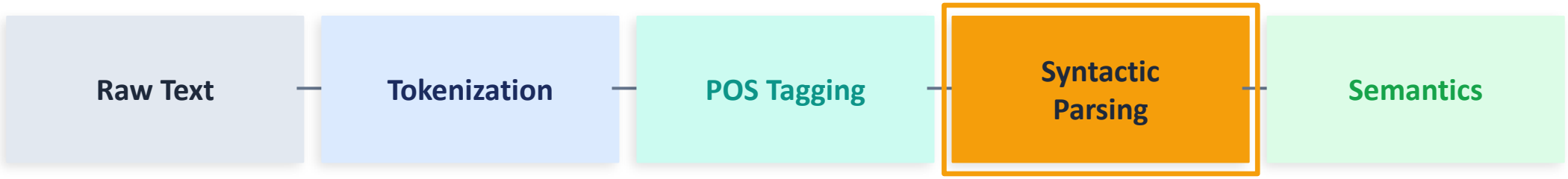
Syntactic Parsing
Constituency & Dependency Parsing

PREPARED BY: AHMAD ALAA ALDINE

The NLP Pipeline – Where Are We?



Tokenization gives us:

```
words: ['I', 'saw', 'the', 'man', 'with', 'the', 'telescope']
```

POS tagging gives us:

```
tags: [PRP, VBD, DT, NN, IN, DT, NN]
```

But we still don't know:

```
Who has the telescope? Relationships between words?
```

Why Tokenization & POS Tagging Are Not Enough

WHAT THEY GIVE US ✓

- ✓ Individual word identities
- ✓ Grammatical category (noun, verb…)
- ✓ Local morphological features

WHAT THEY MISS ✗

- ✗ Which words modify which others
- ✗ Argument structure (who did what to whom)
- ✗ Scope and grouping of phrases
- ✗ Hierarchical relationships

Syntax vs. Semantics

SYNTAX

Form & Structure

- Rules governing sentence structure
- Word order and grammatical relations
- Phrase groupings and hierarchies
- Language-level constraints

"Colorless green ideas sleep furiously."

✓ **Grammatically VALID**

SEMANTICS

Meaning & Content

- Meaning of words and sentences
- Truth conditions and entailments
- Compositional meaning-building
- World-knowledge constraints

"Colorless green ideas sleep furiously."

✗ **Semantically NONSENSICAL**

Noam Chomsky (1957) used this sentence to show syntax and semantics are independent dimensions of language.

Structural Ambiguity – The Core Challenge

Same words, same POS tags — yet multiple valid meanings

"I saw the man with the telescope."

Reading 1

I used a telescope to see the man.
[VP attachment: telescope modifies 'saw']

Reading 2

I saw a man who had a telescope.
[NP attachment: telescope modifies 'man']

Key Insight: Tokenization and POS tagging CANNOT distinguish these two readings. We need structure — this is exactly why syntactic parsing exists.

Structural Ambiguity Example 1: PP Attachment

Where does the Prepositional Phrase attach?

"We saw the Eiffel Tower visiting Paris."

Attach to VP (saw)

```
[VP saw [NP the Eiffel Tower] [PP visiting Paris]]
```

We were visiting Paris when we saw it.

Attach to NP (Tower)

```
[VP saw [NP the Eiffel Tower [PP visiting Paris]]]
```

The Eiffel Tower was the thing visiting Paris? 🤔

PP attachment ambiguity is one of the most common sources of parse ambiguity. In English, a sentence with n PPs can have exponentially many parse trees!

Structural Ambiguity Example 2: Coordination Ambiguity

"Old men and women got discount tickets."

Reading A – 'old' modifies both

```
[[old men] and [old women]]
```

Both men AND women are old.

Reading B – 'old' modifies only men

```
[[old men] and [women]]
```

Only the men are old; the women may be any age.

More coordination ambiguity:

"I saw [nice boys and girls]" vs "I saw [nice boys] and [girls]"

Structural Ambiguity Example 3: Garden-Path Sentences

Sentences that mislead the parser before a later word forces reanalysis

"The horse raced past the barn fell."

The horse raced past the barn [fell → unexpected!]

The horse [that was] raced past the barn fell.

"The complex houses married couples."

'complex' = adjective, 'houses' = noun → but then 'married' breaks it

'complex' = noun (building), 'houses' = verb → The complex houses [verb] them.

Constituency Parsing

PHRASE STRUCTURE · CONTEXT-FREE GRAMMARS · PARSE TREES

Constituency & Phrase Structure Grammar

Syntactic constituency is the idea that groups of words can behave as single units.
A constituency parser groups words into nested phrases (constituents).

```
"[S [NP The cat] [VP [V sat] [PP on [NP the mat]]]]"
```

S **Sentence**

The cat sat on the mat

NP **Noun Phrase**

"the cat", "the mat"

VP **Verb Phrase**

"sat on the mat"

PP **Prepositional Phrase**

"on the mat"

V **Verb**

"sat"

Key idea: phrases can be nested! An NP can contain a PP, which can contain another NP → recursion!

Context-Free Grammar (CFG)

A formal system to describe phrase structure with rewrite rules

A CFG is a 4-tuple G = (N, Σ, R, S) where:

N = Non-terminals | Σ = Terminals (words) | R = Rules | S = Start symbol

Example CFG Rules:

```
S  → NP VP
NP → Det N
NP → Det N PP
VP → V NP
VP → V NP PP
PP → P NP
Det → 'the' | 'a'
N  → 'cat' | 'mat' | 'telescope'
V  → 'sat' | 'saw'
P  → 'on' | 'with'
```

CFG Properties:

- Context-free: LHS is always a single non-terminal
- Recursive: NP → Det N PP → P NP → Det N PP → ...
- Generates all valid trees for a language
- Parseable in $O(n^3)$ with CYK, Earley algorithms

Non-Terminals vs. Terminals

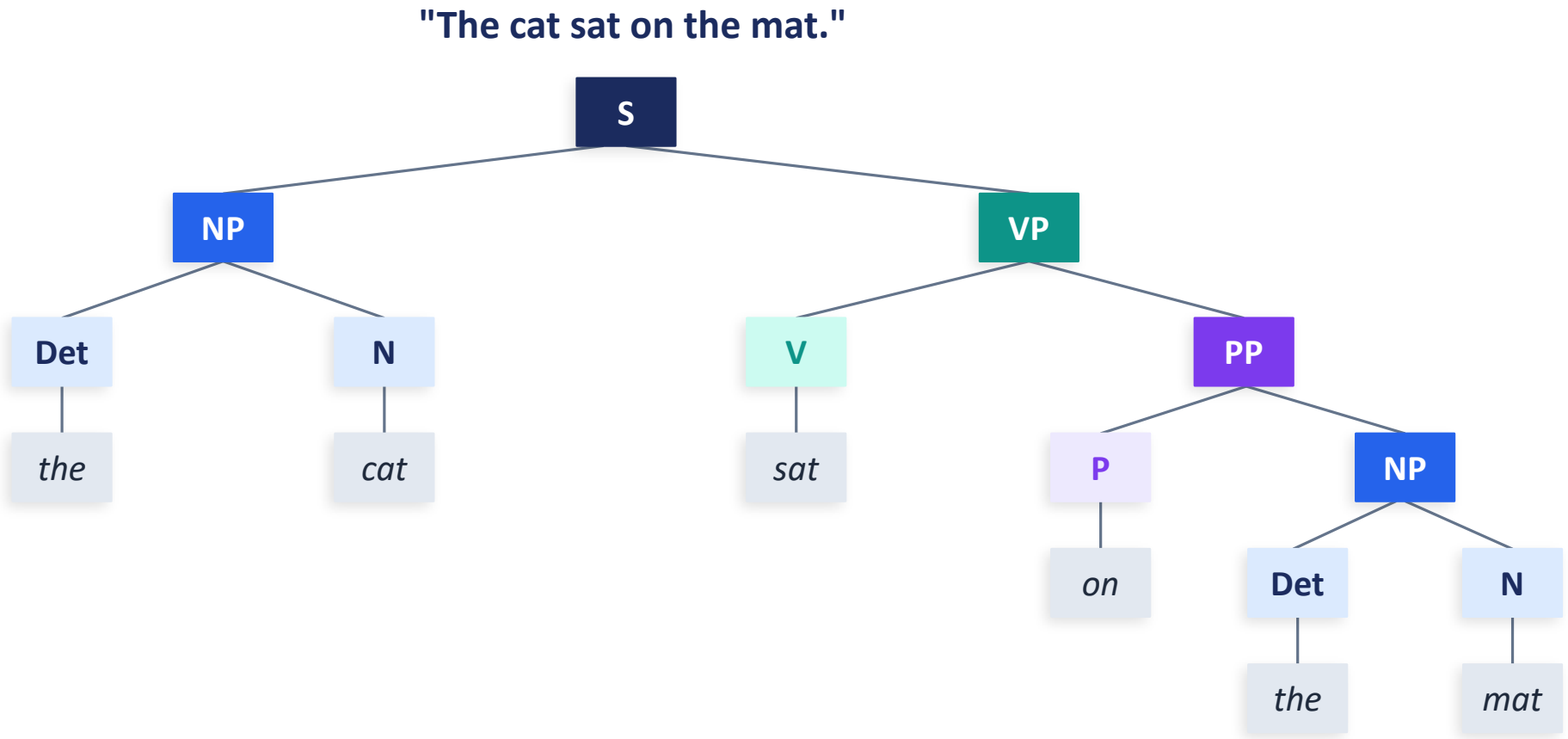
Non-Terminals (N) Abstract grammatical categories	
S	Sentence – the root symbol
NP	Noun Phrase
VP	Verb Phrase
PP	Prepositional Phrase
ADJP	Adjective Phrase
ADVP	Adverb Phrase

Appear on both side of rules, except S appear on LEFT only.
Never as leaves of the parse tree.

Terminals (Σ) Actual words in the vocabulary	
the	Determiner
cat	Noun
sat	Verb (past tense)
on	Preposition
quickly	Adverb
beautiful	Adjective

Appear on RIGHT side only.
They ARE the leaves of the parse tree.

Parse Trees – Visualizing Structure



Hierarchical Structure & Recursion in CFGs

Phrases can be nested inside phrases — this is what makes language productive

```
[S
  [NP the spy ]
  [VP reported
    [NP [Det the] [N report]
      [PP [P from] [NP Moscow] ] ] ] ]
```

Why Hierarchy Matters:

🔁 Recursion: NP → Det N PP, PP → P NP → Det N PP → ... Infinitely many sentences from finite rules.

🎯 Compositionality: Meaning of a phrase = function of its parts. Structure tells us HOW to compose meaning.

🔍 Long-range dependencies: "The dog that chased the cat that ate the mouse ran away" — subject-verb agree across distance.

CFG Derivation – Step by Step

Deriving: "The cat sat on the mat" from S

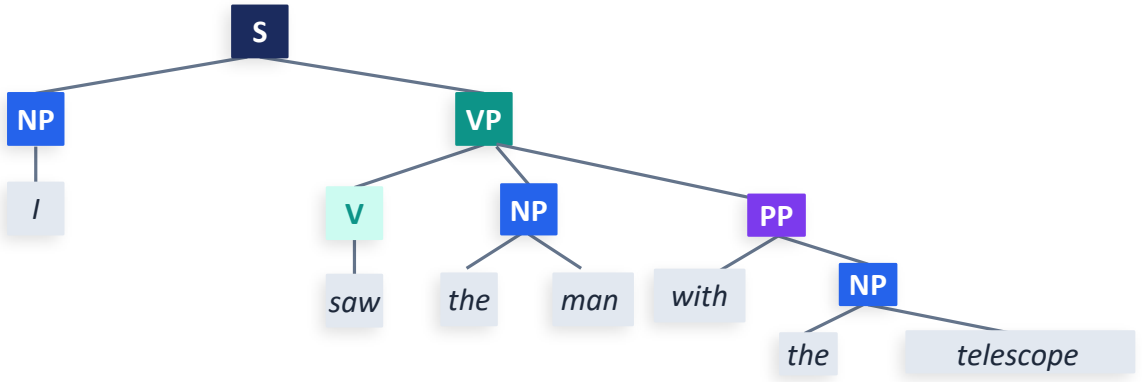
1	S → NP VP **Rule Applied**	NP VP
2	NP → Det N	Det N VP
3	Det → 'the'	the N VP
4	N → 'cat'	the cat VP
5	VP → V PP	the cat V PP
6	V → 'sat'	the cat sat PP
7	PP → P NP	the cat sat P NP
8	P → 'on', NP → Det N	the cat sat on Det N
9	Det → 'the', N → 'mat'	**the cat sat on the mat** ✓

Parse Tree Ambiguity – Two Trees, One Sentence

"I saw the man with the telescope." → 2 parse trees

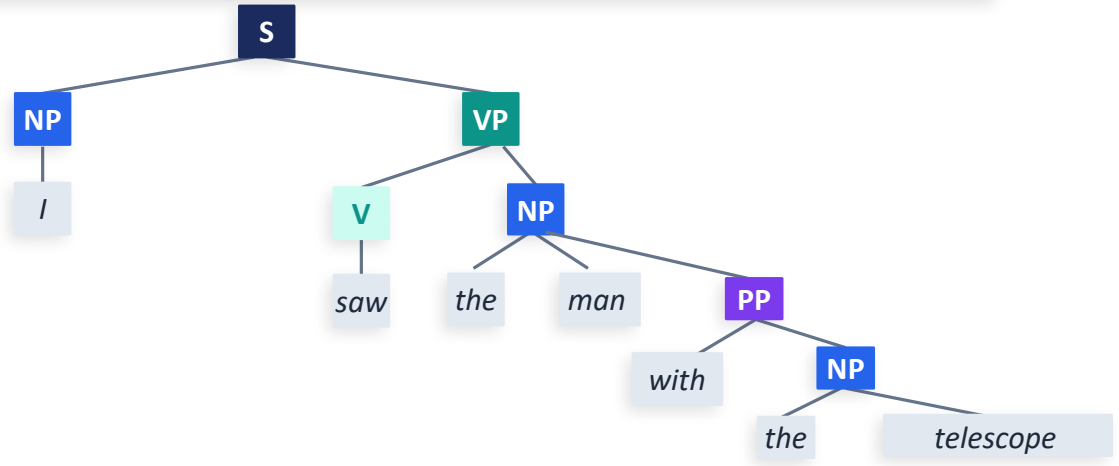
Tree 1: VP Attachment (I used the telescope)

```
[S [NP I]
  [VP [V saw]
      [NP [the man]]
      [PP with NP [the telescope]]]]
```



Tree 2: NP Attachment (the man had the telescope)

```
[S [NP I]
  [VP [V saw]
      [NP [the man]
          [PP with NP[the telescope]]]]]
```

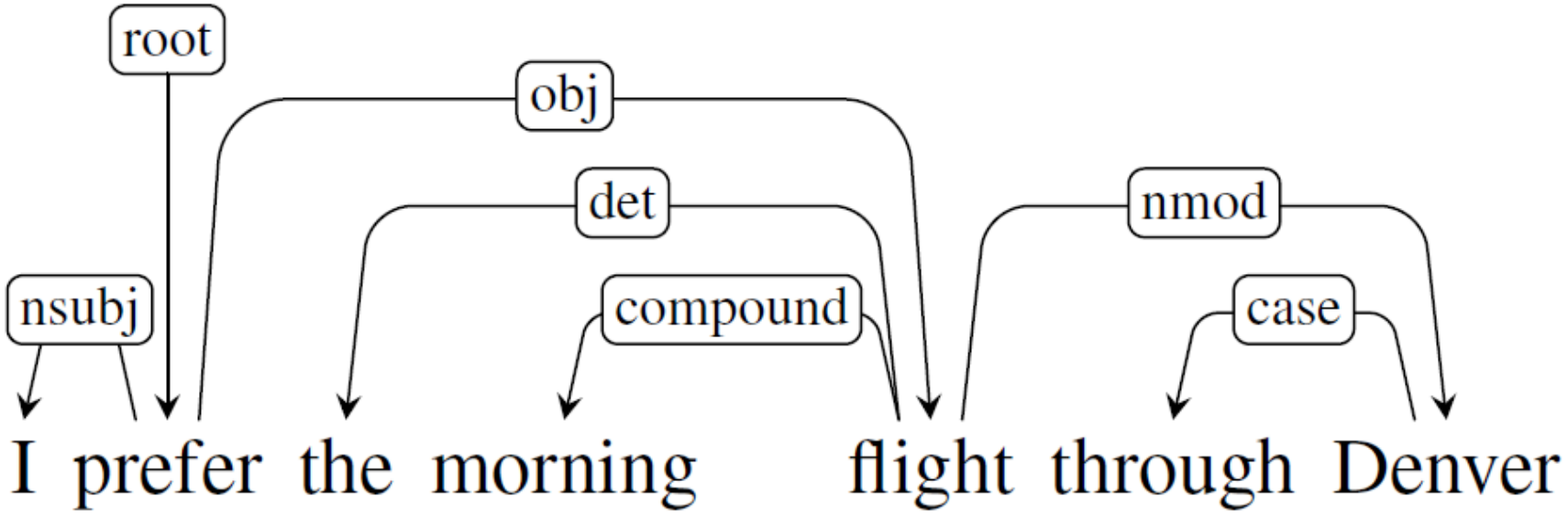


Dependency Parsing

MAPPING WORD-TO-WORD GRAMMATICAL RELATIONS

Dependency Parsing

The syntactic structure of a sentence is described solely in terms of directed **binary grammatical relations** between the words.



Head–Dependent Relations

Every word depends on exactly one other word (its head) — forming a tree

HEAD: the central, syntactically dominant word of a phrase.
DEPENDENT: a word that modifies or complements the head.

Example: "The dog eats fish quickly"

eats ← dog [nsubj]
Subject: who is eating?

eats ← fish [obj]
Object: what is being eaten?

eats ← quickly [advmod]
Adverb: how is eating done?

eats ← dog [nsubj]
Nominal subject relation

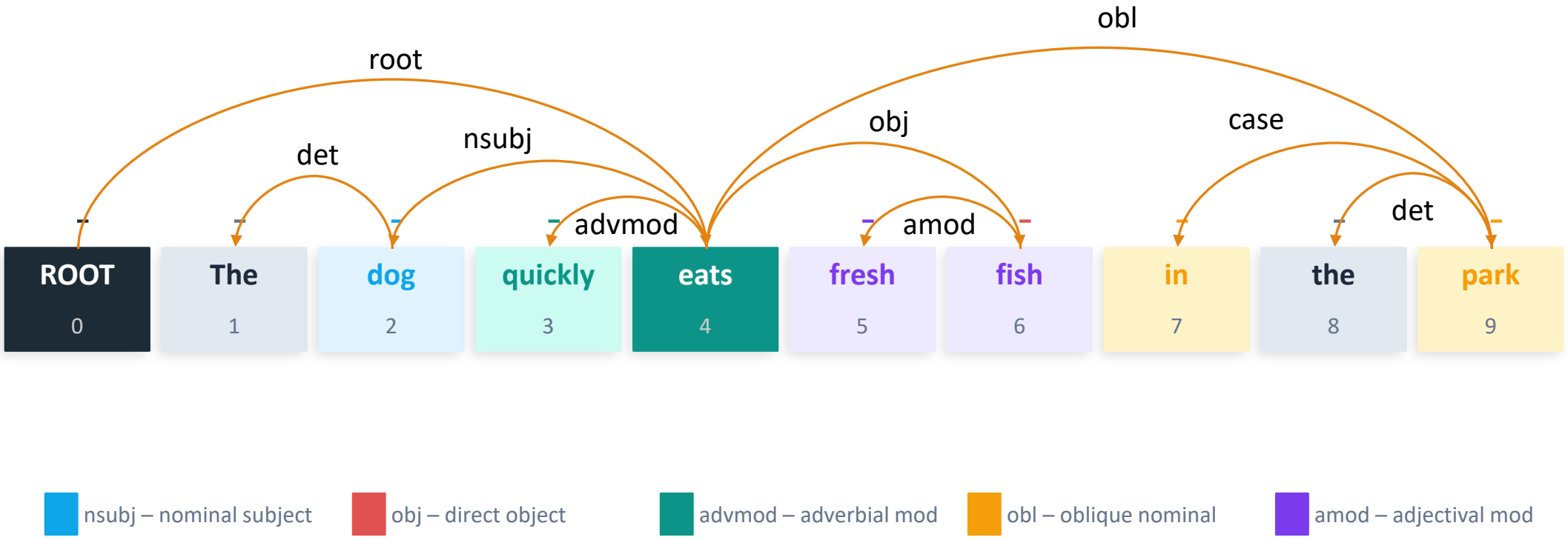
eats ← fish [obj]
Direct object relation

fish ← the [det]
Determiner attached to noun

Key rule: Every sentence has exactly one ROOT word. All other words trace a path up to ROOT. No cycles allowed.

Dependency Graph – Visualizing Word Relations

"The dog quickly eats fresh fish in the park."



Why Dependency Parsing Is Useful in NLP

Direct word-to-word links power a wide range of NLP tasks

❓ Question Answering (QA)

Q: "Who ate the fish?" → nsubj(ate, dog) → Answer: "The dog"

🌎 Machine Translation

Reorder words across languages using head-dep structure, not phrase brackets

🖇️ Information Extraction (IE)

Extract (subject, relation, object) triples: nsubj + root + obj → ("dog", "eats", "fish")

📋 Semantic Role Labeling (SRL)

Who did what to whom? Dependencies identify predicate-argument structure efficiently

😊 Sentiment Analysis

"not good" → neg(good, not) — dependency captures negation that bag-of-words misses

🧠 Coreference Resolution

Link pronouns to referents via shared grammatical roles across sentences

Constituency vs. Dependency Parsing

"The dog eats fish."

CONSTITUENCY

```
[S
  [NP [Det The] [N dog]]
  [VP [V eats]
      [NP [N fish]]]]
```

- ✓ Groups words into phrases
- ✓ Reveals phrase boundaries
- ✓ Good for syntax-level tasks

DEPENDENCY

```
eats → ROOT
dog  → eats  [nsubj]
The  → dog   [det]
fish → eats  [obj]
```

- ✓ Direct word-to-word links
- ✓ Language-agnostic structure
- ✓ Great for SRL, QA, IE

Universal Dependencies (UD)

An open-source multilingual treebank project — 100+ languages, standardised relations

UD defines a universal set of **37 dependency relations** and **17 POS tags** that apply consistently across all languages, enabling cross-lingual transfer and joint training.

Core UD Relations with Examples:

nsubj	Nominal Subject	The dog eats → dog → eats	obj	Object	eats fish → fish → eats
iobj	Indirect Object	gives me a book → me → gives	det	Determiner	the dog → the → dog
amod	Adjectival Modifier	big dog → big → dog	advmod	Adverb Modifier	runs quickly → quickly → runs
nmod	Nominal Modifier	book of John → John → book	case	Case Marking	in the park → in → park
aux	Auxiliary	has eaten → has → eaten	conj	Conjunct	cats and dogs → dogs → cats
cc	Coordinating Conj	and / but / or → cc → head	punct	Punctuation	. ! ? , → head → punct

UD Across Languages – Same Relations, Different Orders

Dependencies capture grammar cross-lingually despite different word orders

GB English: "The dog eats fish." (SVO — Subject–Verb–Object)

```
dog →[nsubj]→ eats     fish →[obj]→ eats     The →[det]→ dog
```

JP Japanese: "犬が魚を食べる" (SOV — Subject–Object–Verb)

```
犬(dog) →[nsubj]→ 食べる(eats)     魚(fish) →[obj]→ 食べる(eats)
```

SA Arabic: "يأكل الكلب السمكة" (VSO — Verb–Subject–Object)

```
╫ᵕ‿‿ᵔ╗(dog) →[nsubj]→ ᵕ╷ᵀ‿(eats)     ╜‿‿ᵕᵔ╗(fish) →[obj]→ ᵕ╷ᵀ‿(eats)
```

Key insight: The **dependency labels are identical** across all three languages — only the linear word order changes. A model trained on English can transfer UD knowledge to other languages!

Dependency Parsing Algorithms

HOW MACHINES FIND DEPENDENCY STRUCTURE (NO MATH — INTUITION ONLY)

Transition-Based Parsing (Arc-Standard)

Parsing "The dog eats fish" step by step with a stack and buffer

STACK — Words being processed (push/pop)

BUFFER — Input words waiting to be read

ARCS — Dependency edges already built

Step	Action	Stack	Buffer	Arcs so far
Init	–	[ROOT]	[The, dog, eats, fish]	–
SHIFT	Shift	[ROOT, The]	[dog, eats, fish]	–
SHIFT	Shift	[ROOT, The, dog]	[eats, fish]	–
LEFT-ARC	Left-Arc det	[ROOT, dog]	[eats, fish]	det(dog,The)
SHIFT	Shift	[ROOT,dog,eats]	[fish]	det(dog,The)
LEFT-ARC	Left-Arc nsubj	[ROOT, eats]	[fish]	…nsubj(eats,dog)
SHIFT	Shift	[ROOT,eats,fish]	[ ]	…
RIGHT-ARC	Right-Arc obj	[ROOT, eats]	[ ]	…obj(eats,fish)
RIGHT-ARC	Right-Arc root	[ROOT]	[ ]	root(ROOT,eats) ✓

The Three Core Transitions – Intuition

At each step the parser chooses ONE action — this is a sequence of local decisions



SHIFT

When: Buffer is not empty
Action: Move the next word from the buffer onto the top of the stack
You're reading left-to-right — pick up the next word

```
Stack:[ROOT,The]  Buffer:[dog,eats]  →  Stack:[ROOT,The,dog]  Buffer:[eats]
```



LEFT-ARC (label)

When: Top of stack depends on second-top of stack
Action: Create arc: second-to-top → top. Remove second-to-top from stack.
The right word is the head; the left word is the dependent

```
Stack:[ROOT,The,dog]  →  arc det(dog,The)  Stack:[ROOT,dog]
```



RIGHT-ARC (label)

When: Top of stack depends on second-top's position
Action: Create arc: top → second-to-top. Remove top from stack.
The left word is the head; the right word is the dependent

```
Stack:[ROOT,eats,fish]  →  arc obj(eats,fish)  Stack:[ROOT,eats]
```

Total complexity: O(n) transitions for a sentence of n words — extremely fast!

Graph-Based Parsing – The Big Picture

Score ALL possible arcs first, then find the best tree globally

Transition-Based

- Process left to right, one word at a time
- Choose SHIFT / LEFT-ARC / RIGHT-ARC at each step
- Greedy — each decision is local and final
- O(n) steps — very fast (real-time parsing!)
- May propagate early mistakes downstream

Used in:

spaCy · Stanford NLP (old) · UDPipe

Graph-Based

- Score every possible (head, dependent, label) arc
- Build a complete directed graph of scores
- Find Maximum Spanning Tree (MST)
- $O(n^2)$ or $O(n^3)$ — slower but globally optimal
- Not constrained by left-to-right ordering

Used in:

Biaffine Parser (Dozat&Manning 2017) · SuPar · DiaParser

Neural Networks Learn Dependency Structure

Modern parsers replace hand-crafted features with learned representations

Rule-Based
1960s–80s

Hand-written grammars
Slow, brittle, not scalable

Feature-Based ML
1990s–2000s

SVMs, MaxEnt
Hundreds of hand-crafted features

Neural MLP
2014 (Chen & Manning)

Replace features with word/tag embeddings

Biaffine Attention
2017 (Dozat & Manning)

Score head-dep pairs
State-of-the-art

Pre-trained LMs
2018–now (BERT etc.)

Contextual embeddings
Near-human accuracy

How a Neural Dependency Parser Works (Biaffine, simplified):

1. Input

Sentence tokens:
"The dog eats fish"

2. Embed

BERT / word vectors:
Context-aware token reps

3. Encode

BiLSTM or
Transformer layers

4. Score Arcs

MLP scores every (head, dep) pair

5. Score Labels

MLP scores each dep label

6. Decode

MST or greedy
→ final dep tree

Key advantage: Neural parsers learn **distributed representations** — no need to enumerate hand-crafted features like 'POS tag of word i-1' etc. BERT embeddings capture long-range context automatically.

Evaluation

HOW DO WE MEASURE IF A DEPENDENCY PARSER IS CORRECT?

UAS and LAS – The Two Key Metrics

Both metrics compare the predicted tree against the gold (human-annotated) tree

UAS — Unlabeled Attachment Score

What it measures:
% of words where the predicted HEAD is correct.
Label is IGNORED — only arc direction counts.

Formula: `# correct heads / total words`

LAS — Labeled Attachment Score

What it measures:
% of words where BOTH head AND label are correct.
Most commonly reported metric in papers.

Formula: `# correct (head+label) / total words`

Worked Example – "The dog eats fish" (4 words, excl. ROOT)

Word	Gold Head	Gold Label	Pred Head	Pred Label	UAS ✓?	LAS ✓?
The	dog	det	dog	det	✓	✓
dog	eats	nsubj	eats	nsubj	✓	✓
eats	ROOT	root	ROOT	root	✓	✓
fish	eats	obj	ROOT	dep	✗	✗

Result: UAS = 3/4 = 75% | LAS = 3/4 = 75% (fish was attached to ROOT instead of eats)

UAS vs LAS – Nuances & Real Benchmarks

UAS > LAS always

Getting the head right but the label wrong counts toward UAS but NOT LAS. So UAS ≥ LAS for any parser.

LAS is the standard metric

When papers say "our parser achieves 95.2" they mean LAS. Always check which metric is reported!

Punctuation handling

UD evaluation typically excludes punctuation tokens. Some benchmarks include them — results are not directly comparable.

LA (Label Accuracy)

A third metric: % of correct labels given correct head. Rarely used alone but helps diagnose label classifier separately.

State-of-the-Art LAS Benchmarks (English PTB, UD):

Parser / Model	Approach	LAS (en)	Year
MaltParser	Transition (SVM)	86.9%	2006
TurboParser	Graph-Based (LP)	90.6%	2013
Chen & Manning	Neural Transition (MLP)	91.8%	2014
Dozat & Manning	Biaffine Attention	95.7%	2017
BERT-based parsers	Pre-trained LM + Biaffine	**96.5%+**	2019+