B. K. Birla College of Arts, Science & Commerce (Autonomous), Kalyan

(Department of Computer Science)

SEMESTER : II

SUBJECT : NATURAL LANGUAGE PROCESSING

NAME : YUKTA KRISHNA CHAUDHARI

CLASS : M.SC.COMPUTER SCIENCE PART-1

ROLL NO. : 10

B. K. Birla College of Arts, Science & Commerce (Autonomous), Kalyan

(Department of Computer Science)

CERTIFICATE

Miss. **Yukta Krishna Chaudhari.**

Roll No. **10** *Exam Seat No.* _____ *has satisfactorily completed the Practical in* **Natural Language Processing** *as laid down in the regulation of University of Mumbai for the purpose of MSc Computer Science* **Semester-II (Practical)** *Examination* **2022-2023.**

Date:

Place: **Kalyan**

_____

Head
Department of Computer Science

_____

Professor In-Charge
Computer Science

Signature of Examiners

1) _____

2) _____

INDEX

Practical No 1

Aim:- Write a program to implement sentence segmentation and word tokenization

Background Information:-

Tokenization can be done to either separate word or sentences.

Sentence segmentation:-

- Sentence tokenization (also called sentence segmentation) is the problem of dividing a string of written language into its component sentences.
- Same separation done for sentences is called sentence segmentation.

Word tokenization:-

- If the text is split into words using some separation technique it is called word tokenization.
- For example, the sentence “I won” can be tokenized into two word-tokens “I” and “won”.

Code:-

```
import nltk


text = "A good traveler has no fixed plans and is not intent on arriving"


sentences = nltk.sent_tokenize(text)


for sentence in sentences:
    words = nltk.word_tokenize(sentence)
    print(words)
```

Output:-

```
['A', 'good', 'traveler', 'has', 'no', 'fixed', 'plans', 'and', 'is', 'not', 'intent', 'on', 'arriving']
```

Practical No 2

Aim:- Write a program to Implement stemming and lemmatization

Background Information:-

Stemming:-

- Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling.
- For instance, stemming the word 'Caring' would return 'Car'.
- Stemming is used in case of large dataset where performance is an issue.
- Stemming is faster because it chops words without knowing the context of the word in given sentences.
- It is a rule-based approach.
- When we convert any word into root-form then stemming may create the non-existence meaning of a word.
- When we convert any word into root-form then stemming may create the non-existence meaning of a word.
- Stemming accuracy is less.
- Stemming is preferred when the meaning of the word is not important for analysis.

Example: Spam Detection

Lemmatization:-

- Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma.
- For instance, lemmatizing the word 'Caring' would return 'Care'.
- Lemmatization is computationally expensive since it involves look-up tables and what not.
- It's used in computational linguistics, natural language processing (NLP) and chatbots.
- Lemmatization is slower as compared to stemming but it knows the context of the word before proceeding.
- It is a dictionary-based approach.
- Lemmatization always gives the dictionary meaning word while converting into root-form.

- Lemmatization accuracy is more as compared to Stemming.
- Lemmatization would be recommended when the meaning of the word is important for analysis.

Example: Question Answer

Code:-

```
import nltk

# nltk.download('punkt')

# nltk.download('wordnet')

words = ['eating', 'eats', 'eaten', 'eat']

stemmer = nltk.stem.PorterStemmer()

stemmed_words = [stemmer.stem(word) for word in words]

print("Stemmed words:", stemmed_words)

lemmatizer = nltk.stem.WordNetLemmatizer()

lemmatized_words = [lemmatizer.lemmatize(word) for word in words]

print("Lemmatized words:", lemmatized_words)
```

Output:-

```
Stemmed words: ['eat', 'eat', 'eaten', 'eat']
Lemmatized words: ['eating', 'eats', 'eaten', 'eat']
```

Practical NO 3

Aim:- Write a program to Implement a tri-gram model

Background Information:-

- It's a probabilistic model that's trained on a corpus of text. Such a model is useful in many NLP applications including speech recognition, machine translation and predictive text input. An N-gram model is built by counting how often word sequences occur in corpus text and then estimating the probabilities.
- N-gram is a sequence of the N-words in the modeling of NLP.
- N-gram is probably the easiest concept to understand in the whole machine learning space, I guess. An N-gram means a sequence of N words. So for example, “Medium blog” is a 2-gram (a bigram), “A Medium blog post” is a 4-gram, and “Write on Medium” is a 3-gram (trigram).

Code:-

```
import nltk

from nltk.util import ngrams

text = "The flame that burns Twice as bright burns half as long"

words = nltk.word_tokenize(text)

trigrams = ngrams(words, 3)

for trigram in trigrams:

   print(trigram)
```

Output:-

```
('The', 'flame', 'that')
('flame', 'that', 'burns')
('that', 'burns', 'Twice')
('burns', 'Twice', 'as')
('Twice', 'as', 'bright')
('as', 'bright', 'burns')
('bright', 'burns', 'half')
('burns', 'half', 'as')
('half', 'as', 'long')
```

Practical NO 4

Aim:- Write a program to Implement PoS tagging using HMM & Neural Model

Background Information:-

PoS tagging using HMM:-

HMM (Hidden Markov Model) is a Stochastic technique for POS tagging. Hidden Markov models are known for their applications to reinforcement learning and temporal pattern recognition such as speech, handwriting, gesture recognition, musical score following, partial discharges, and bioinformatics.

Neural Model:-

A neural network is a simplified model of the way the human brain processes information. It works by simulating a large number of interconnected processing units that resemble abstract versions of neurons. The processing units are arranged in layers.

Due to the advantages of recurrent neural networks in time series, in recent years, many researchers in the field of natural language processing have applied recurrent neural networks to research such as machine translation, language model learning, semantic role tagging, and part-of-speech tagging and achieved good ...

These language models are based on neural networks and are often considered as an advanced approach to execute NLP tasks. Neural language models overcome the shortcomings of classical models such as n-gram and are used for complex tasks such as speech recognition or machine translation.

Code:-

```
import nltk


nltk.download('punkt')

nltk.download('averaged_perceptron_tagger')

nltk.download('universal_tagset')
```

```
text = "Joe waited for the train, but the train was late"


words = nltk.word_tokenize(text)


hmm_tagged = nltk.pos_tag(words)
print("PoS tagging using HMM:", hmm_tagged)


nn_tagged = nltk.pos_tag(words, tagset='universal')
print("PoS tagging using NN:", nn_tagged)
```

Output:-

PoS tagging using HMM: [('Joe', 'NNP'), ('waited', 'VBD'), ('for', 'IN'), ('the', 'DT'), ('train', 'NN'), (',', ','), ('but', 'CC'), ('the', 'DT'), ('train', 'NN'), ('was', 'VBD'), ('late', 'JJ')]

PoS tagging using NN: [('Joe', 'NOUN'), ('waited', 'VERB'), ('for', 'ADP'), ('the', 'DET'), ('train', 'NOUN'), (',', '.'), ('but', 'CONJ'), ('the', 'DET'), ('train', 'NOUN'), ('was', 'VERB'), ('late', 'ADJ')]

Practical NO 5

Aim:- Write a program to Implement syntactic parsing of a given text

Background Information:-

Syntactic parsing involves the analysis of words in the sentence for grammar and their arrangement in a manner that shows the relationships among the words. Dependency grammar is a segment of syntactic text analysis. It determines the relationship among the words in a sentence.

Example, “I was on the hill when I used the telescope to see a man.” “I saw a man who was on a hill and who had a telescope.”

Code:-

```
import nltk


# Download the required resources , only if necessary/on first try!
# nltk.download('punkt')
# nltk.download('averaged_perceptron_tagger')
# nltk.download('maxent_ne_chunker')
# nltk.download('words')
# nltk.download('treebank')


text = "I ate hot ice-cream ,before match start"
words = nltk.word_tokenize(text)
tagged_words = nltk.pos_tag(words)


syntactic_tree = nltk.ne_chunk(tagged_words, binary=True)
print("Syntactic tree:", syntactic_tree)
```

Output:-

```
Syntactic tree: (S I/PRP ate/VBP hot/JJ ice-cream/NN ,/, before/IN match/JJ start/NN)
```

Practical NO 6

Aim:- Write a program to Implement dependency parsing of a given text

Background Information:-

In natural language processing, dependency parsing is a technique used to identify semantic relations between words in a sentence. Dependency parsers are used to map the words in a sentence to semantic roles, thereby identifying the syntactic relations between words.

In Dependency parsing, various tags represent the relationship between two words in a sentence. These tags are the dependency tags. For example, In the phrase 'rainy weather,' the word rainy modifies the meaning of the noun weather.

Code:-

```
import spacy


nlp = spacy.load("en_core_web_sm")
text = "Nitin is studying at Indian Instute of technology Bombay."
doc = nlp(text)


for entity in doc.ents:
    print(entity.label_, entity.text)
```

Output:-

```
PERSON Nitin
ORG Indian Instute of technology
GPE Bombay
```

Practical NO 7

Aim:- Write a program to Implement Named Entity Recognition (NER)

Background Information:-

Named Entity Recognition (NER) is a field of computer science and natural language processing that deals with the identification and classification of named entities in text. The goal of NER is to automatically extract information from unstructured text, such as names of people, organizations, locations, and so on.

For example, an NER machine learning (ML) model might detect the word “super.AI” in a text and classify it as a “Company”. NER is a form of natural language processing (NLP), a subfield of artificial intelligence.

Named entity recognition (NER) helps you easily identify the key elements in a text, like names of people, places, brands, monetary values, and more. Extracting the main entities in a text helps sort unstructured data and detect important information, which is crucial if you have to deal with large datasets.

Code:-

```
import spacy


nlp = spacy.load("en_core_web_sm")

text = "Nitin is studying at Indian Instute of technology Bombay."

doc = nlp(text)


for entity in doc.ents:

    print("entity.label_", entity.text)
```

Output:-

```
entity.label_ Nitin
entity.label_ Indian Instute of technology
entity.label_ Bombay
```

Practical NO 8

Aim:- Write a program to Implement Text Summarization for the given sample text

Background Information:-

NLP text summarization is the process of breaking down lengthy text into digestible paragraphs or sentences. This method extracts vital information while also preserving the meaning of the text. This reduces the time required for grasping lengthy pieces such as articles without losing vital information.

Text summarization is a very useful and important part of Natural Language Processing (NLP). First let us talk about what text summarization is. Suppose we have too many lines of text data in any form, such as from articles or magazines or on social media. We have time scarcity so we want only a nutshell report of that text. We can summarize our text in a few lines by removing unimportant text and converting the same text into smaller semantic text form.

The main objective of a text summarization system is to identify the most important information from the given text and present it to the end users.

Code:-

```
from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize, sent_tokenize

from heapq import nlargest


text = """

Natural language processing (NLP) is a branch of artificial intelligence that focuses on the interaction between computers and human language. NLP has been around for several decades, but recent advances in machine learning and deep learning have dramatically improved its capabilities. NLP is used in a wide range of applications, from virtual assistants like Siri and Alexa to sentiment analysis, machine translation, and even content generation. NLP involves a range of techniques, including tokenization, part-of-speech tagging, named entity recognition, and sentiment analysis, among others. These techniques can be used to analyze and understand human language in a variety of contexts, from social media posts to
```

scientific literature. Despite its many successes, NLP remains a challenging field, as natural language is complex and often ambiguous. As NLP continues to evolve, it has the potential to transform the way we interact with technology and with each other, opening up new possibilities for communication, collaboration, and creativity.

"""

```
num_sentences = 2

sentences = sent_tokenize(text)

words = word_tokenize(text)

stop_words = set(stopwords.words('english'))


word_freq = {}

for word in words:

    if word not in stop_words:

        if word not in word_freq:

            word_freq[word] = 1

        else:

            word_freq[word] += 1


            max_freq = max(word_freq.values())


for word in word_freq.keys():

    word_freq[word] = (word_freq[word]/max_freq)


sent_scores = {}

for sentence in sentences:
```

```
    for word in word_tokenize(sentence.lower()):
        if word in word_freq.keys():
            if len(sentence.split(' ')) < 30:
                if sentence not in sent_scores.keys():
                    sent_scores[sentence] = word_freq[word]
                else:
                    sent_scores[sentence] += word_freq[word]

summary_sentences = nlargest(num_sentences, sent_scores, key=sent_scores.get)

summary = ' '.join(summary_sentences)
print(summary)
```

Output:-

NLP involves a range of techniques, including tokenization, part-of-speech tagging, named entity recognition, and sentiment analysis, among others. NLP is used in a wide range of applications, from virtual assistants like Siri and Alexa to sentiment analysis, machine translation, and even content generation.