Name	Hatim Sawai
UID No.	2021300108
Experiment No.	7

Experiment 7

Aim	Perform chunking by analyzing the importance of selecting proper features for training a model and size of training.

1. Installation of NLTK and downloading the required corpus

In [ ]:
```
import re
import warnings
import nltk
import pandas as pd
from nltk import pos_tag, ne_chunk
from nltk.tokenize import word_tokenize
from nltk.chunk import RegexpParser
from prettytable import PrettyTable
warnings.filterwarnings('ignore')
```

In [ ]:
```
# download the necessary nltk data including ne_chunk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download("maxent_ne_chunker")
nltk.download("words")
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\hatim\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\hatim\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\hatim\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping chunkers\maxent_ne_chunker.zip.
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\hatim\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\words.zip.
```

Out[ ]: True

2. Loading the corpus and preprocessing

In [ ]:
```
# Load csv
df = pd.read_csv('../dataset/exp7.csv')
df.head()
```

Out[ ]:

	text
0	The new study reveals shocking statistics abou...
1	The latest iPhone model exceeds expectations w...
2	Customers are raving about the delicious flavo...
3	Political tensions rise as negotiations betwee...
4	Researchers announce a breakthrough in cancer ...

In [ ]:
```
def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^\w\s]', '', text) # remove punctuation
    text = re.sub(r'\W', ' ', text)  # Remove non-word characters
    text = re.sub(r'\s+', ' ', text).strip()  # Remove extra whitespaces
    text = re.sub(r'\d', '', text)  # Remove digits
    return text
```

In [ ]:
```
phrase_mapping = {
    'NP: {<DT>?<JJ>*<NN>}': 'noun phrase',
    'PP: {<IN><NP>}': 'prepositional phrase',
    'VP: {<VB.*><NP|PP|CLAUSE>+$}': 'verb phrase'
}
```

3. Chunking

In [ ]:
```
# Tokenize the text
df['text'] = df['text'].apply(preprocess)
sentences = df['text'].tolist()
```

In [ ]:
```
# Function to perform chunking using regular expressions
def chunk_with_regex(sentence):
    grammar = r"""
        NP: {<DT>?<JJ>*<NN>}   # Chunk sequences of DT, JJ, NN
        PP: {<IN><NP>}         # Chunk prepositions followed by NP
        VP: {<VB.*><NP|PP>*}   # Chunk verbs followed by NP or PP
        ADJP: {<JJ>+}          # Chunk sequences of JJ
        ADVP: {<RB.*>}         # Chunk adverbs
    """
    parser = RegexpParser(grammar)
    parsed_sentence = parser.parse(sentence)
    return parsed_sentence
```

In [ ]:
```
# Function to perform chunking using NLTK library
def chunk_with_nltk(sentence):
    words = word_tokenize(sentence)
    tagged_words = pos_tag(words)
    grammar = r"""
        NP: {<DT>?<JJ>*<NN>}    # Chunk sequences of DT, JJ, NN
        PP: {<IN><NP>}          # Chunk prepositions followed by NP
        VP: {<VB.*><NP|PP>*}    # Chunk verbs followed by NP or PP
        ADJP: {<JJ>+}           # Chunk sequences of JJ
        ADVP: {<RB.*>}          # Chunk adverbs
    """
    parser = nltk.RegexpParser(grammar)
    parsed_sentence = parser.parse(tagged_words)
    return parsed_sentence
```

In [ ]:
```
# Perform chunking using regular expressions
# Initialize PrettyTable
chunk_table = PrettyTable(["Chunk", "Tag"])
print("Chunking using regular expressions")
for sentence in sentences:
    parsed_sentence = chunk_with_regex(pos_tag(word_tokenize(sentence)))
    for subtree in parsed_sentence.subtrees():
        if subtree.label() in ['NP', 'PP', 'VP', 'ADJP', 'ADVP']:
            chunk_table.add_row([" ".join(word for word, tag in subtree.leaves()),

# Print the table
print(chunk_table[:20])
```

```
Chunking using regular expressions
+-------------------------------------------+------+
|                   Chunk                   | Tag  |
+-------------------------------------------+------+
|                the new study              |  NP  |
|                  shocking                 |  VP  |
|                about climate              |  PP  |
|                  climate                  |  NP  |
|                   change                  |  NP  |
|                   iphone                  |  NP  |
|                   model                   |  NP  |
|                  exceeds                  |  VP  |
|                 innovative                | ADJP |
|                    are                    |  VP  |
|                   raving                  |  VP  |
|                 delicious                 | ADJP |
|                  new ice                  |  NP  |
|                   cream                   |  NP  |
|                 political                 | ADJP |
|                    rise                   |  VP  |
|                   stall                   |  VP  |
| announce a breakthrough in cancer treatment |  VP  |
|               a breakthrough              |  NP  |
|                 in cancer                 |  PP  |
+-------------------------------------------+------+
```

In [ ]:

```
# Perform chunking using NLTK library
chunk_table = PrettyTable(["Chunk", "Tag"])
print("\nChunking using NLTK library:")
for sentence in sentences:
    parsed_sentence = chunk_with_nltk(sentence)
    for subtree in parsed_sentence.subtrees():
        if subtree.label() in ['NP', 'PP', 'VP', 'ADJP', 'ADVP']:
            chunk_table.add_row([" ".join(word for word, tag in subtree.leaves()),

print(chunk_table[:20])
```

Chunking using NLTK library:

Chunk	Tag
the new study	NP
shocking	VP
about climate	PP
climate	NP
change	NP
iphone	NP
model	NP
exceeds	VP
innovative	ADJP
are	VP
raving	VP
delicious	ADJP
new ice	NP
cream	NP
political	ADJP
rise	VP
stall	VP
announce a breakthrough in cancer treatment	VP
a breakthrough	NP
in cancer	PP

4. Named Entity Recognition (NER)

In [ ]:
```
df1 = pd.read_csv('../dataset/exp7a.csv')
sentences = df1["text"].tolist()
df1.head()
```

Out[ ]:

	text
0	Barack Obama was the 44th President of the Uni...
1	Apple Inc. is headquartered in Cupertino, Cali...
2	The Eiffel Tower is located in Paris, France.
3	Albert Einstein was a renowned physicist born ...
4	The Beatles were an influential band formed in...

In [ ]:
```
# Define a function for Named Entity Recognition
def ner_with_nltk(sentence):
    tokens = word_tokenize(sentence)
    tagged_tokens = pos_tag(tokens)
    named_entities = ne_chunk(tagged_tokens)

    # Extract named entity labels
    named_entities_labels = []
    for subtree in named_entities:
        if isinstance(subtree, nltk.Tree):
            entity_label = subtree.label()
            named_entities_labels.append(
                (entity_label, " ".join(word for word, tag in subtree.leaves()))
            )
    return named_entities_labels
```

In [ ]:
```
# Perform Named Entity Recognition for each sentence
named_entities = []
for sentence in sentences:
    named_entities.extend(ner_with_nltk(sentence))

# Initialize PrettyTable
named_entity_table = PrettyTable(["Entity", "Phrase Type"])
for entity, phrase in named_entities:
    named_entity_table.add_row([entity, phrase])

# Print the PrettyTable
print("\nNamed Entities:")
print(named_entity_table)
```

Named Entities:

Entity	Phrase Type
PERSON	Barack
PERSON	Obama
GPE	United States
PERSON	Apple
ORGANIZATION	Inc.
GPE	Cupertino
GPE	California
ORGANIZATION	Eiffel Tower
GPE	Paris
GPE	France
PERSON	Albert
PERSON	Einstein
ORGANIZATION	Beatles
GPE	Liverpool
GPE	England
PERSON	Google
PERSON	Larry Page
PERSON	Sergey Brin
ORGANIZATION	Great Wall
GPE	China
GPE	China
PERSON	Nelson
PERSON	Mandela
ORGANIZATION	NASA
GPE	New York City
PERSON	Mount
ORGANIZATION	Everest
PERSON	Elon
ORGANIZATION	Musk
ORGANIZATION	CEO of SpaceX
PERSON	Tesla
ORGANIZATION	Mona Lisa
ORGANIZATION	Louvre Museum
GPE	Paris
GPE	France
GPE	Shakespeare
ORGANIZATION	United Nations
GPE	Leonardo
PERSON	Vinci

6. Curiosity Questions

Q1. How does Named Entity Recognition (NER) contribute to information extraction in natural language processing?

Ans: Named Entity Recognition plays a crucial role in information extraction by identifying and classifying entities such as persons, organizations, locations, dates, and more in unstructured text. This helps in tasks like summarization, question answering, and knowledge graph construction.

Q2. Challenges related to NER in real-life?

Ans: One challenge is ambiguity, where a word can have multiple possible entity types depending on context (e.g., "Paris" could refer to the city or a person's name). Another challenge is handling out-of-vocabulary entities or entities with diverse variations (e.g., person names with different spellings or titles).

Q3. How does chunking differ from Named Entity Recognition, and how are they related??

Ans: Chunking involves grouping adjacent words in a sentence into syntactic phrases, such as noun phrases (NP) or verb phrases (VP), without assigning specific semantic labels. Named Entity Recognition, on the other hand, specifically identifies and classifies named entities in text. While chunking focuses on syntactic structure, NER focuses on semantic meaning. However, NER can be considered a specialized form of chunking that targets named entities.

Q4. What are some potential applications of chunking and Named Entity Recognition?

Ans: Chunking can be applied in tasks such as information extraction, sentiment analysis, and machine translation by identifying and extracting meaningful phrases from text. Named Entity Recognition finds applications in information retrieval, document classification, and entity linking for organizing and retrieving information from large text corpora.

Q5. How do different approaches to Named Entity Recognition and chunking affect the accuracy and performance?

Ans: Various approaches, such as rule-based systems, statistical models, and deep learning methods, can be employed for Named Entity Recognition and chunking. Each approach has its strengths and weaknesses in terms of accuracy, scalability, and generalization to diverse text domains. Evaluating the performance of these approaches on benchmark datasets helps in understanding their effectiveness in real-world applications.

6. Conclusion

In this experiment we learned about the concepts of chunking and Named Entity Recognition (NER) in natural language processing and their applications in information extraction.