Name: Bodhisatya Ghosh
UID: 2021700026
Branch: CSE DS
Experiment: 7
Batch: A

Aim : Chunking and Named Entity Recognition

Theory:

- Q1. What is chunking?

-> Chunking is a process of extracting phrases from unstructured text, which means analysing a sentence to identify the constituents(Noun Groups, Verbs, verb groups, etc.) However, it does not specify their internal structure, nor their role in the main sentence. It works on top of POS tagging. It uses POS-tags as input and provides chunks as output. In short, Chunking means grouping of words/tokens into chunks. Chunking can break sentences into phrases that are more useful than individual words and yield meaningful results. Chunking is very important when you want to extract information from text such as locations, person names. Groups of words make up phrases and there are five major categories. • Noun Phrase(NP) • Verb phrase (VP) • Adjective phrase (ADJP) • Adverb phrase (ADVP) • Prepositional phrase (PP) For example, the sentence 'He reckons the current account deficit will narrow to only 1.8 billion in September.' can be divided as follows: [NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only 1.8 billion] [PP in] [NP September]

- Q2. Explain Named Entity Recognition.

-> Named Entity Recognition (NER) is a natural language processing (NLP) task that involves identifying and classifying named entities within a body of text. Named entities are typically real-world objects that have names, such as persons, organizations, locations, dates, numerical expressions, and more. The goal of Named Entity Recognition is to locate and classify these entities into predefined categories. Named Entity Recognition is a crucial component in various NLP applications, including information extraction, question answering, document summarization, and more. It helps in understanding the structure and semantics of text data by identifying important entities and their relationships.

In [ ]:
```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.tree import Tree
from tabulate import tabulate
from nltk.chunk import ne_chunk
```

In [ ]:
```
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\Rommel\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\Rommel\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
```

Out[ ]: True

In [ ]:
```
test_sentence = "A natural language processing program always includes cleaning the Data before moving on. With John Grinder,
```

In [ ]:
```
sentences = sent_tokenize(test_sentence)
```

In [ ]:
```
grammar = r"""
        NP:{<DT|JJ|NN.*>+}
        PP:{<IN><NP>}
        VP:{<VB.*><NP|PP|CLAUSE>+$}
        ADJP:{<JJ.*><PP|CLAUSE>*}
        ADVP:{<RB.*><PP|CLAUSE>*}
        CLAUSE:{<NP><VP>}
"""
```

In [ ]:
```
def chunk_text(text):
    tokenized = nltk.word_tokenize(text)
    tagged = nltk.pos_tag(tokenized)
    chunk_parser = nltk.RegexpParser(grammar)
    parsed = chunk_parser.parse(tagged)
    return parsed
```

In [ ]:
```python
def extract_chunks(parsed_sentence):
    chunks = []
    for subtree in parsed_sentence:
        if type(subtree) == Tree:
            label = subtree.label()
            phrase = " ".join([word for word, tag in subtree.leaves()])
            chunks.append((label, phrase))
    return chunks
```

In [ ]:
```python
table_data = []
for sentence in sentences:
    parsed_sentence = chunk_text(sentence)
    chunks = extract_chunks(parsed_sentence)
    # print(parsed_sentence)
    if chunks:
        table_data.extend(chunks)

print(tabulate(table_data, headers=["Chunk type","Chunk Phrase"], tablefmt="grid"))
```

```
+--------------+---------------------------------------+
| Chunk type   | Chunk Phrase                          |
+==============+=======================================+
| NP           | A natural language processing program |
+--------------+---------------------------------------+
| ADVP         | always                                |
+--------------+---------------------------------------+
| NP           | the Data                              |
+--------------+---------------------------------------+
| PP           | With John Grinder                     |
+--------------+---------------------------------------+
| NP           | the neuro-linguistic programming      |
+--------------+---------------------------------------+
| NP           | NLP                                   |
+--------------+---------------------------------------+
| NP           | approach                              |
+--------------+---------------------------------------+
| PP           | in the                                |
+--------------+---------------------------------------+
```

In [ ]:
```
words  = word_tokenize(test_sentence)
tagged_words = nltk.pos_tag(words)
named_entities = ne_chunk(tagged_words)
for entity in named_entities:
    if isinstance(entity, nltk.tree.Tree):
        label = entity.label()
        phrase = " ".join(word for word, tag in entity.leaves())
        print(f"Named entity: {phrase}, Type: {label}")
```

```
Named entity: Data, Type: ORGANIZATION
Named entity: John Grinder, Type: PERSON
Named entity: NLP, Type: ORGANIZATION
```

Conclusion:

The experiment helps in understanding the use of chunking and how to apply it. I have also learnt how to identify Named Entities in the sentence.