FACULTY OF ENGINEERING AND TECHNOLOGY

BACHELOR OF TECHNOLOGY

DEEP LEARNING WITH NLP LABORATORY (303105480)

7TH SEMESTER

ARTIFICIAL INTELLIGENCE & DATA SCIENCE (A.I & D.S) DEPARTMENT

LABORATORY MANUAL

CERTIFICATE

This is to certify that

*Ms.**Gopi Kota** with Enrollment No.* **2303031247002** has *successfully completed his laboratory experiments in **Deep Learning with NLP (303105480)** from the department of* **Artificial Intelligence & Data Science (A.I & D.S)** *during the academic year **2025-2026.***

Date of Submission ………………. **Staff In charge ……………….**

Head of Department……………….

TABLE OF CONTENTS

SR.NO	Experiment Title	Page No		Date of Performance	Date of Submission	Marks	Sign
		From	To				
1	Implementation of preprocessing of Text with NLTK (Tokenization, Stemming, Lemmatization) and removal of stop words in NLP.						
2	Implementation to Convert the text to word count vectors with ScikitLearn (CountVectorizer).						
3	Implementation to Convert the text to word frequency vectors with ScikitLearn (TfidfVectorizer).						
4	Implementation to Convert the text to unique integers with ScikitLearn (HashingVectorizer).						
5	Use the Keras deep learning library and split words with (text_to_word_sequence)						
6	Use the Keras deep learning library and write a code for encoding with (one_hot).						
7	Use the Keras deep learning library and write a code for Hash Encoding with (hashing_trick).						
8	Use the Keras deep learning library give a demo of Tokenizer API						
9	Perform an experiment to do sentiment analysis on any of the dataset (like twitter tweets, movie review etc.).						
10	Perform an experiment using Gensim python library for Word2Vec Embedding.						

PRACTICAL: 01

Aim: Implementation of preprocessing of Text with NLTK (Tokenization, Stemming, Lemmatization) and removal of stop words in NLP.

Objective: The objective of this practical is to understand and apply basic text preprocessing techniques using NLTK that prepare raw textual data for NLP tasks such as classification, sentiment analysis, or information retrieval.

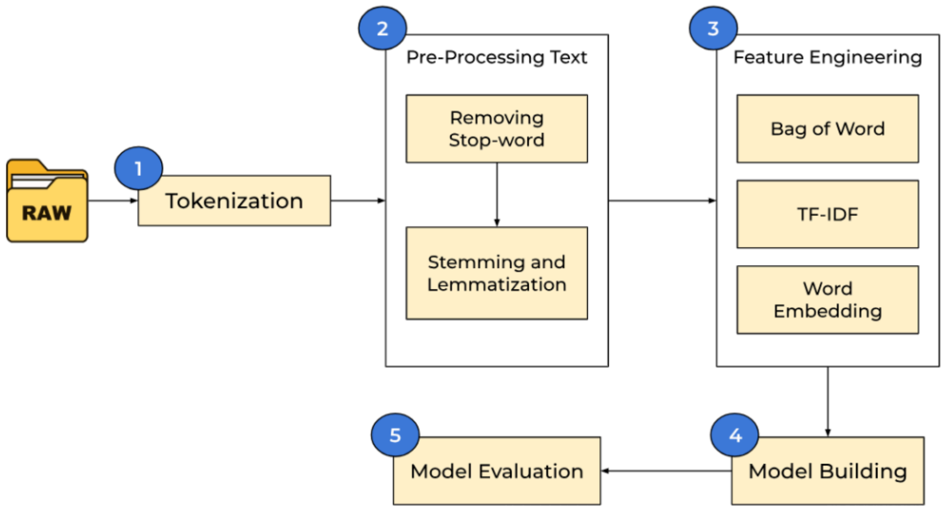
Theory:

- **Text preprocessing** is an essential step in NLP to convert raw text into a clean and analyzable format. Common preprocessing tasks include:
  - **Tokenization**: Splits text into individual words or tokens.
  - **Stemming**: Reduces words to their root form by chopping off suffixes (e.g., "playing" → "play").
  - **Lemmatization**: Reduces words to their dictionary form (lemma) considering grammar (e.g., "better" → "good").
  - **Stop Word Removal**: Eliminates common words (e.g., "is", "the") that do not contribute to meaning in many NLP tasks.
- These techniques help reduce dimensionality and improve the performance of NLP models.

About NLTK:

- NLTK (Natural Language Toolkit) is a powerful Python library used for working with human language data (text). It provides easy-to-use tools for tokenization, stemming, lemmatization, stop word removal, parsing, and more. NLTK is widely used in research and education for performing natural language processing tasks and building NLP models.

NLTK :

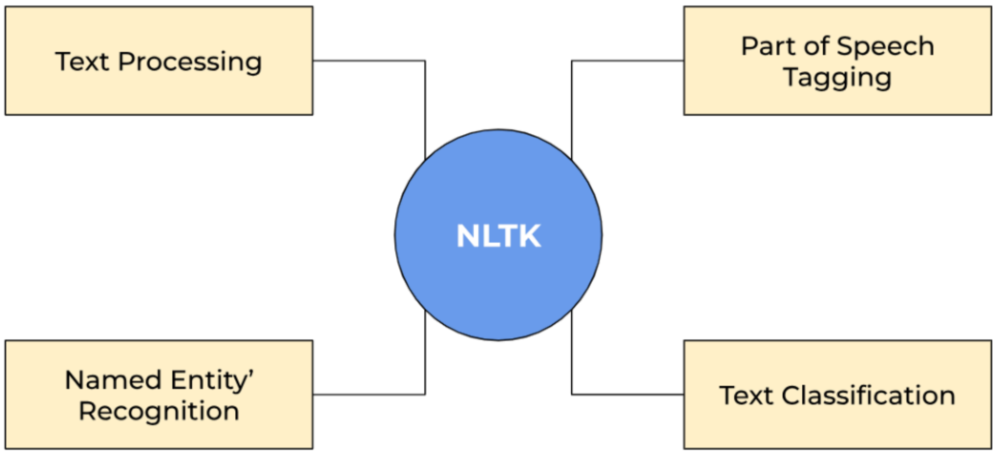


Explanation of the Text Processing Pipeline:

1. **Tokenization**:
   - Splits raw text into words or tokens.
   - Example: "The cat sat" → ["The", "cat", "sat"]
2. **Pre-Processing Text**:
   - Removing Stop-words: Eliminates common words (e.g., "the", "is") that carry less meaning.
   - Stemming & Lemmatization: Reduces words to their base or root form.
     - Example: "running" → "run"
3. **Feature Engineering**:
   - Converts processed text into numerical form for model input:
     - Bag of Words (BoW): Counts word occurrences.
     - TF-IDF (Term Frequency - Inverse Document Frequency): Measures word importance.
     - Word Embedding: Represents words in dense vector form (semantic meaning).

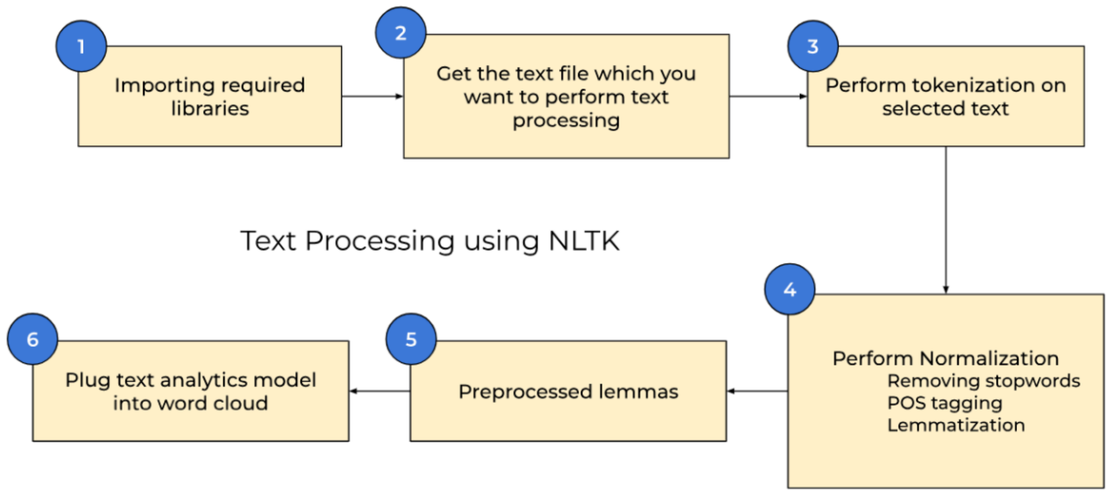
4. **Model Building**:
    - Uses the engineered features to train machine learning or deep learning models.
5. **Model Evaluation:**
    - Assesses model performance using metrics like accuracy, precision, recall, etc.

Natural Language Processing with NLTK – Key Capabilities:



- **Text Processing -** Basic tasks like tokenization, stop word removal, stemming, and lemmatization.
- **Part of Speech (POS) Tagging -** Identifies grammatical roles of words (e.g., noun, verb, adjective).
- **Named Entity Recognition (NER) -** Detects entities like names, locations, dates in text (e.g., "India", "Google").
- **Text Classification -** Categorizes text into predefined labels (e.g., spam vs. not spam, sentiment analysis).

Text Processing using NLTK – Steps Explained:


1 Importing required libraries
2 Get the text file which you want to perform text processing
3 Perform tokenization on selected text
Text Processing using NLTK
4 Perform Normalization
Removing stopwords
POS tagging
Lemmatization
5 Preprocessed lemmas
6 Plug text analytics model into word cloud

1. **Importing Required Libraries**
   - Load necessary Python packages like nltk for NLP tasks.
2. **Load Text File**
   - Open and read the input text file that you want to process.
3. **Tokenization**
   - Break the text into smaller units like words or sentences using word_tokenize() or sent_tokenize().
4. **Perform Normalization**
   - Clean and standardize the text:
     - **Remove Stopwords** (e.g., "the", "is")
     - **POS Tagging** (assign parts of speech to each word)
     - **Lemmatization** (convert words to their base forms)
5. **Preprocessed Lemmas**
   - The output is a clean list of lemmatized, tagged, and filtered words ready for analysis.
6. **Plug into WordCloud for Analytics Model**
   - Use the final cleaned data for visualizations like WordCloud or feed it into a text analytics or ML model.

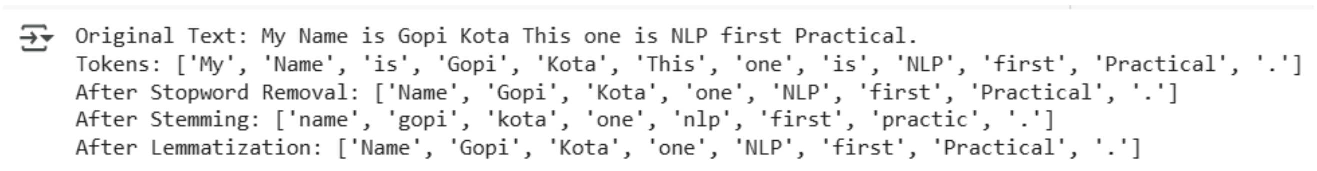
Code:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer

# Download required resources
# nltk.download('punkt')
# nltk.download('stopwords')
# nltk.download('wordnet')
# or use nltk.download('all')

# Sample input
text = "My Name is Gopi Kota This one is NLP first Practical."

# Tokenization
tokens = word_tokenize(text)

# Stopword Removal
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]

# Stemming
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]

# Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]

# Output
print("Original Text:", text)
print("Tokens:", tokens)
print("After Stopword Removal:", filtered_tokens)
print("After Stemming:", stemmed_tokens)
print("After Lemmatization:", lemmatized_tokens)
```

Input:

- **My Name is Gopi Kota This one is NLP first Practical."**

Output:



```
Original Text: My Name is Gopi Kota This one is NLP first Practical.
Tokens: ['My', 'Name', 'is', 'Gopi', 'Kota', 'This', 'one', 'is', 'NLP', 'first', 'Practical', '.']
After Stopword Removal: ['Name', 'Gopi', 'Kota', 'one', 'NLP', 'first', 'Practical', '.']
After Stemming: ['name', 'gopi', 'kota', 'one', 'nlp', 'first', 'practic', '.']
After Lemmatization: ['Name', 'Gopi', 'Kota', 'one', 'NLP', 'first', 'Practical', '.']
```

Conclusion:

- In this practical, we successfully implemented basic text preprocessing techniques using the NLTK library. The steps included **tokenization**, **stopword removal**, **stemming**, and **lemmatization**. These techniques are essential for preparing raw text data for further Natural Language Processing tasks, such as classification, sentiment analysis, or language modeling.

- By applying these preprocessing steps, we reduce noise, normalize the data, and extract meaningful information from text, making it more suitable for machine learning models.

PRACTICAL: 02

Aim: Implementation to Convert the text to word count vectors with ScikitLearn (CountVectorizer).

Objective: To understand how to represent textual data in a structured numerical format using the Bag-of-Words model, which helps transform text into fixed-size feature vectors suitable for machine learning algorithms.

Theory:

The **Bag-of-Words (BoW)** model is a simple technique to convert text documents into numerical vectors. It:

- Ignore grammar and word order.
- Focuses on word frequency.
- Represents each document as a vector of word counts.

CountVectorizer is a tool in sklearn.feature_extraction.text that:

- Tokenizes the text.
- Builds a vocabulary of known words.
- Encodes new documents using this vocabulary into word count vectors.

This format is commonly used for classification, clustering, and NLP tasks.

Code:

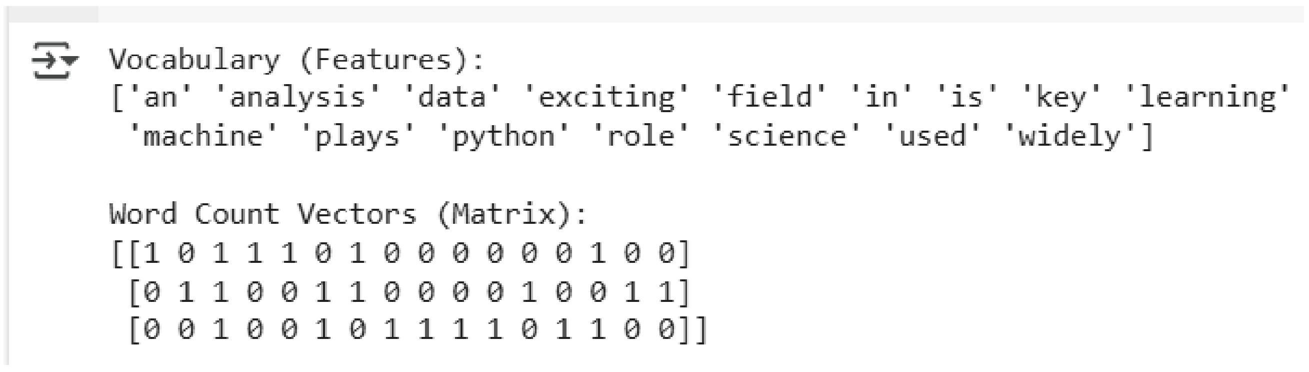
```
#P2
# Install scikit-learn if not already installed
!pip install scikit-learn --quiet

from sklearn.feature_extraction.text import CountVectorizer
```

```
# Sample corpus (text data)
corpus = [
    "Data science is an exciting field",
    "Python is widely used in data analysis",
    "Machine learning plays a key role in data science"
]

# Initialize CountVectorizer
vectorizer = CountVectorizer()

# Fit and transform the corpus
X = vectorizer.fit_transform(corpus)

# Convert result to array and get feature names
print("Vocabulary:", vectorizer.get_feature_names_out())
print("\nWord Count Vectors:\n", X.toarray())
```

Input:

```
corpus = [
    "Data science is an exciting field",
    "Python is widely used in data analysis",
    "Machine learning plays a key role in data science"
]
```

Output:



```
Vocabulary (Features):
['an' 'analysis' 'data' 'exciting' 'field' 'in' 'is' 'key' 'learning'
 'machine' 'plays' 'python' 'role' 'science' 'used' 'widely']

Word Count Vectors (Matrix):
[[1 0 1 1 1 0 1 0 0 0 0 0 0 1 0 0]
 [0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1]
 [0 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0]]
```

Conclusion:

- In this practical, we used CountVectorizer from Scikit-learn to convert text documents into numerical word count vectors. This technique allows raw text to be transformed into a format that can be used as input to machine learning models. It is a foundational step in building text classification, sentiment analysis, or information retrieval systems.