CKY Parser Implementation

Overview

This project implements a CKY (Cocke-Kasami-Younger) parser in Python to parse sentences based on a given context-free grammar (CFG) in Chomsky Normal Form (CNF). The parser checks whether a sentence can be generated by the grammar and outputs the parsing table (CKY table) along with the parsing result to `output.txt`.

The project includes:

1. **Grammar File (`grammar.txt`)**: Contains grammar rules in CNF.
2. **Python Code (`cky_parser.py`)**: Implements the CKY algorithm to parse a sentence and generate the CKY table.
3. **Output File (`output.txt`)**: Stores the CKY table and parsing result for the provided input sentence.

Methodology

1. Understanding Chomsky Normal Form (CNF)

To use the CKY algorithm, the grammar must be in CNF, where each production rule is in one of the following formats:

- **Unary Rule**: A non-terminal produces a terminal (e.g., `Det -> the`).
- **Binary Rule**: A non-terminal produces exactly two non-terminals (e.g., `S -> NP VP`).

2. CKY Algorithm Overview

The CKY algorithm builds a table that represents possible non-terminals that can generate each substring of the input sentence. Each cell in this table corresponds to a substring and stores possible non-terminals that can produce that substring.

3. Project Workflow

1. **Grammar Input**: Grammar rules are read from an external file (`grammar.txt`) and stored in dictionaries for efficient lookups.
2. **Table Initialization**: An empty CKY table (upper triangular matrix) is initialized for parsing the sentence.
3. **Filling the Diagonal**: Each word in the sentence is checked against unary rules in the grammar to fill the table's diagonal cells.
4. **Filling the Upper Triangular Table**:
    - For substrings of increasing length, each cell is populated by combining pairs of non-terminals from previous cells using binary rules in the grammar.
5. **Parsing Result**:

- If the start symbol (e.g., `S`) is found in the top-right cell of the CKY table, the sentence can be derived from the grammar.

4. Writing to Output

The program writes the generated CKY table and parsing result to `output.txt` for easy reference and verification.

Code Explanation

`read_grammar(filename)`

This function reads the grammar from `grammar.txt`, parsing each line to separate unary and binary rules. These rules are stored in dictionaries (`unary_rules` and `binary_rules`) for quick lookups when filling the CKY table.

`cky_parse(sentence, unary_rules, binary_rules, start_symbol='S')`

This function performs the core CKY parsing algorithm:

- **Input**:
  - **sentence**: The sentence to be parsed.
  - **unary_rules** and **binary_rules**: Dictionaries containing the grammar rules.
  - **start_symbol**: The initial non-terminal (default `'S'`).
- **Output**:
  - **table**: The CKY table, represented as a 2D list of sets, where each set stores non-terminals that can produce specific substrings.
  - **is_parsed**: A boolean value indicating if the sentence was successfully parsed.

`display_table(table, file)`

This function writes the CKY table to the output file, formatting each cell to show non-terminals that produce the corresponding substrings.

`output.txt`

This file contains the CKY table and the parsing result, indicating whether the input sentence is derivable from the grammar.

Running the Code

Prerequisites

- Python 3 installed

Steps

1. Save grammar rules in `grammar.txt` in the same directory.

2. Run the Python script `cky_parser.py`:

   ```
   python cky_parser.py
   ```

3. Open `output.txt` to see the CKY table and parsing result.

Example

Given the sentence "she eats the fish" and the following rules in `grammar.txt`:

```
S -> NP VP
VP -> V NP
VP -> eats
NP -> Det N
NP -> she
Det -> the
N -> fish
V -> eats
```

The content of `output.txt` should look like:

```
CKY Table:
[{NP}, {VP, V}, {}, {S}]
[{}, {VP, V}, {NP}, {VP}]
[{}, {}, {Det}, {NP}]
[{}, {}, {}, {N}]

The sentence 'she eats the fish' is successfully parsed!
```

Conclusion

This project provides a clear implementation of the CKY algorithm for parsing sentences with context-free grammars in CNF. The CKY table and parsing results show whether a sentence is valid according to the grammar, and the method can be adapted to other sentences or grammars with minor modifications. This parsing method is foundational for syntactic analysis in natural language processing and computational linguistics.

Future Enhancements

Potential future improvements include:

- Adding error handling for grammars that aren't in CNF.
- Extending the parser to support probabilistic CFGs (PCFGs).
- Adding visualization tools for more intuitive CKY table interpretation.
