



## FACULTY OF ENGINEERING AND TECHNOLOGY

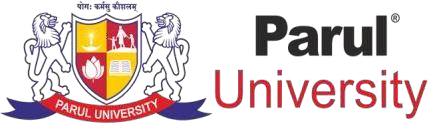
**BACHELOR OF TECHNOLOGY**

## DEEP LEARNING WITH N L P LABORATORY (203105477)

**7th SEMESTER**

### COMPUTER SCIENCE & ENGINEERING DEPARTMENT

LABORATORY MANUAL



# CERTIFICATE

This is to certify that **Mr.** **VOLADRI SAIKIRAN**

with enrolment no. **200303124537** has successfully completed his

laboratory experiments in the **DEEP LEARNING WITH NLP LABORATORY**

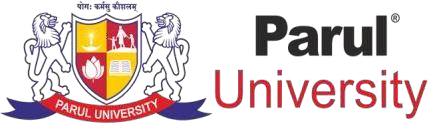
(203105477) from the department of CSE-AIduring the academic

year 2023-24



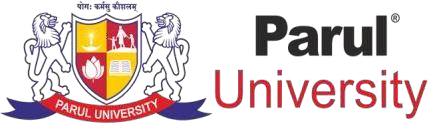
Date of Submission: ......................... Staff In charge: ......................................

Head Of Department: ...........................................



# TABLE OF CONTENT

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr. No** | **Experiment Title** | **Page No** | | **Date of Start** | **Date of Completion** | **Sign** | **Marks (Out of 10)** |
| **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 sentimental analysis on any of the dataset (like twitter tweets,  movie review etc.). |  |  |  |  |  |  |
| 10. | Perform an experiment using Gensim python library for Word2Vec Embedding. |  |  |  |  |  |  |



## PRACTICAL – 1

**AIM: -** Implementation of preprocessing of Text with NLTK (Tokenization, Stemming, Lemmatization and removal of stop words in NLP.

**TOKENIZATION: -**

[Tokenization](https://www.geeksforgeeks.org/tokenize-text-using-nltk-python/) refers to break down the text into smaller units. It entails splitting paragraphs into sentences and sentences into words. It is one of the initial steps of any NLP preprocessing.

**CODE: -**

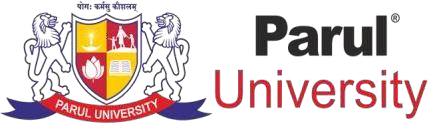
text = "this is deep learning practical 1

" tokens = text.split()

print(tokens)

### OUTPUT: -

****



#### STEMMING: -

[Stemming](https://www.geeksforgeeks.org/python-stemming-words-with-nltk/) generates the base word from the word by removing the affixes of the word. It must be noted that stemmers might not always result in semantically meaningful base words.

### CODE: -

import nltk

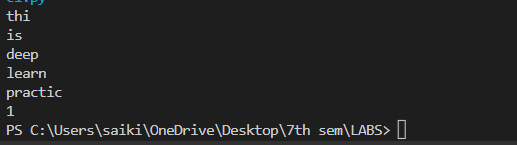
from nltk.stem import PorterStemmer ps = PorterStemmer()

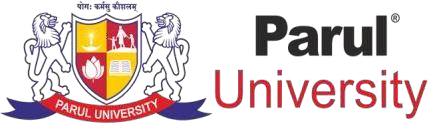
sentence = "this is deep learning practical 1 "

for word in sentence.split():

print(ps.stem(word))

### OUTPUT: -

****



#### LEMMATIZATION: -

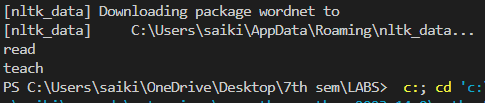
[Lemmatization](https://www.geeksforgeeks.org/python-lemmatization-with-nltk/) involves grouping together the inflected forms of the same word. This way, we can reach out to the base form of any word which will be meaningful in nature. The base from here is called the Lemma.

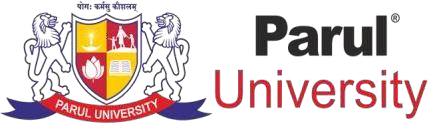
### CODE: -

import nltk nltk.download('wordnet')

from nltk.stem import WordNetLemmatizer lemmatizer = WordNetLemmatizer() print(lemmatizer.lemmatize("reading", pos= "v")) print(lemmatizer.lemmatize("teaching", pos= "v"))

### OUTPUT: -

****



#### REMOVAL OF STOP WORDS: -

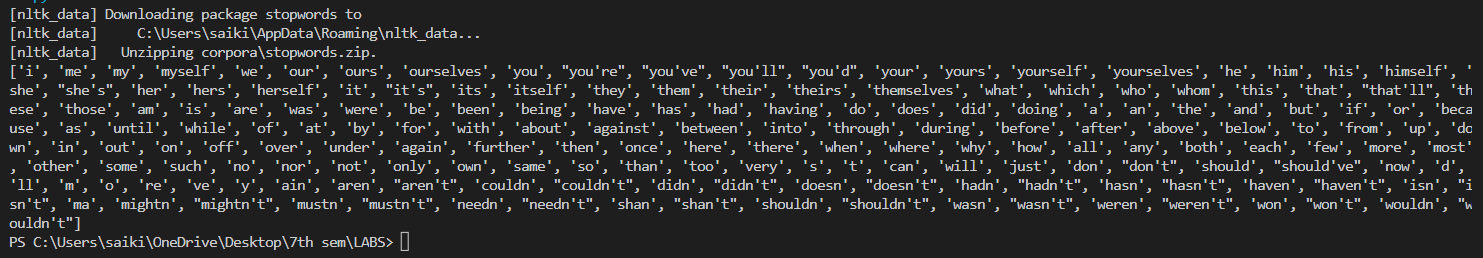
Stop word removal is one of the most commonly used preprocessing steps across different NLP applications. The idea is simply removing the words that occur commonly across all the documents in the corpus.

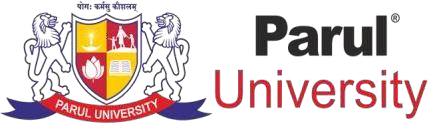
### CODE: -

import nltk

from nltk.corpus import stopwords nltk.download('stopwords') print(stopwords.words('english'))

### OUTPUT: -

****

**PRACTICAL - 2**

**AIM: -** Implementation to Convert the text to word count vectors with ScikitLearn (CountVectorizer).

### CODE: -

from sklearn.feature\_extraction.text import CountVectorizer

corpus = ["Sometimes life can get confusing and hard",

"If you don’t sacrifice for what you want then what you want is your sacrifice."]

vectorizer = CountVectorizer()

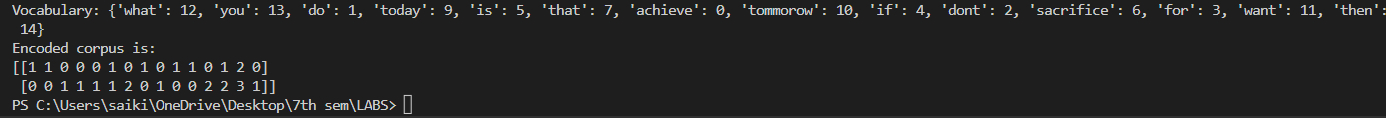
vectorizer.fit(corpus)

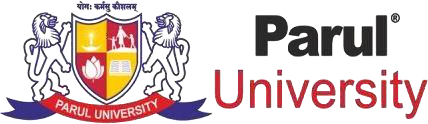
print("Vocabulary:" , vectorizer.vocabulary\_)

vector = vectorizer.transform(corpus)

print("Encoded corpus is:") print(vector.toarray()

**OUTPUT:**

****



## PRACTICAL – 3

**AIM: -** Implementation to Convert the text to word frequency vectors with ScikitLearn (TfidfVectorizer).

**TFIDF: -** TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction.

**FORMULAS: -**

Frequency = 𝑁𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑡𝑒𝑟𝑚𝑠 𝑟𝑒𝑝𝑒𝑎𝑡 𝑖𝑛 𝑡ℎ𝑒 𝑑𝑜𝑐𝑢𝑚𝑒𝑛𝑡

𝑇𝑜𝑡𝑎𝑙 𝑛𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑡𝑒𝑟𝑚𝑠 𝑖𝑛 𝑡ℎ𝑒 𝑑𝑜𝑐𝑢𝑚𝑒𝑛𝑡

IDF = 𝑁𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑑𝑜𝑐𝑢𝑚𝑒𝑛𝑡𝑠 𝑖𝑛 𝑡ℎ𝑒 𝑐𝑜𝑟𝑝𝑢𝑠

𝑇𝑜𝑡𝑎𝑙 𝑛𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑑𝑜𝑐𝑢𝑚𝑒𝑛𝑡𝑠 𝑖𝑛 𝑡ℎ𝑒 𝑐𝑜𝑟𝑝𝑢𝑠 𝑐𝑜𝑛𝑡𝑎𝑖𝑛 𝑡𝑒𝑟𝑚

**CODE: -**

from sklearn.feature\_extraction.text import TfidfVectorizer corpus = ["Sometimes life can get confusing and hard",

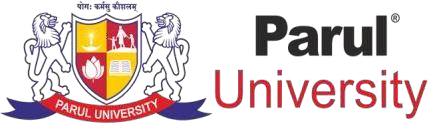
"In such times it can be useful to turn to the wisdom of poetry."]

vectorizer = TfidfVectorizer()

vectorizer.fit(corpus)

mykeys = list(vectorizer.vocabulary\_.keys()) mykeys.sort()

sorted\_dict = {i : vectorizer.vocabulary\_[i] for i in mykeys}



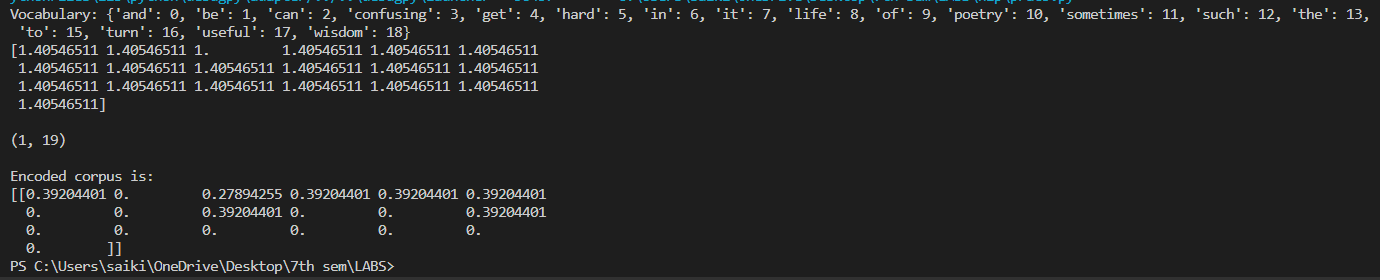
print("Vocabulary:" , sorted\_dict) print(vectorizer.idf\_)

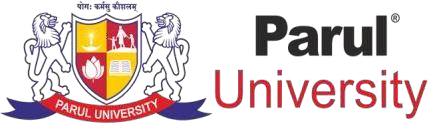
vector = vectorizer.transform([corpus[0]]) print()

print(vector.shape) print()

print("Encoded corpus is:") print(vector.toarray())

**OUTPUT: -**

****



## PRACTICAL – 4

**AIM: -** Implementation to Conveít the text to unique integeís withScikitLeaín(Hashing Vectoíizeí).

**Hashing Vectorizer:**

Hashing vectorizer is a vectorizer which uses the hashing trick to find the token string name to feature integer index mapping. Conversion of text documents into matrix is done by this vectorizer where it turns the collection of documents into a sparse matrix which are holding the token occurence counts.

**CODE: -**

from sklearn.feature\_extraction.text import HashingVectorizer corpus = ["Sometimes life an get confusing and hard",

"In such times it can be useful to turn to the wisdom of poetry."]

vectorizer = HashingVectorizer(n\_features = 3\*\*5)

vector = vectorizer.fit\_transform(corpus) print(vector.shape)

### OUTPUT: -

****

# PRACTICAL – 5

**AIM: -** Use the Keras deep learning library and split words with (text\_to\_word\_sequence).

**KERAS: -**

Keras is an open-source deep learning library written in Python. It

provides a user-friendly and modular interface for designing, building, training, and deploying various types of artificial neural networks, particularly deep neural networks. Keras was developed with a focus on enabling rapid experimentation and prototyping of deep learning models.

The features of the keras are User-Friendly, Modularity, Compatibility, Flexibility, Extensibility, Visualization.

For installation of the keras in colab we have to use the command

**!pip install -q keras**

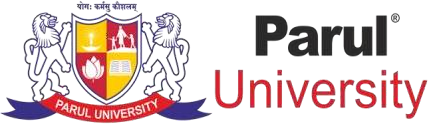
**CODE:**

from keras.preprocessing.text import text\_to\_word\_sequence txt = "todays hardwork is tomorrows life"

result = text\_to\_word\_sequence(txt) print(result)

## OUTPUT: -

****



**CODE: -**

from keras.preprocessing.text import text\_to\_word\_sequence txt = "todays hardwork is tomorrows life"

words = set(text\_to\_word\_sequence(txt)) vocab\_size = len(words) print(vocab\_size)

**OUTPUT: -**

****

# PRACTICAL – 6

**AIM: -** Use the Keras deep learning library and write a code for encoding with(one\_hot).

**ONE\_HOT: -**

One-hot encoding is a technique used in deep learning and natural language processing to represent categorical data, such as words or labels, as binary vectors. In one-hot encoding, each category is represented by a vector where all elements are set to zero except for the element corresponding to the category's index, which is set to one. This creates a unique binary representation for each category, allowing them to be easily fed into machine learning algorithms.

For example, consider a simple vocabulary of three words: "apple", and "banana”. To one-hot encode these words:

"apple" could be represented as [1, 0, 0]

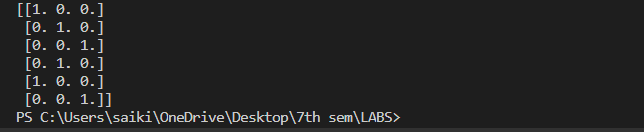
"banana" could be represented as [0, 1, 0]

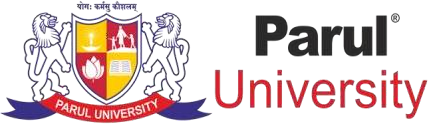
**CODE:**

from keras.utils import to\_categorical color\_labels = [0, 1, 2, 1, 0, 2]

one\_hot\_encoded = to\_categorical(color\_labels) print(one\_hot\_encoded)

**OUTPUT: -**





# PRACTICAL – 7

**AIM: -** Use the Keras deep learning library and write a code for Hash Encoding with (hashing\_trick).

**HASH ENCODING: -**

Hashing Encoding, often implemented using the **"hashing\_trick"** function in Keras, is a technique used to convert categorical data into numerical representations. This is particularly useful when dealing with a large number of categories and you want to reduce the dimensionality of the representation. Let's go through an example using movie genres.

Imagine you have a dataset of movies, and each movie is associated with one or more genres. The genres include "Action", "Comedy", "Drama", "Horror", "Romance", and so on. You want to represent these genres numerically using Hashing Encoding.

Before Hashing:

from keras.preprocessing.text import text\_to\_word\_sequence

text = " If you don’t sacrifice for what you want then what you want is your sacrifice”

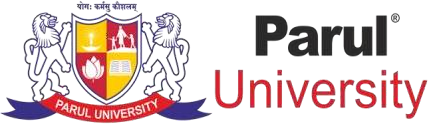
result = text\_to\_word\_sequence(text) print(result)

words = set(text\_to\_word\_sequence(text)) vocab\_size = len(words)

print(vocab\_size)

**CODE:**

****



**BY USING ONE\_HOT: -**

**CODE:**

from keras.preprocessing.text import one\_hot

from keras.preprocessing.text import text\_to\_word\_sequence

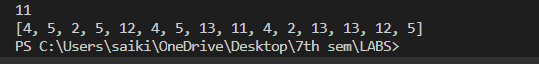
text = " If you don’t sacrifice for what you want then what you want is your sacrifice”

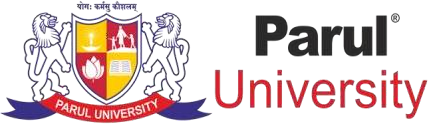
words = set(text\_to\_word\_sequence(text)) vocab\_size = len(words)

print(vocab\_size)

result = one\_hot(text, round(vocab\_size\*1.3)) print(result)

**OUTPUT:**





**AFTER HASHING: -**

**CODE:**

from keras.preprocessing.text import hashing\_trick

from keras.preprocessing.text import text\_to\_word\_sequence

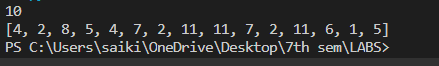
text = " If you don’t sacrifice for what you want then what you want is your sacrifice”

words = set(text\_to\_word\_sequence(text)) vocab\_size = len(words)

print(vocab\_size)

result = hashing\_trick(text, round(vocab\_size\*1.3), hash\_function='md5') print(result)

**OUTPUT: -**

****

# PRACTICAL – 8

**AIM :** Use the Keras deep learning library give a demo of Tokenizer API.

1. Tokenization: The tokenizer first breaks down the text into individual words or tokens. This is done using the fit\_on\_texts method.
2. Indexing: Each unique word is then assigned a unique integer index. This is done by the word\_index property of the tokenizer object.
3. Text to sequence conversion: The texts\_to\_sequences method can be used to convert a list of text samples to a list of sequences of integers.

**CODE:**

import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer(num\_words=1000)

texts = ["This is a text.", "This is another text."]

tokenizer.fit\_on\_texts(texts)

tokens = tokenizer.texts\_to\_sequences(texts)

print(tokens)

**OUTPUT:**

