Experiment No. 6
Perform Chunking for the given text input.
Date of Performance:
Date of Submission:

Exp. No.: 6

Title: Perform Chunking for the given text input.

Theory:

Chunking in NLP is a process to take small pieces of information and group them into large units. The primary use of Chunking is making groups of "noun phrases." It is used to add structure to the sentence by following POS tagging combined with regular expressions. The resulting group of words are called "chunks." It is also called shallow parsing.

In shallow parsing, there is maximum one level between roots and leaves while deep parsing comprises more than one level. Shallow parsing is also called light parsing or chunking.

Rules for Chunking:

There are no predefined rules, but you can combine them according to need and requirement.

For example, you need to tag Noun, verb (past tense), adjective, and coordinating junction from the sentence. You can use the rule as below

chunk: {<NN.?>*<VBD.?>*<JJ.?>*<CC>?}

Code:

```
# Laading Library
from nltk.chunk.regexp import tag_pattern2re_pattern

# Chunk Pattern to RegEx Pattern
print("Chunk Pattern : ", tag_pattern2re_pattern('<DT>?<NN.*>+'))
```

```
Chunk Pattern :  (<(DT)>)?(<(NN[^\{\}<>]*)>)+
```

```
!pip install svgling

from nltk.chunk import RegexpParser
```

```
# Introducing the Pattern
chunker = RegexpParser(r'''
    NP: {<DT><NN.*>+}
    VP: {<VB.*>.*}
''')
```

```
# Sample POS-tagged input
```

```
pos_tags = [('the', 'DT'), ('book', 'NN'), ('has', 'VBZ'), ('many', 'JJ'),
('chapters', 'NNS')]
```

```
# Parsing the POS-tagged input
result = chunker.parse(pos_tags)
```

```
# Displaying the result
print(result)
```

```
Collecting svgling
  Downloading svgling-0.5.0-py3-none-any.whl.metadata (7.4 kB)
Collecting svgwrite (from svgling)
  Downloading svgwrite-1.4.3-py3-none-any.whl.metadata (8.8 kB)
Downloading svgling-0.5.0-py3-none-any.whl (31 kB)
Downloading svgwrite-1.4.3-py3-none-any.whl (67 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.1/67.1 kB 3.6 MB/s eta 0:00:00
Installing collected packages: svgwrite, svgling
Successfully installed svgling-0.5.0 svgwrite-1.4.3
(S (NP the/DT book/NN) (VP has/VBZ) many/JJ chapters/NNS)
```

```
!pip install svgling

import nltk
from nltk import Tree
from nltk import pos_tag, word_tokenize
from nltk.parse import CoreNLPParser
```

```
# Ensure required resources are downloaded
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
# Define a simple grammar for parsing
grammar = '''
  S -> NP VP
  NP -> DT NN | NP JJ NN
  VP -> VBZ NP | VP JJ
  DT -> 'the'
  NN -> 'book' | 'chapters'
  VBZ -> 'has'
  JJ -> 'many'
'''
```

```
# Create a parser
parser = nltk.ChartParser(nltk.CFG.fromstring(grammar))
```

```
# Sample sentence
sentence = "the book has many chapters"
```

```
# Tokenizing the sentence
tokens = word_tokenize(sentence)
```

```
# Parse the sentence
for tree in parser.parse(tokens):
    print(tree)
    tree.pretty_print()  # Display the parse tree in text format
```

```
    # Visualize the tree using svgling (if you want a graphical representation)
    try:
        from svgling import draw_tree
        draw_tree(tree).show()
```

```
    except Exception as e:
        print("Visualization error:", e)
```

```
Requirement already satisfied: svgling in /usr/local/lib/python3.10/dist-packages (0.5.0)
Requirement already satisfied: svgwrite in /usr/local/lib/python3.10/dist-packages (from svgling) (1.4.3)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Conclusion: