Treebank

- A **Treebank** is a **large collection of sentences**, where **each sentence is manually annotated with its correct syntactic parse tree** by linguists.

Treebank = Corpus + Syntax Trees

It contains:

- sentences
- POS tags
- phrase structure (NP, VP, PP)
- complete parse trees

Example

Sentence: **"The dog chased the cat."**

Treebank stores it like this:

```
scss

(S
  (NP (DT The) (NN dog))
  (VP (VBD chased)
      (NP (DT the) (NN cat)))
)
```

- (VBD chased) → "chased" is a **past-tense verb**
- (NN dog) → "dog" is a **singular noun**
- (NN cat) → "cat" is a **singular noun**
- (DT The / DT the) → "the" is a **determiner**

This is **one entry** of the Treebank.

Why is it called “Treebank”?

- Because it is a **bank (collection)** of **trees (parse trees)**.
- Treebank = **Bank of parse trees**

What is Penn Treebank?

- **Penn Treebank (PTB)** is a **very large, famous Treebank** created by the **University of Pennsylvania (UPenn)**.
  It contains **millions of words** of English text, where each sentence is annotated with:
- ✓ POS tags
  ✓ Phrase structure (NP, VP, PP...)
  ✓ Syntactic parse trees
  ✓ Function labels (SBJ, OBJ, etc.)
- It is the **most commonly used Treebank** for training and evaluating statistical parsers.

HOW TREEBANKS ARE CONSTRUCTED

- A Treebank is built through **manual linguistic annotation + some automatic tools**.
- Example:-
- The dog chased the cat.

□ STEP 1 — Collect Raw Sentences

Treebanks begin with collecting text from:

- newspapers
- books
- articles
- spoken language transcripts

Example source: Wall Street Journal (for Penn Treebank)

Our example sentence comes from the raw text collection.

□ STEP 2 — Tokenization

The text is split into **tokens (words)**.

Example:

The | dog | chased | the | cat | .

This makes it easier to annotate.

□ STEP 3 — POS Tagging (Manual or Automatic)

Each word gets a **Part-of-Speech** label.

Example:

Word	POS
The	DT
dog	NN
chased	VBD
the	DT
cat	NN

Sometimes this is done by:

- Annotator 1
- Then checked by Annotator 2

This ensures correctness.

□ STEP 4 — Add Phrase Structure (Human Linguists)

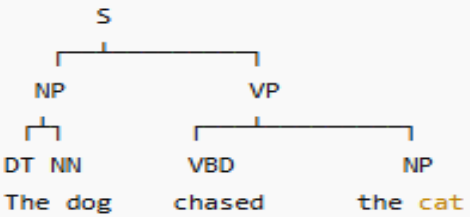
Linguists manually draw the syntactic structure: NP, VP, PP etc.

For our sentence:

- "The dog" = NP
- "chased the cat" = VP
- "the cat" = NP

The expert creates a **tree**:

bash Copy code



This is called **phrase-structure annotation**.

□ STEP 5 — Add Function Tags (optional)

Some Treebanks add extra labels:

- NP-SBJ = subject
- NP-OBJ = object

Example:

scss Copy code

```
(S
  (NP-SBJ (DT The) (NN dog))
  (VP (VBD chased)
      (NP-OBJ (DT the) (NN cat)))
)
```

These help parsers understand grammatical roles.

□ STEP 6 — Double Annotation + Agreement

Two linguists annotate the same sentence:

- Annotator A creates tree
- Annotator B reviews
- Both compare
- A senior linguist resolves disagreements

This ensures **very high accuracy** of the final Treebank.

□ STEP 7 — Store in Standardized Format

The final annotated sentence is stored in formats like:

- Penn Treebank format (bracketed tree)
- XML format
- JSON tree format

Our example Treebank entry becomes:

scss Copy code

```
(S
  (NP (DT The) (NN dog))
  (VP (VBD chased)
      (NP (DT the) (NN cat)))
)
```

This is now an **official Treebank record**.

What is Syntactic Parsing

- **Syntactic parsing** is the process of analyzing a sentence to identify its **grammatical structure** and building a **parse tree** that shows how words group into phrases like NP (noun phrase), VP (verb phrase), PP (prepositional phrase), etc.

Syntactic parsing = finding the structure of a sentence (who did what) using grammar rules.

Why do we need syntactic parsing?

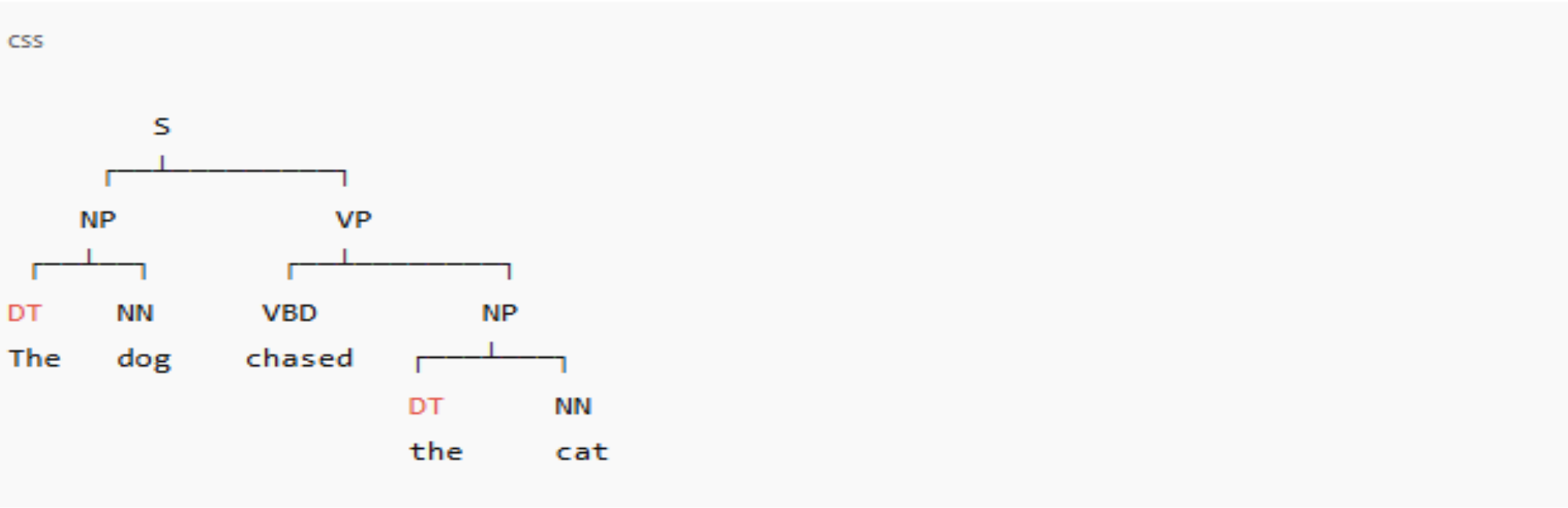
computer cannot understand a sentence unless it knows:

- the subject
- the verb
- the object
- the modifiers
- the relationships between words
- Syntactic parsing gives this structure.

Example

“The dog chased the cat.”

Syntactic parsing produces this tree:



This tells the computer:

- NP (The dog) = subject
- VP (chased the cat) = predicate
- NP (the cat) = object

Types of Syntactic Parsing

1. Constituency Parsing

- Creates phrase structure trees (NP, VP, PP).
  Used in PCFG, Collins parser, Charniak parser.

2. Dependency Parsing

- Finds word-to-word relations.