Tecnológico de Monterrey

Evidence 2

Luis Gabriel Delfín Paulín

A01701482

29/04/2024

Description

The language I chose was german. Since it is a really huge language with too many options of grammar I decided to divided into small sentences with the following order:

1. Article

2. Noun

3. Adjective (optional)

4. Verb

This grammar consists defining how sentences (S), noun phrases (NP), determiners (DET), adjectives (ADJ), nouns (N), and verbs (V) can be constructed.

Models

This is the model we are using to define our sentences (S).

S -> NP V '.'

NP -> ART ADJ N | ART N

ART -> 'Der' | 'Die' | 'Ein' | 'Eine'

ADJ -> 'großer' | 'kleiner' | 'schöner' | 'große' | 'kleine' | 'schöne'

N -> 'Mann' | 'Frau' | 'Hund' | 'Katze'

V -> 'geht' | 'singt' | 'läuft'

We are not using left recursion and the vocabulary is restricted. If we wish to add more words to this, we would need to add them manually to the code.

Download

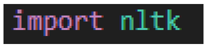
If you wish to download and run the code you will need the following:

Python: Make sure you have Python installed on your system.

NLTK Library: The code utilizes the Natural Language Toolkit (NLTK) library for Python. You can install NLTK using *pip install nltk* in your terminal or cmd.

Code Explanation

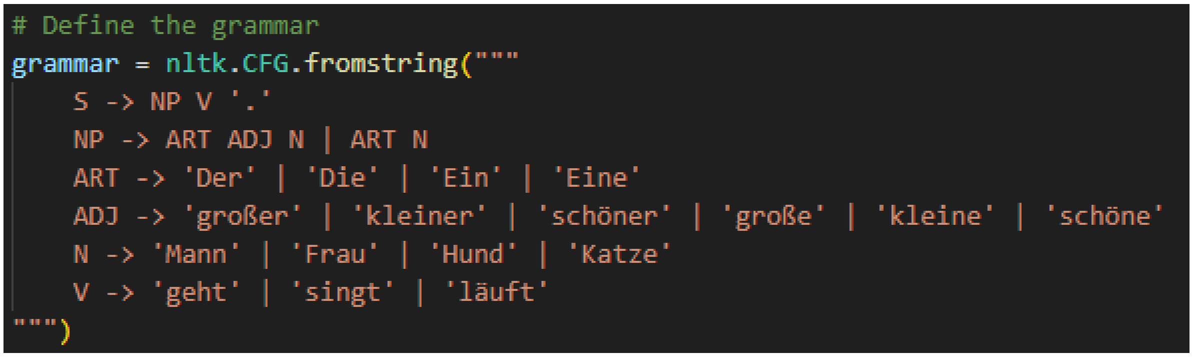
Library



```python
import nltk
```

The library we will be using is NLTK, which allow us to use grammar definition , tokenization (splits the characters), and parsing (checks tokens with the grammar rules), which are used in the code.

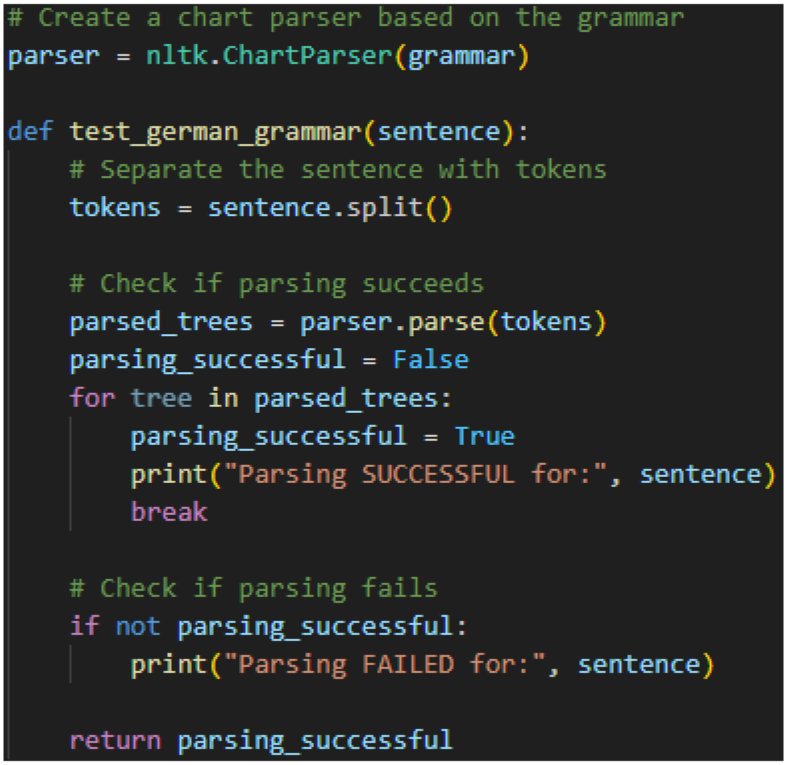
Grammar



```python
# Define the grammar
grammar = nltk.CFG.fromstring("""
    S -> NP V '.'
    NP -> ART ADJ N | ART N
    ART -> 'Der' | 'Die' | 'Ein' | 'Eine'
    ADJ -> 'großer' | 'kleiner' | 'schöner' | 'große' | 'kleine' | 'schöne'
    N -> 'Mann' | 'Frau' | 'Hund' | 'Katze'
    V -> 'geht' | 'singt' | 'läuft'
""")
```

Here is where we define the grammar mentioned previously.

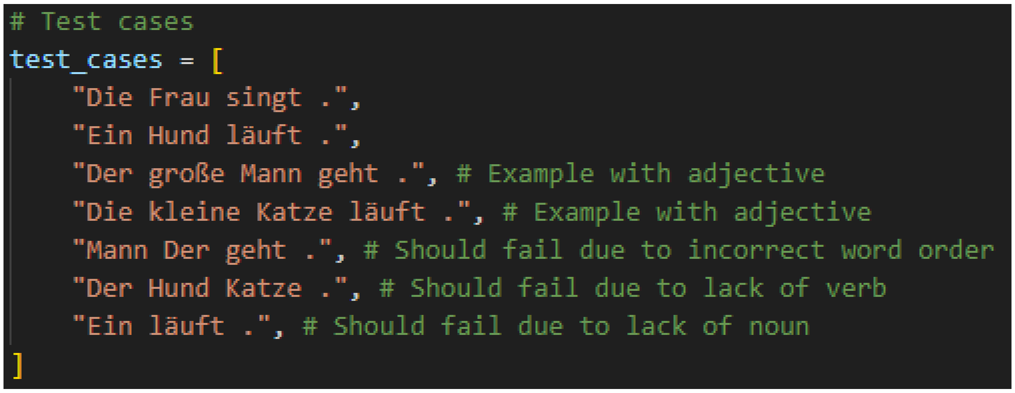
Tokens and parsing



```python
# Create a chart parser based on the grammar
parser = nltk.ChartParser(grammar)

def test_german_grammar(sentence):
    # Separate the sentence with tokens
    tokens = sentence.split()

    # Check if parsing succeeds
    parsed_trees = parser.parse(tokens)
    parsing_successful = False
    for tree in parsed_trees:
        parsing_successful = True
        print("Parsing SUCCESSFUL for:", sentence)
        break

    # Check if parsing fails
    if not parsing_successful:
        print("Parsing FAILED for:", sentence)

    return parsing_successful
```

In here we create a chart parser based on the grammar and defines a function to test German sentence parsing. The function tokenizes the input sentence, attempts parsing, and prints whether parsing succeeded or failed.

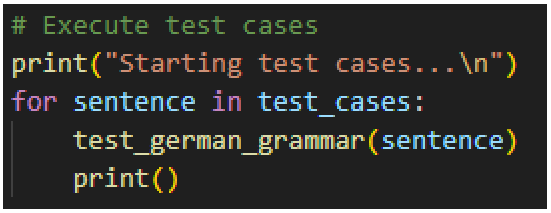
Test cases



```
# Test cases
test_cases = [
    "Die Frau singt .",
    "Ein Hund läuft .",
    "Der große Mann geht .", # Example with adjective
    "Die kleine Katze läuft .", # Example with adjective
    "Mann Der geht .", # Should fail due to incorrect word order
    "Der Hund Katze .", # Should fail due to lack of verb
    "Ein läuft .", # Should fail due to lack of noun
]
```

In here we define our test cases, which the first 4 should be successful and the last 3 should fail because it is not following the grammar we defined above.

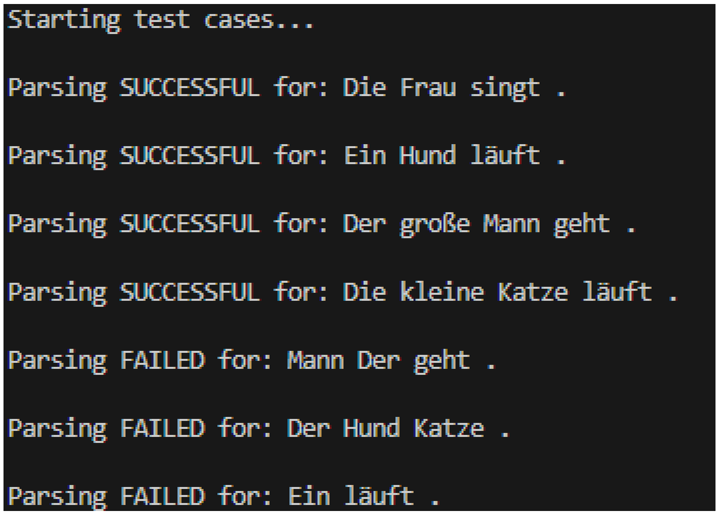
Print solution



```
# Execute test cases
print("Starting test cases...\n")
for sentence in test_cases:
    test_german_grammar(sentence)
    print()
```

Finally we call our function for each of the test cases mentioned above and prints our solution.

Execution



```
Starting test cases...

Parsing SUCCESSFUL for: Die Frau singt .

Parsing SUCCESSFUL for: Ein Hund läuft .

Parsing SUCCESSFUL for: Der große Mann geht .

Parsing SUCCESSFUL for: Die kleine Katze läuft .

Parsing FAILED for: Mann Der geht .

Parsing FAILED for: Der Hund Katze .

Parsing FAILED for: Ein läuft .
```

As we can see, the first 4 test cases are successful and the last 3 failed, so the solution provided is successful.