SATHYABAMA INSTITUTE OF SCIENCE & TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCSA 2604 NATURAL LANGUAGE PROCESSING LAB

LAB 7: CHUNKING

AIM: To perform Noun Phrase chunking

PROCEDURE:

In Natural Language Processing (NLP), chunking is the process of extracting short, meaningful phrases (chunks) from a sentence based on specific patterns of parts of speech (POS). Python provides tools like NLTK (Natural Language Toolkit) to perform chunking. This example demonstrates a basic noun phrase (NP) and verb phrase (VP) chunking using NLTK. You can adjust the chunk grammar patterns to capture different types of phrases or entities based on your specific needs.

The chunk_grammar variable contains patterns defined using regular expressions for identifying noun phrases and verb phrases. Adjusting these patterns can help extract different types of chunks like prepositional phrases, named entities, etc.

Tokenization: Breaking the sentence into individual tokens or words.

POS Tagging: Assigning part-of-speech tags to each token (identifying whether it's a noun, verb, adjective, etc.).

Chunking: Grouping tokens into larger structures (noun phrases, verb phrases) based on defined grammar rules.

Chunk Grammar: Regular expressions defining patterns for identifying specific chunk structures (like noun phrases).

Chunk Parser: Utilizing the chunk grammar to parse and extract chunks based on the provided POS-tagged tokens.

The following algorithm outlines the steps involved in the noun phrase chunking process using NLTK in Python, highlighting the key processes and the role of chunk grammar in identifying and extracting specific syntactic structures from text data.

ALGORITHM:

1. Import Necessary Libraries: Import required modules from NLTK for tokenization, POS tagging, and chunking.

2. Download NLTK Resources (if needed): Ensure NLTK resources like tokenizers and POS taggers are downloaded (nltk.download('punkt'), nltk.download('averaged_perceptron_tagger')).

3. Define a Sample Sentence: Set a sample sentence that will be used for chunking.

4. Tokenization: Break the sentence into individual words or tokens using NLTK's word_tokenize() function.

5. Part-of-Speech (POS) Tagging: Tag each token with its corresponding part-of-speech using NLTK's pos_tag() function.

6. Chunk Grammar Definition: Define a chunk grammar using regular expressions to identify noun phrases (NP). For example, NP: {<DT>?<JJ>*<NN>} captures sequences with optional determiners (DT), adjectives (JJ), and nouns (NN) as noun phrases.

7. Chunk Parser Creation: Create a chunk parser using RegexpParser() and provide the defined chunk grammar.

8. Chunking: Parse the tagged sentence using the created chunk parser to extract chunks based on the defined grammar.

9. Display Chunks: Iterate through the parsed chunks and print the subtrees labeled as 'NP', which represent the identified noun phrases.

PROGRAM:

```
!pip install nltk
import nltk
from nltk import RegexpParser
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag

# Download NLTK resources (run only once if not downloaded)
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

# Sample sentence
sentence = "The quick brown fox jumps over the lazy dog"

# Tokenize the sentence
tokens = word_tokenize(sentence)

# POS tagging
tagged = pos_tag(tokens)

# Define a chunk grammar using regular expressions
# NP (noun phrase) chunking: "NP: {<DT>?<JJ>*<NN>}"
# This grammar captures optional determiner (DT), adjectives (JJ), and nouns (NN) as a noun
phrase
chunk_grammar = r"""
   NP: {<DT>?<JJ>*<NN>}
"""

# Create a chunk parser with the defined grammar
chunk_parser = RegexpParser(chunk_grammar)
```

```
# Parse the tagged sentence to extract chunks
chunks = chunk_parser.parse(tagged)


# Display the chunks
for subtree in chunks.subtrees():
    if subtree.label() == 'NP':  # Print only noun phrases
        print(subtree)
```

OUTPUT:

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk)
(8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk)
(1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages
(from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk)
(4.66.1)
(NP The/DT quick/JJ brown/NN)
(NP fox/NN)
(NP the/DT lazy/JJ dog/NN)
[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!
```