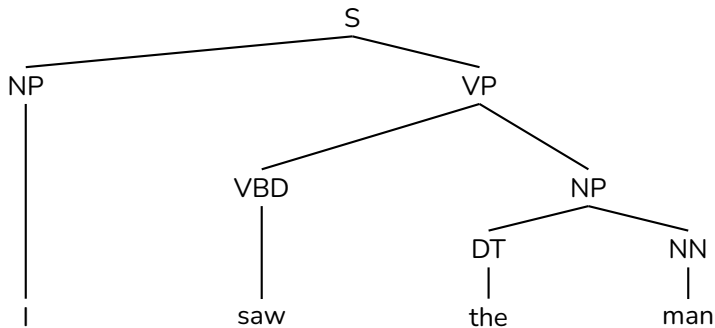
A phrase's **head** determines its central syntactic properties.

Phrase	Category	Head
the man	NP	**man** (noun)
saw the man	VP	**saw** (verb)
with the telescope	PP	**with** (preposition)

Head rules select one child of each constituent and propagate its lexical head upward. Choosing head conventions is part of specifying the representation.

Constituency and Dependency

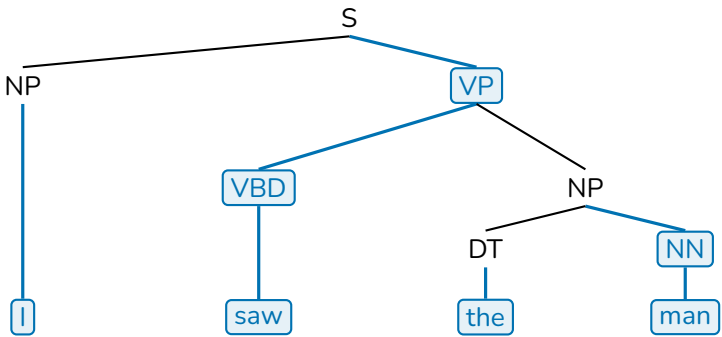
Constituency: nested phrases. **Dependency:** word-to-word relations.



Start with the constituency tree for *I saw the man*.

Constituency and Dependency

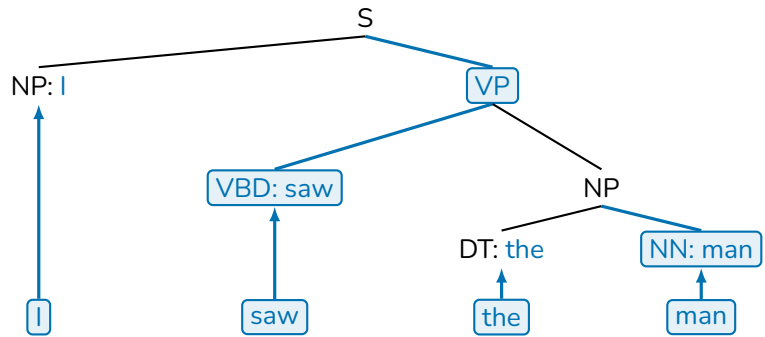
Constituency: nested phrases. **Dependency:** word-to-word relations.



Blue marks each parent's **head child**.
At branching nodes: S chooses VP, VP chooses VBD, and NP chooses NN.

Constituency and Dependency

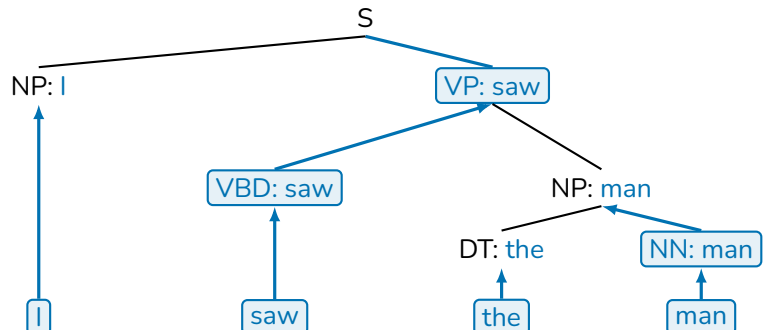
Constituency: nested phrases. **Dependency:** word-to-word relations.



Copy each terminal word to its parent.

Constituency and Dependency

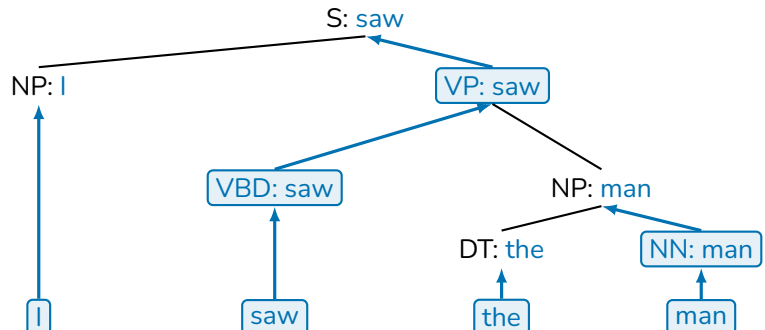
Constituency: nested phrases. **Dependency:** word-to-word relations.



Propagate *man* to NP and *saw* to VP along the selected head edges.

Constituency and Dependency

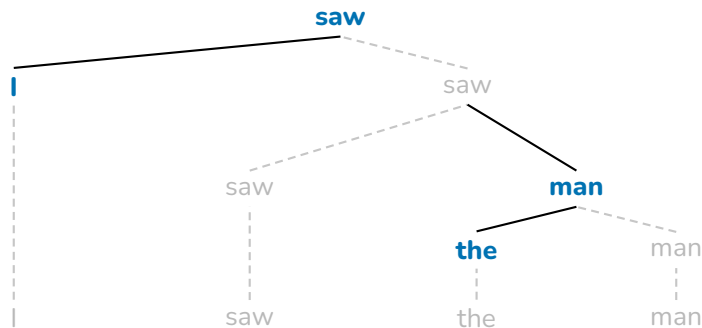
Constituency: nested phrases. **Dependency:** word-to-word relations.



Propagate *saw* from VP to S, all the way to the root.

Constituency and Dependency

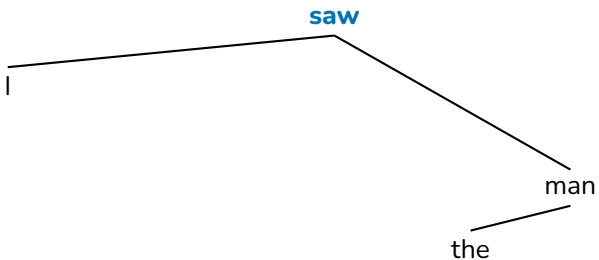
Constituency: nested phrases. **Dependency:** word-to-word relations.



Keep the highest copy of each word; fade the lower copies on its head path.

Constituency and Dependency

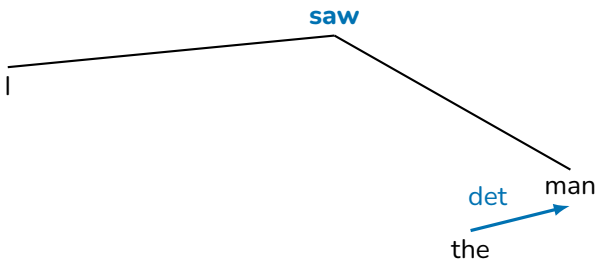
Constituency: nested phrases. **Dependency:** word-to-word relations.



Remove the lower copies and reconnect the remaining words.

Constituency and Dependency

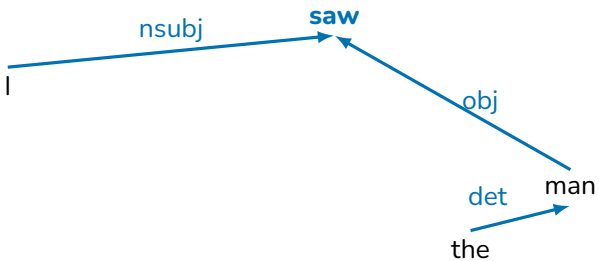
Constituency: nested phrases. **Dependency:** word-to-word relations.



Add the determiner dependency: *the* → *man*.
Here, arrows point from **dependent to head**.

Constituency and Dependency

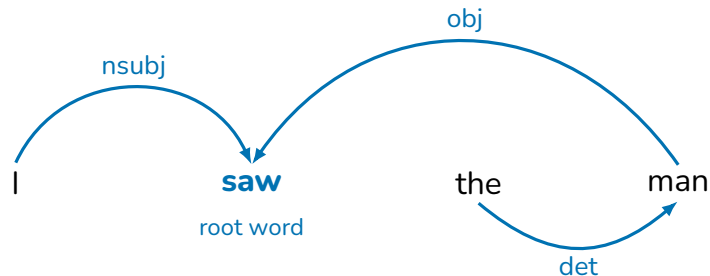
Constituency: nested phrases. **Dependency:** word-to-word relations.



Add *I* → *saw* and *man* → *saw*.
Here, arrows point from **dependent to head**.

Constituency and Dependency

Constituency: nested phrases. **Dependency:** word-to-word relations.



Each non-root word has one head; *saw* is the root word.
Here, arrows point from **dependent to head**.

Extracting Dependencies with Head Rules

For *I saw the man*, choose the noun as the NP head, the verb as the VP head, and the VP as the S head.

Constituent	Head child	Other child's head	New dependency
NP: the man	NN: man	DT: the	
VP: saw the man	VBD: saw	NP: man	
S: I saw the man	VP: saw	NP: I	

Extracting Dependencies with Head Rules

For *I saw the man*, choose the noun as the NP head, the verb as the VP head, and the VP as the S head.

Constituent	Head child	Other child's head	New dependency
NP: the man	NN: man	DT: the	the $\rightarrow$ man
VP: saw the man	VBD: saw	NP: man	
S: I saw the man	VP: saw	NP: I	

Extracting Dependencies with Head Rules

For *I saw the man*, choose the noun as the NP head, the verb as the VP head, and the VP as the S head.

Constituent	Head child	Other child's head	New dependency
NP: the man	NN: man	DT: the	the $\rightarrow$ man
VP: saw the man	VBD: saw	NP: man	man $\rightarrow$ saw
S: I saw the man	VP: saw	NP: I	

Extracting Dependencies with Head Rules

For *I saw the man*, choose the noun as the NP head, the verb as the VP head, and the VP as the S head.

Constituent	Head child	Other child's head	New dependency
NP: the man	NN: man	DT: the	the $\rightarrow$ man
VP: saw the man	VBD: saw	NP: man	man $\rightarrow$ saw
S: I saw the man	VP: saw	NP: I	I $\rightarrow$ saw

Extracting Dependencies with Head Rules

For *I saw the man*, choose the noun as the NP head, the verb as the VP head, and the VP as the S head.

Constituent	Head child	Other child's head	New dependency
NP: the man	NN: man	DT: the	the $\rightarrow$ man
VP: saw the man	VBD: saw	NP: man	man $\rightarrow$ saw
S: I saw the man	VP: saw	NP: I	I $\rightarrow$ saw

For each non-head child, draw an arrow from its lexical head to the parent's lexical head. The lexical head at the top, **saw**, becomes the root word.

Dependency labels require labeling conventions in addition to these head choices. The resulting arcs do not retain every phrase boundary from the constituency tree.

Readings

Required:

- Jurafsky and Martin, *Speech and Language Processing* (2026 draft).
  Chapter 19: Context-Free Grammars and Constituency Parsing.
  Chapter 20: Dependency Parsing.

Further reading:

- Marcus, Santorini, and Marcinkiewicz (1993).
  Building a Large Annotated Corpus of English: The Penn Treebank.
  Computational Linguistics, 19(2).
- Collins (2003).
  Head-Driven Statistical Models for Natural Language Parsing.
- Kitaev and Klein (2018).
  Constituency Parsing with a Self-Attentive Encoder.

Next Lecture

Probing Neural LMs I: Syntactic Knowledge

We now have explicit representations of syntax. How can we test whether a trained language model has learned them?