Minor Project

Seventh Semester

Project Title

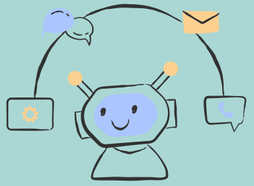
The CYK Algorithm: A Grammatical Detective for Natural Language

Building a Personal Grammar Checker

Why this project?

What is Natural Language Processing (NLP)?

NLP is a field of artificial intelligence that focuses on the interactions between computers and human language. Its goal is to enable computers to understand, interpret, and generate human language in a way that is both meaningful and useful.



: The bots that answer your questions on websites use NLP to understand what you're asking.

Tools like Grammarly or the ones built into word processors analyze your text for grammatical errors and style improvement

When you type a query into a search engine, NLP helps it understand your intent and find the most relevant results, even if your phrasing is imperfect.

The Problem of Structure

- Before a computer can truly understand language, it has to break it down. It must figure out the grammatical structure—how the words relate to one another.

Consider this simple sentence:

"The dog ran."

How do we know that **"the"** and **"dog"** form a single unit (a noun phrase), and that **"ran"** is the action performed by that unit?

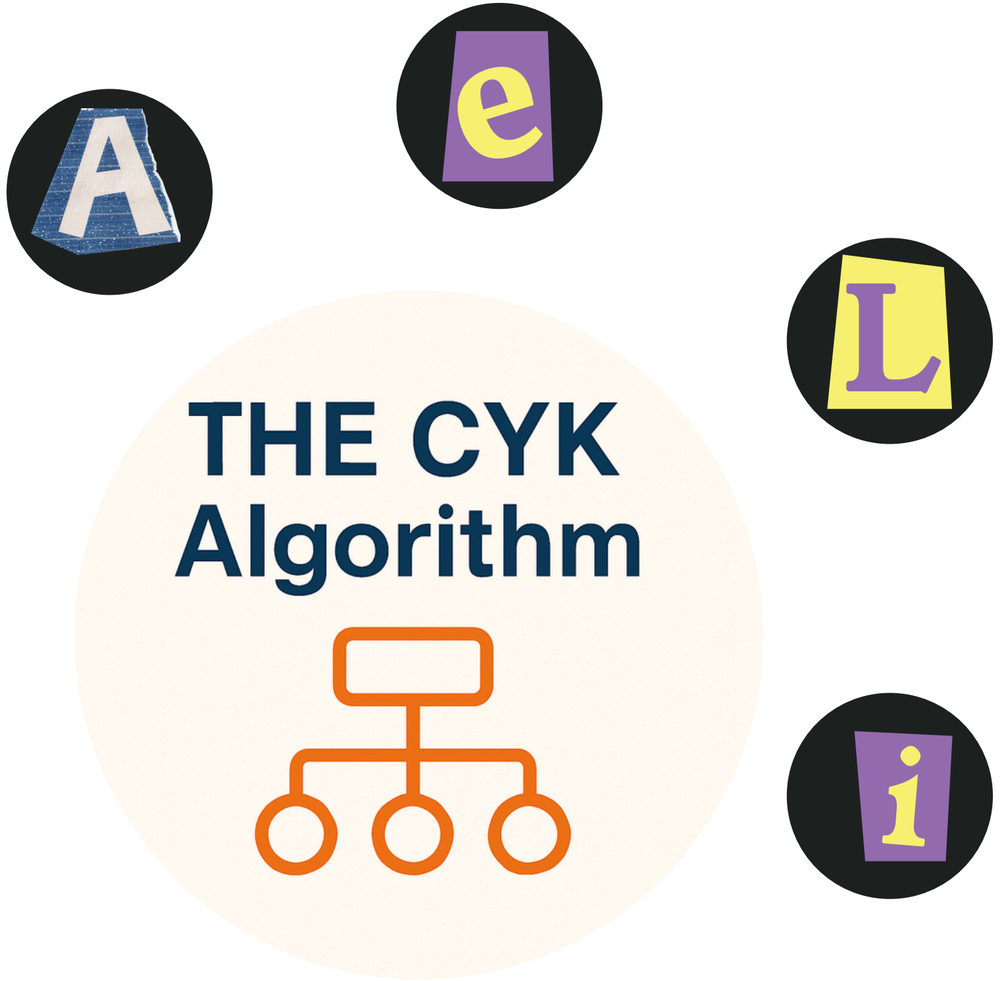
This is the central challenge that a process called parsing solves. It's the first step in giving a computer the ability to see the "blueprint" of a sentence. Without this, the computer sees only a string of individual words, not a coherent thought.

Solving the Problem Using CYK

The CYK Algorithm

The CYK Algorithm is a powerful and efficient parser for analyzing sentence structure.

1. Acts as a "grammatical detective" for sentences.
2. Systematically checks if a sentence fits a specific set of rules.
3. Provides a simple verdict: Is the sentence grammatically valid?
4. If valid, it also reveals the complete structural blueprint of the sentence.



The Grammar Rulebook

The CYK algorithm needs a "rulebook," known as a Context-Free Grammar (CFG), to know what valid sentences look like.

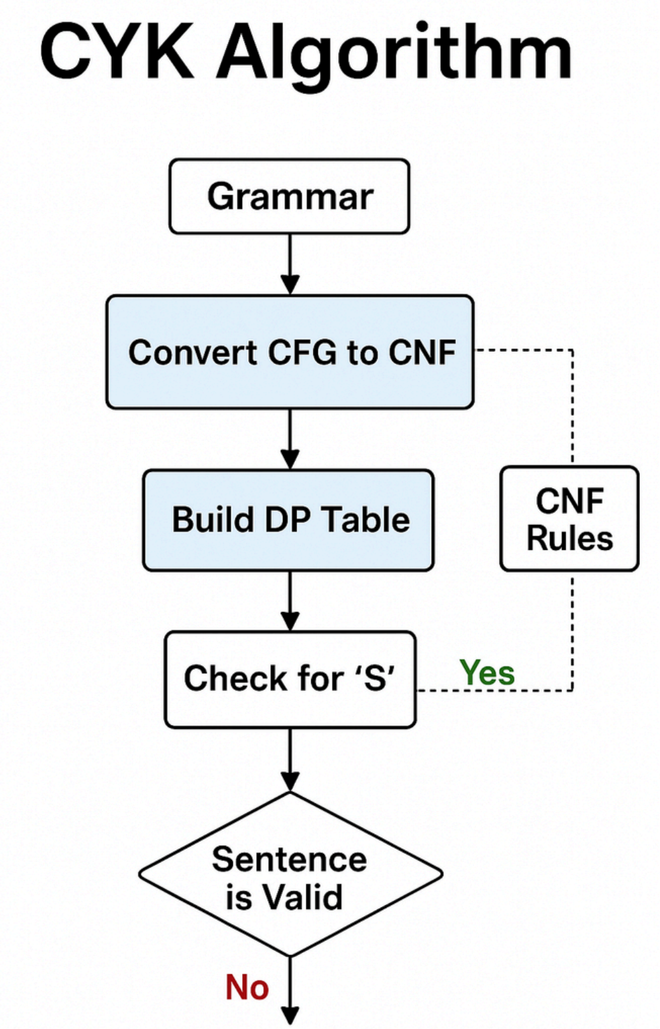
Context-Free Grammar (CFG) is a set of rules that describes how to build sentences from smaller parts. It's like a recipe for a language. CFGs are made of two main types of symbols:

- **Non-terminals**: These are abstract concepts or parts of a sentence, like 'S' for Sentence, 'NP' for Noun Phrase, or 'VP' for Verb Phrase. Other symbols must replace them according to the rules.

- **Terminals**: These are the actual words or symbols in the sentence, like "the," "dog," or "ran." They are the final building blocks.
  Simple Rule Examples:
  S→NP VP (A sentence is a noun phrase followed by a verb phrase.)
  NP→Det N (A noun phrase can be a determiner and a noun.)
  Det→the (The word "the" is a determiner.)
  N→dog (The word "dog" is a noun.)
  V→ran (The word "ran" is a verb.)

Key Ideas Behind CYK

- **Grammar Rules:** Define how words combine into phrases.
  - **CNF Restriction:** Rules must be of form:
    1. A → BC (two non-terminals)
    2. A → a (a single terminal/word)
  - **Bottom-Up Parsing:** Start with words → build phrases → build sentence.
  - **Parse Tree:** Shows how words group together.

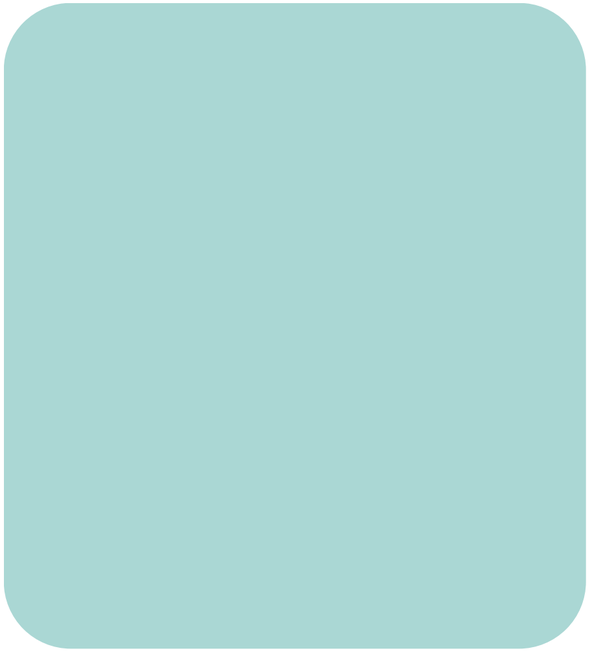


The CYK Algorithm Table in Action

Let's use our grammar to test the sentence:

"The dog ran."

1. Start with the Words: We first fill the bottom row of our table, assigning the non-terminal symbols to each word based on our grammar rules.
   - "the" → Det
   - "dog" → N
   - "ran" → V
2. Combine & Conquer: We then work our way up the table. For each cell, we check if the words in that span can be formed by combining two smaller non-terminals from the rows below.
   - Det + N → NP (Noun Phrase)
   - NP + V → S (Sentence)



The Final Verdict:
We continue this process until we reach the top of the table. If the S (Sentence) symbol is in the final cell, it means the entire sentence is grammatically correct according to our rulebook. In this case, since the S is in the top cell, our sentence is valid!

PARSE TREE

The parse tree is a visual representation of how the words group together to form the grammatical structure of the sentence. It's essentially the "blueprint" we found by filling out the CYK table.

The Parse Tree for "The dog ran"

The parse tree shows the relationships between the non-terminal and terminal symbols. We start from the top with the S symbol and work our way down to the individual words.

- The top node, S (Sentence), is the root of the tree.
- The S node has two child nodes: NP (Noun Phrase) and VP (Verb Phrase), showing that a sentence is composed of these two parts.
- The NP node has two children: Det (Determiner) and N (Noun), which correspond to the words "The" and "dog."
- The VP node has one child: V (Verb), which corresponds to the word "ran."

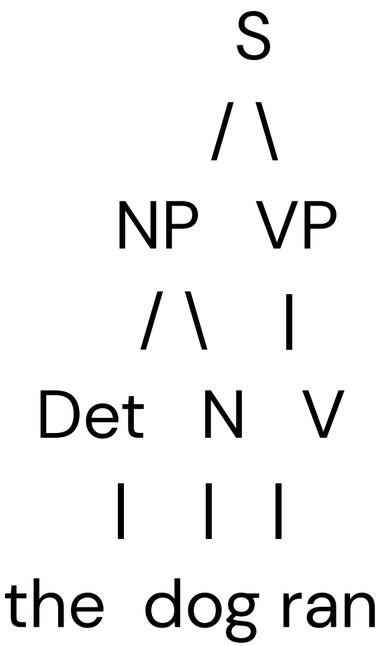
Take some examples and generate the rule table and parse treee

"The dog ran."

- **CYK Table**

Word	Column 1	Column 2	Column 3
the	Det	NP	S
dog	N	-	-
ran	V	-	-

- **Parse Tree**



Valid sentence : "the cat chased a dog"

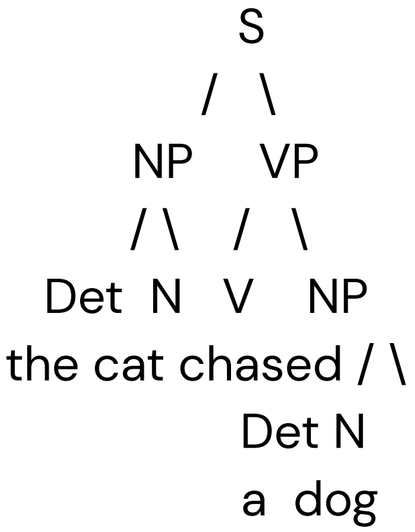
- First, we need a grammar (CFG) with rules to parse this sentence.

S → NP VP
NP → Det N
VP → V NP
Det → "the" | "a"
N → "cat" | "dog"
V → "chased"

- **CYK Table**

word	col1		col 2	col 3	col 4	col 5
the	Det		NP	-	-	S
cat	N		-	-	-	-
chased	V		-	VP	-	---
a	Det		NP	-	-	-
dog	N		-	-	-	-

- **Parse Tree**



- **Invalid sentence:"cat the chased dog"**

word	col1	col2	col3	col4
cat	N	-	-	-
the	Det	-	-	-
chased	V	-	-	-
dog	N	-	-	-

- **Why it fails:**

- Two-word spans needed for NP are Det + N (e.g., the cat), but here the first two tokens are N + Det (cat the), which does not match any A → B C rule.

- Final/top cell does not contain S, so the sentence is invalid under this grammar.

Demo Application & Live Simulator

- Link :CYK-Algorithm-Simulation

Code Repositories

- GitHub links

Thank you very much!

Presented by:

Saloni Kumari
Shashi Kant
Divyannshi Singh
Ketan Khandekar