NEW SYLLABUS:

EXERCISES FOR NATURAL LANGUAGE PROCESSING LABORATORY

WEEK\_1:

1. Introduction, Installation and Accessing Additional Resources:

Natural Language Toolkit (NLTK) is one of the vastest Python libraries for performing various Natural

Language Processing tasks. From rudimentary tasks such as text pre-processing to tasks like vectorized

representation of text – NLTK’s API has covered everything. In this article, we will accustom ourselves

to the basics of NLTK and perform some crucial NLP tasks: Tokenization, Stemming, Lemmatization,

and POS Tagging.

**INSTALLATION:**

**Code:** pip install nltk

**NOTE:** Make sure you care connected to internet

1. **1.1 Tokenization:**

**Code:**

import nltk

from nltk.stem import PorterStemmer, WordNetLemmatizer

# Sample text

text = "Tokenization is an important step in natural language processing. It helps break down text into smaller units."

# Word tokenization

words = nltk.word\_tokenize(text)

print("Word Tokenization:")

print(words)

# Sentence tokenization

sentences =nltk.sent\_tokenize(text)

print("\nSentence Tokenization:")

print(sentences)

**output:**

Word Tokenization:

['Tokenization', 'is', 'an', 'important', 'step', 'in', 'natural', 'language', 'processing', '.', 'It', 'helps', 'break', 'down', 'text', 'into', 'smaller', 'units', '.']

Sentence Tokenization:

['Tokenization is an important step in natural language processing.', 'It helps break down text into smaller units.']

1. **1.2 Stemming and Lemmatization:**

Stemming and lemmatization are two important text normalization techniques used in natural language processing (NLP). They both aim to reduce words to their base or root form, but they operate differently.

1. Stemming: Stemming involves cutting off prefixes or suffixes of words to obtain their root forms, known as stems. Stemming algorithms are typically rule-based and may not always produce actual dictionary words. However, they are computationally efficient. For example, "running" would be stemmed to "run", and "happiness" would be stemmed to "happi".

2. Lemmatization: Lemmatization, on the other hand, involves reducing words to their canonical or dictionary forms, known as lemmas. Lemmatization algorithms use lexical knowledge bases to ensure that the resulting lemmas are actual words. Lemmatization is more accurate than stemming but may be computationally more expensive. For example, both "running" and "ran" would be lemmatized to "run", and "happiness" would be lemmatized to "happy".

**Code:**

import nltk

from nltk.stem import PorterStemmer, WordNetLemmatizer

from nltk.tokenize import word\_tokenize

# Sample text

text = "Stemming and lemmatization are important techniques for text normalization in NLP."

# Tokenize the text

tokens = word\_tokenize(text)

# Perform stemming

stemmer = PorterStemmer()

stemmed\_words = [stemmer.stem(word) for word in tokens]

# Perform lemmatization

lemmatizer = WordNetLemmatizer()

lemmatized\_words = [lemmatizer.lemmatize(word) for word in tokens]

print("Stemmed words:", stemmed\_words)

print("Lemmatized words:", lemmatized\_words)

**output:**

Stemmed words: ['stem', 'and', 'lemmat', 'are', 'import', 'techniqu', 'for', 'text', 'normal', 'in', 'nlp', '.']

Lemmatized words: ['Stemming', 'and', 'lemmatization', 'are', 'important', 'technique', 'for', 'text', 'normalization', 'in', 'NLP', '.'

This code snippet demonstrates how to use NLTK's `PorterStemmer` for stemming and `WordNetLemmatizer` for lemmatization. Both techniques can help in reducing the vocabulary size and improving the performance of NLP tasks by treating similar words as identical.

**1.3 Parts of Speech Tagging:**

Part-of-speech (POS) tagging is a fundamental task in natural language processing(NLP) that involves assigning a grammatical category (such as noun, verb, adjective, etc.) to each word in a text. POS tagging helps in understanding the syntactic structure of sentences and is useful for various NLP tasks like information extraction, sentiment analysis, and machine translation.

Here's a brief overview of the commonly used parts of speech:

- \*\*Noun (NN)\*\*: Represents a person, place, thing, or idea. Example: "cat", "table", "idea".

- \*\*Verb (VB)\*\*: Represents an action or state of being. Example: "run", "eat", "is".

- \*\*Adjective (JJ)\*\*: Describes or modifies a noun. Example: "big", "beautiful", "happy".

- \*\*Adverb (RB)\*\*: Describes or modifies a verb, adjective, or other adverb. Example: "quickly", "very", "often".

- \*\*Pronoun (PRP)\*\*: Replaces a noun in a sentence. Example: "he", "she", "it".

- \*\*Preposition (IN)\*\*: Indicates a relationship between nouns or pronouns and other words in a sentence. Example: "on", "in", "under".

- \*\*Conjunction (CC)\*\*: Connects words, phrases, or clauses. Example: "and", "but", "or".

- \*\*Interjection (UH)\*\*: Expresses emotion or sentiment. Example: "wow", "ouch", "oh".

Python provides various libraries and tools for POS tagging, including NLTK and spaCy. Below is a Python code snippet demonstrating POS tagging using NLTK:

**Code:**

import nltk

from nltk.tokenize import word\_tokenize

nltk.download('averaged\_perceptron\_tagger')

# Sample text

text = "Parts of speech tagging helps in understanding the syntactic structure of sentences."

# Tokenize the text

tokens = word\_tokenize(text)

# Perform POS tagging

pos\_tags = nltk.pos\_tag(tokens)

print("POS tags:", pos\_tags)

**output:**

POS tags: [('Parts', 'NNS'), ('of', 'IN'), ('speech', 'NN'), ('tagging', 'NN'), ('helps', 'VBZ'), ('in', 'IN'), ('understanding', 'VBG'), ('the', 'DT'), ('syntactic', 'JJ'), ('structure', 'NN'), ('of', 'IN'), ('sentences', 'NNS'), ('.', '.')]

In this code snippet, we tokenize the input text using NLTK's `word\_tokenize` function and then perform POS tagging using `pos\_tag`. The `pos\_tag` function assigns a POS tag to each token in the text, resulting in a list of tuples where each tuple contains a token and its corresponding POS tag.