Name:	Pranay Singhvi
UID:	2021300126

Experiment 7

Aim:	Text Analysis: Exploring Chunking and Named Entity Recognition for Information Extraction from Text Corpora
Theory:	• **Chunking** ▪ **Purpose:** Groups words into meaningful phrases (chunks). It's a step above part-of-speech (POS) tagging but less complex than full parsing. ▪ **Types:** ◦ **Noun Phrase (NP):** A phrase built around a noun (e.g., "the big dog") ◦ **Verb Phrase (VP):** A phrase built around a verb (e.g., "is running quickly") ◦ **Adjective Phrase (ADJP):** A phrase built around an adjective (e.g., "very happy") ◦ **Adverb Phrase (ADVP):** A phrase built around an adverb (e.g., "quite slowly") ◦ **Prepositional Phrase (PP):** A phrase starting with a preposition and including its object (e.g., "on the table") • **Named Entity Recognition (NER)** ▪ **Purpose:** Locates and classifies named entities in text into predefined categories. ▪ **Common NER Types:** ◦ **Persons (PER):** Names of people ◦ **Organizations (ORG):** Companies, government bodies, etc. ◦ **Locations (LOC):** Countries, cities, mountains, etc. ◦ **Dates and Times:** Specific dates, time expressions ◦ **Quantities:** Numerical values, measurements ◦ **Miscellaneous (MISC):** Other entities (products, events) **Methods** **1. Chunking** **a) Regular Expressions** • **Pros:** Flexible if you're familiar with regex. • **Cons:** Regexes can get complex; they might not be as accurate as library-based methods. **b) Libraries (like NLTK)** • **NLTK (Natural Language Toolkit):** - Train a chunk parser for basic chunking. - Use rule-based grammars for more customized patterns. ▪ **Pros:** Generally more accurate and robust than simple regex. ▪ **Cons:** Can require some setup and understanding of NLP libraries. **2. Named Entity Recognition** **a) Rule-Based with Regex** • **Pros:** Simple if you need to target very specific entities. • **Cons:** Limited; less adaptable to variations in entity forms. **b) Libraries (like NLTK or spaCy)** ▪ **NLTK:** Offers a pre-trained NER model (`nltk.ne_chunk()`). ▪ **spaCy:** Popular library with efficient and highly customizable NER models. ▪ **Pros:** Powerful models, pre-trained on large datasets for better accuracy. ▪ **Cons:** Might require more setup and resources.

1. Installation of NLTK and downloading the required corpus

In [ ]:
```
import re
import nltk
import spacy
from prettytable import PrettyTable
```

2. Declare the sample text corpus

In [ ]:
```
# Sample text corpus
text_corpus = """
President Donald Trump visited France and met with French President Emmanuel Macron.
The movie was directed by Steven Spielberg and starred Tom Hanks.
The cat sat lazily on the mat.
"""
```

3. Regular expression patterns for chunking

In [ ]:
```
grammar = r"""
    NP: {<DT|JJ|NN.*>+}          # Chunk sequences of DT, JJ, NN
    VP: {<VB.*><NP|PP|CLAUSE>+$}  # Chunk verbs and their arguments
    ADJP: {<JJ.*><PP>?}           # Chunk adjectives with optional prepositional phrase
    ADVP: {<RB.*>}                 # Chunk adverbs
    PP: {<IN><NP>}                 # Chunk prepositions and their objects
"""
```

4. Chunking using regular expressions

In [ ]:
```
def chunk_with_regex(text):
    sentences = nltk.sent_tokenize(text)
    for sentence in sentences:
        words = nltk.word_tokenize(sentence)
        tagged_words = nltk.pos_tag(words)
        chunk_parser = nltk.RegexpParser(grammar)
        chunks = chunk_parser.parse(tagged_words)
        print(chunks)
```

5. Chunking using NLTK library

In [ ]:
```
def chunk_with_nltk(text):
    sentences = nltk.sent_tokenize(text)
    for sentence in sentences:
        words = nltk.word_tokenize(sentence)
        tagged_words = nltk.pos_tag(words)
        chunk_parser = nltk.RegexpParser(grammar)
        chunks = chunk_parser.parse(tagged_words)
        print(chunks)
```

6. Named Entity Recognition using spaCy library

In [ ]:
```
def ner_with_spacy_pretty_table(text):
    nlp = spacy.load("en_core_web_sm")
    doc = nlp(text)
    table = PrettyTable(["Entity", "Label"])
    for ent in doc.ents:
        table.add_row([ent.text, ent.label_])
    print(table)
```

7. Display the results

In [ ]:
```
# Chunking with regular expressions
print("Chunking with regular expressions:")
chunk_with_regex(text_corpus)
```

```
Chunking with regular expressions:
(S
  (NP President/NNP Donald/NNP Trump/NNP)
  visited/VBD
  (NP France/NNP)
  and/CC
  met/VBN
  (PP with/IN (NP French/NNP President/NNP Emmanuel/NNP Macron/NNP))
  ./.)
(S
  (NP The/DT movie/NN)
  was/VBD
  directed/VBN
  (PP by/IN (NP Steven/NNP Spielberg/NNP))
  and/CC
  starred/VBD
  (NP Tom/NNP Hanks/NNP)
  ./.)
(S
  (NP The/DT cat/NN)
  sat/VBD
  (ADVP lazily/RB)
  (PP on/IN (NP the/DT mat/NN))
  ./.)
```

In [ ]:
```
# Chunking with NLTK library
print("\nChunking with NLTK library:")
chunk_with_nltk(text_corpus)
```

```
Chunking with NLTK library:
(S
  (NP President/NNP Donald/NNP Trump/NNP)
  visited/VBD
  (NP France/NNP)
  and/CC
  met/VBN
  (PP with/IN (NP French/NNP President/NNP Emmanuel/NNP Macron/NNP))
  ./.)
(S
  (NP The/DT movie/NN)
  was/VBD
  directed/VBN
  (PP by/IN (NP Steven/NNP Spielberg/NNP))
  and/CC
  starred/VBD
  (NP Tom/NNP Hanks/NNP)
  ./.)
(S
  (NP The/DT cat/NN)
  sat/VBD
  (ADVP lazily/RB)
  (PP on/IN (NP the/DT mat/NN))
  ./.)
```

In [ ]:
```
# Named Entity Recognition with spaCy library
print("\nNamed Entity Recognition with spaCy:")
ner_with_spacy_pretty_table(text_corpus)
```

```
Named Entity Recognition with spaCy:
+-----------------+--------+
|      Entity     | Label  |
+-----------------+--------+
|   Donald Trump  | PERSON |
|      France     |  GPE   |
|      French     |  NORP  |
| Emmanuel Macron | PERSON |
| Steven Spielberg | PERSON |
|    Tom Hanks    | PERSON |
+-----------------+--------+
```

Conclusion

Through this experiment, I discovered that combining regular expressions and libraries provides the most robust solution for extracting meaningful chunks and named entities from text. While regular expressions offer flexibility, libraries like NLTK and spaCy provide superior accuracy and efficiency for real-world natural language processing tasks.