



# **MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY**

**(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)**

Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015  
Maisammaguda, Dhulapally, Komapilly, Secunderabad - 500100, Telangana State, India

## **LABORATORY MANUAL & RECORD**

Name: .....

Roll No: ..... Branch: .....

Year: ..... Sem: .....





**MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY**  
**(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)**

Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015  
Maisammaguda, Dhulapally, Komaply, Secunderabad - 500100, Telangana State, India

# Certificate

Certified that this is the Bonafide Record of the Work Done by  
Mr./Ms.....Roll.No.....of  
B.Tech.....year..... Semester for Academic year.....  
in.....Laboratory.

Date:

Faculty Incharge

HOD

Internal Examiner

External Examiner

# INDEX

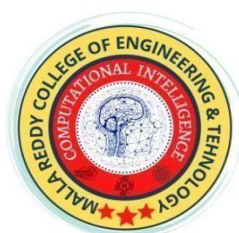
[illegible]

**NATURAL LANGUAGE PROCESSING  
LAB MANUAL  
(R22A6682)**

**B.TECH**



**(IV YEAR –I SEM)  
(2025-26)**



**DEPARTMENT OF CSE(AIML),AIML**

**MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY  
(Autonomous Institution – UGC, Govt. of India)**

Recognized under 2(f) and 12(B) of UGC ACT 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

**Department of Computer Science & Engineering**  
**(Artificial Intelligence & Machine Learning)**

**Vision**

To be a premier center for academic excellence and research through innovative interdisciplinary collaborations and making significant contributions to the community, organizations, and society as a whole.

.

**Mission**

- To impart cutting-edge Artificial Intelligence technology in accordance with industry norms.
- To instil in students a desire to conduct research in order to tackle challenging technical problems for industry by sustaining the ethical values.
- To develop effective graduates who are responsible for their professional growth, leadership qualities and are committed to lifelong learning.

**Quality Policy**

- To provide sophisticated technical infrastructure and to inspire students to reach their full potential.
- To provide students with a solid academic and research environment for a comprehensive learning experience.
- To provide research development, consulting, testing, and customized training to satisfy specific industrial demands, thereby encouraging self-employment and entrepreneurship among students.

### **Programme Educational Objectives (PEO):**

Graduates of the program will be able to

PEO1: Build successful careers in AI & ML and related fields by applying fundamental concepts of computer science, maths and specialized knowledge of intelligent systems.

PEO2: Design and implement AI-based solutions to real-world problems, demonstrating, creativity, critical thinking.

PEO3: Leverage the professional expertise to enter the workforce, seek higher education, and conduct research on AI-based problem resolution.

PEO4: Uphold ethical values and consider societal, legal, and environmental Consequences while developing intelligent systems, safeguarding responsible AI development.

### **Programme Specific Outcomes (PSO):**

After successful completion of the program a student is expected to have

Specific abilities to:

PSO 1: Translate end-user requirements into system and software requirements.

PSO 2: Generate a high-level design of the system from the software requirements.

PSO 3: Experience and/or awareness of testing problems and will be able to develop a simple testing report.

PSO 4: Understand and develop various structure and behavior UML diagrams.

PSO 5: Explain the knowledge of project management tool Demonstrate how to manage file using Project Libre project management tool.

## PROGRAM OUTCOMES (POs)

**Engineering Graduates should possess the following:**

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments.
12. **Life- long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



**MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY**  
**Maisammaguda, Dhulapally Post, Via Hakimpet, Secunderabad – 500100**  
**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**  
**(Artificial Intelligence & Machine Learning)**

**GENERAL LABORATORY INSTRUCTIONS**

1. Students are advised to come to the laboratory at least 5 minutes before (to starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:
  - a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
  - b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.
  - c. Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation notebook, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high-end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

## **Lab Objectives:**

- To introduce the basic concepts and techniques of Machine Learning and the need of Machine Learning techniques in real-world problems.
- To provide understanding of various Machine Learning algorithms and the way to evaluate performance of the Machine Learning algorithms.
- To apply Machine Learning to learn, predict and classify the real-world problems in the Supervised Learning paradigms as well as discover the Unsupervised Learning paradigms of Machine Learning.
- To inculcate in students professional and ethical attitude, multidisciplinary approach and an ability to relate real-world issues and provide a cost effective solution to it by developing ML applications.

## **Lab Outcomes:**

Upon successful completion of this course, the students will be able to:

- Understand the basic concepts and techniques of Machine Learning and the need of Machine Learning techniques in real-world problems.
- Understand various Machine Learning algorithms and the way to evaluate performance of the Machine Learning algorithms.
- Apply Machine Learning to learn, predict and classify the real-world problems in the Supervised Learning paradigms as well as discover the Unsupervised Learning paradigms of Machine Learning.
- Understand, learn and design Artificial Neural Networks of Supervised Learning for the selected problems.
- Understand the concept of Reinforcement Learning and Ensemble Methods

**Head of the Department**

**Principal**

## Introduction about lab

System configurations are as follows:

- **Hardware/Software's installed:** Intel®CORE™i3-3240CPU@3.40GHZRAM:4GB/Anaconda Navigator or Python and Jupyter Notebook or Google Colab.
- **Packages required to run the programs:** Math, Scipy, Numpy, Matplotlib, Pandas, Sklearn, Tensorflow, Keras etc.
- Systems are provided for students in the **1:1 ratio**.
- Explanation on today's experiment by the concerned faculty using PPT covering the following aspects  
Systems are assigned numbers and same system is allotted for students when they do the lab.
- All Systems are configured in LINUX, it is open source and students can use any different programming environments through package installation.

## Guidelines to students

### A. Standard operating procedure

a) :

- 1) Name of the experiment
- 2) Aim
- 3) Software/Hardware requirements
- 4) Writing the python programs by the students
- 5) Commands for executing programs

### Writing of the experiment in the Observation Book

The students will write the today's experiment in the Observation book as per the following format:

- a) Name of the experiment
- b) Aim
- c) Writing the program
- d) Viva-Voce Questions and Answers
- e) Errors observed (if any) during compilation/execution

Signature of the Faculty

### Instructions to maintain the record

- Before start of the first lab they have to buy their record and bring their record to the lab.
- Regularly (Weekly) update the record after completion of the experiment and get it corrected with concerned lab in-charge for continuous evaluation. In case the record is lost inform the same day to the faculty in charge and get the new record within 2 days the record has to be submitted and get it corrected by the faculty.
- If record is not submitted in time or record is not written properly, the evaluation marks (5M) will be deducted.

### Awarding the marks for day to day evaluation

Total marks for day to day evaluation is 15 Marks as per Autonomous (JNTUH). These 15 Marks are distributed as:

Regularity	3Marks
Program written	3Marks
Execution & Result	3Marks
Viva-Voce	3Marks
Dress Code	3Marks

### Allocation of Marks for Lab Internal

Total marks for lab internal are 40 Marks as per Autonomous (JNTUH.)

These 40 Marks are distributed as:

Average of day to day evaluation marks:15 Marks

Lab Mid exam:15 Marks

VIVA&Observation:10 Marks

### Allocation of Marks for Lab External

Total marks for lab Internal and External are 60Marks as per Autonomous/ (JNTUH).

These 60 External Lab Marks are distributed as:

Program Written	15Marks
Program Execution and Result	25Marks
Viva-Voce	10Marks
Record	10Marks

## INDEX

S.No.	Week	Program Name	Page No.
1	Week1	a. Write a python program to perform tokenization by word and sentence using nltk.	12
		b. Write a python program to eliminate stopwords using nltk.	
		c. Write a python program to perform stemming using nltk.	
2	Week2	a. Write a python program to perform Parts of Speech tagging using nltk.	19
		b. Write a python program to perform lemmatization using nltk.	
3	Week3	a. Write a python program for chunking using nltk.	23
		b. Write a python program to perform Named Entity Recognition using nltk.	
4	Week4	a. Write a python program to find Term Frequency and Inverse Document Frequency (TF-IDF).	27
		b. Write a python program for CYK parsing ( <b>Cocke-Younger-Kasami Parsing</b> ) or <b>Chart Parsing</b> .	
5	Week5	a. Write a python program to find all unigrams, bigrams and trigrams present in the given corpus.	31
		b. Write a python program to find the probability of the given statement “This is my cat” by taking the an exmple corpus into consideration.	
6	Week6	Use the Stanford named Entity recognizer to extract entities from the documents. Use it programmatically and output for each document which named entities it contains and of which type.	35

7	Week7	Choose any corpus available on the internet freely. For the corpus, for each document, count how many times each stop word occurs and find out which are the most frequently occurring stop words. Further, calculate the term frequency and inverse document frequency as The motivation behind this is basically to find out how important a document is to a given query. For e.g.: If the query is say: “The brown crow”. “The” is less important. “Brown” and “crow” are relatively more important. Since “the” is a more common word, its tf will be high. Hence we multiply it by idf, by knowing how common it is to reduce its weight.	37
8	Week8	Write the python code to perform sentiment analysis using NLP	39
9	Week9	Write the python code to develop Spam Filter using NLP	41
10	Week10	Write the python code to detect Fake News using NLP	43

**WEEK-1**

Date:

**Aim: a) Write a python program to perform tokenization by word and sentence using nltk.****Program for sentence tokenization:**

```
import nltk
nltk.download('punkt') # Download the necessary tokenization models

from nltk.tokenize import sent_tokenize

def tokenize_sentences(text):
    sentences = sent_tokenize(text)
    return sentences

# Example text
text = "NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum."

# Tokenize sentences
sentences = tokenize_sentences(text)

# Print tokenized sentences
for i, sentence in enumerate(sentences):
    print(f"Sentence {i+1}: {sentence}")
```

**Output:**

**Program for word Tokenization:**

```
import nltk
nltk.download('punkt') # Download the necessary tokenization models

from nltk.tokenize import word_tokenize

def tokenize_words(text):
    words = word_tokenize(text)
    return words

# Example text
text = "NLTK is a leading platform for building Python programs to work with human language data."

# Tokenize words
words = tokenize_words(text)

# Print tokenized words
print(words)
```

**Output:**

**b. Write a python program to eliminate stopwords using nltk.**

```
# Stopwords
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Download NLTK stopwords and tokenizer models
nltk.download('stopwords')
nltk.download('punkt')

def remove_stopwords(text):
    # Tokenize the text into words
    words = word_tokenize(text)

    # Get English stopwords
    english_stopwords = set(stopwords.words('english'))

    # Remove stopwords from the tokenized words
    filtered_words = [word for word in words if word.lower() not in english_stopwords]

    # Join the filtered words back into a single string
    filtered_text = ' '.join(filtered_words)

    return filtered_text

# Example text
text = "NLTK is a leading platform for building Python programs to work with human language data."

# Remove stopwords
filtered_text = remove_stopwords(text)

# Print filtered text
print(filtered_text)
```

**Output:****c. Write a python program to perform stemming using nltk.**

```
# Stemming
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

# Download NLTK tokenizer and stemmer models
nltk.download('punkt')
```

B.Tech-CSE(AIML),AIML)

def stem\_text(text):

    # Initialize the Porter Stemmer

    porter\_stemmer = PorterStemmer()

    # Tokenize the text into words

    words = word\_tokenize(text)

    # Apply stemming to each word

    stemmed\_words = [porter\_stemmer.stem(word) for word in words]

    # Join the stemmed words back into a single string

    stemmed\_text = ''.join(stemmed\_words)

    return stemmed\_text

# Example text

text = "NLTK is a leading platform for building Python programs to work with human language data."

# Perform stemming

stemmed\_text = stem\_text(text)

# Print stemmed text

print(stemmed\_text)

**Output:**

**EXERCISE:**

1. Write a python program to perform tokenization by word and sentence using Stanza.
2. Write a python program for word tokenization and sentence segmentation using spaCy.
3. Write a python program to find all the stopwords in the given corpus using spaCy.



Signature of the faculty

**WEEK-2**

Date:

**a. Write a python program to perform Parts of Speech tagging using nltk.****# Parts of Speech Tagging**

```
import nltk
from nltk.tokenize import word_tokenize

# Download NLTK tokenizer and POS tagging models
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

def pos_tagging(text):
    # Tokenize the text into words
    words = word_tokenize(text)

    # Perform POS tagging
    tagged_words = nltk.pos_tag(words)

    return tagged_words

# Example text
text = "NLTK is a leading platform for building Python programs to work with human language data."

# Perform POS tagging
tagged_text = pos_tagging(text)

# Print POS tagged text
print(tagged_text)
```

**Output:**

**b. Write a python program to perform lemmatization using nltk.**

```
#Lemmatization
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('wordnet')

def lemmatize_text(text):
    lemmatizer = WordNetLemmatizer()
    tokens = word_tokenize(text)
    lemmatized_text = ' '.join([lemmatizer.lemmatize(word) for word in tokens])
    return lemmatized_text

text = "The cats are chasing mice and playing in the garden"
lemmatized_text = lemmatize_text(text)
print("Original Text:", text)
print("Lemmatized Text:", lemmatized_text)
```

**Output:**

Signature of the Faculty

**EXERCISE:**

1. Study and use the Stanford Part of speech tagger on a suitable corpus available freely. The corpus should be of decent size. (Use spaCy and stanza).
2. Write a python program for lemmatization using spaCy and stanza.



**WEEK-3**

Date:

**a. Write a python program for chunking using nltk.**

#Chunking

import nltk

from nltk.tokenize import word\_tokenize

from nltk import pos\_tag, RegexpParser

nltk.download('punkt')

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

def chunk\_sentence(sentence):

words = word\_tokenize(sentence)

tagged\_words = pos\_tag(words)

# Define grammar for chunking

grammar = r"""

NP: {&lt;DT|JJ|NN.\*&gt;+} # Chunk sequences of DT, JJ, NN

PP: {&lt;IN&gt;&lt;NP&gt;} # Chunk prepositions followed by NP

VP: {&lt;VB.\*&gt;&lt;NP|PP|CLAUSE&gt;+\$} # Chunk verbs and their arguments

CLAUSE: {&lt;NP&gt;&lt;VP&gt;} # Chunk NP, VP pairs

"""

parser = RegexpParser(grammar)

chunked\_sentence = parser.parse(tagged\_words)

return chunked\_sentence

sentence = "The quick brown fox jumps over the lazy dog"

chunked\_sentence = chunk\_sentence(sentence)

print(chunked\_sentence)

**Output:**

**b. Write a python program to perform Named Entity Recognition using nltk.**

```
#Named Entity Recognition
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag, ne_chunk

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')

def ner(text):
    words = word_tokenize(text)
    tagged_words = pos_tag(words)
    named_entities = ne_chunk(tagged_words)
    return named_entities

text = "Apple is a company based in California, United States. Steve Jobs was one of its founders."
named_entities = ner(text)
print(named_entities)
```

**Output:****Signature of the faculty**

**EXERCISE:**

1. Write a python program for chunking using nltk.
2. Use the Stanford named Entity recognizer to extract entities from the documents. Use it programmatically and output for each document which named entities it contains and of which type.



**WEEK-4****Date:**

- a. Write a python program to find Term Frequency and Inverse Document Frequency (TF-IDF).

```
#tf-idf
import nltk
import string
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

nltk.download('punkt')# Sample documents
documents = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one.",
    "Is this the first document?",
]

# Tokenize and preprocess the documents
def preprocess_text(doc):
    # Tokenize the document into words
    tokens = nltk.word_tokenize(doc)

    # Remove punctuation
    tokens = [word for word in tokens if word not in string.punctuation]

    # Convert words to lowercase
    tokens = [word.lower() for word in tokens]

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # Join the tokens back into a single string
    preprocessed_doc = ' '.join(tokens)

    return preprocessed_doc

# Preprocess all documents
preprocessed_documents = [preprocess_text(doc) for doc in documents]

# Compute TF-IDF scores using scikit-learn
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(preprocessed_documents)

# Print TF-IDF matrix
print(tfidf_matrix.toarray())
```

**Output:**

**b. Write a python program for CYK parsing (Cocke-Younger-Kasami Parsing) or Chart Parsing.**

```
import nltk
grammar = nltk.CFG.fromstring(""" S -> V NP
V -> 'describe' | 'present' NP -> PRP N
PRP -> 'your' N -> 'work'
""")
parser = nltk.ChartParser(grammar) sent = 'describe your work'.split() print (list(parser.parse(sent)))
```

**Output:****Signature of the Faculty**

**EXERCISE:**

1. Write a python program for CYK Parsing by defining your own Grammar.

**Signature of the Faculty**

**WEEK-5**

Date:

**a. Write a python program to find all unigrams, bigrams and trigrams present in the given corpus.**

```
import nltk
nltk.download('punkt')
from nltk.util import ngrams
sampleText='this is a very good book to study'
for i in range(1,4):
    NGRAMS=ngrams(sequence=nltk.word_tokenize(sampleText), n=i)
    for grams in NGRAMS:
        print(grams)
```

**b. Write a python program to find the probability of the given statement “This is my cat” by taking the an exmple corpus into consideration.**

```
'This is a dog',  
'This is a cat',  
'I love my cat',  
'This is my name'
```

```
def readData():
```

```
    data = ['This is a dog','This is a cat','I love my cat','This is my name ']
```

```
    dat=[]
```

```
    for i in range(len(data)):
```

```
        for word in data[i].split():
```

```
            dat.append(word)
```

```
    print(dat)
```

```
    return dat
```

```
def createBigram(data):
```

```
    listOfBigrams = []
```

```
    bigramCounts = { }
```

```
    unigramCounts = { }
```

```
    for i in range(len(data)-1):
```

```
        if i < len(data) - 1 and data[i+1].islower():
```

```
            listOfBigrams.append((data[i], data[i + 1]))
```

```
            if (data[i], data[i+1]) in bigramCounts:
```

```
                bigramCounts[(data[i], data[i + 1])] += 1
```

```
            else:
```

```
                bigramCounts[(data[i], data[i + 1])] = 1
```

```
            if data[i] in unigramCounts:
```

```
                unigramCounts[data[i]] += 1
```

```
            else:
```

```
                unigramCounts[data[i]] = 1
```

```
return listOfBigrams, unigramCounts, bigramCounts
```

```
def calcBigramProb(listOfBigrams, unigramCounts, bigramCounts):
```

```
    listOfProb = { }
```

```
    for bigram in listOfBigrams:
```

```
        word1 = bigram[0]
```

```
        word2 = bigram[1]
```

```
        listOfProb[bigram] = (bigramCounts.get(bigram))/(unigramCounts.get(word1))
```

```
    return listOfProb
```

```
if __name__ == '__main__':
```

```
    data = readData()
```

```
    listOfBigrams, unigramCounts, bigramCounts = createBigram(data)
```

```
    print("\n All the possible Bigrams are ")
```

```
    print(listOfBigrams)
```

```
    print("\n Bigrams along with their frequency ")
```

```
    print(bigramCounts)
```

```
    print("\n Unigrams along with their frequency ")
```

```
    print(unigramCounts)
```

```
    bigramProb = calcBigramProb(listOfBigrams, unigramCounts, bigramCounts)
```

```
    print("\n Bigrams along with their probability ")
```

```
    print(bigramProb)
```

```
    inputList="This is my cat"
```

```
    splt=inputList.split()
```

```
    outputProb1 = 1
```

```
    bilist=[]
```

```
    bigrm=[]
```

```
for i in range(len(splt) - 1):
    if i < len(splt) - 1:

        bilist.append((splt[i], splt[i + 1]))

print("\n The bigrams in given sentence are ")
print(bilist)
for i in range(len(bilist)):
    if bilist[i] in bigramProb:

        outputProb1 *= bigramProb[bilist[i]]
    else:

        outputProb1 *= 0
print("\n' + 'Probablility of sentence \"This is my cat\" = ' + str(outputProb1))
```

**Signature of the Faculty**

**WEEK– 6**

Date:

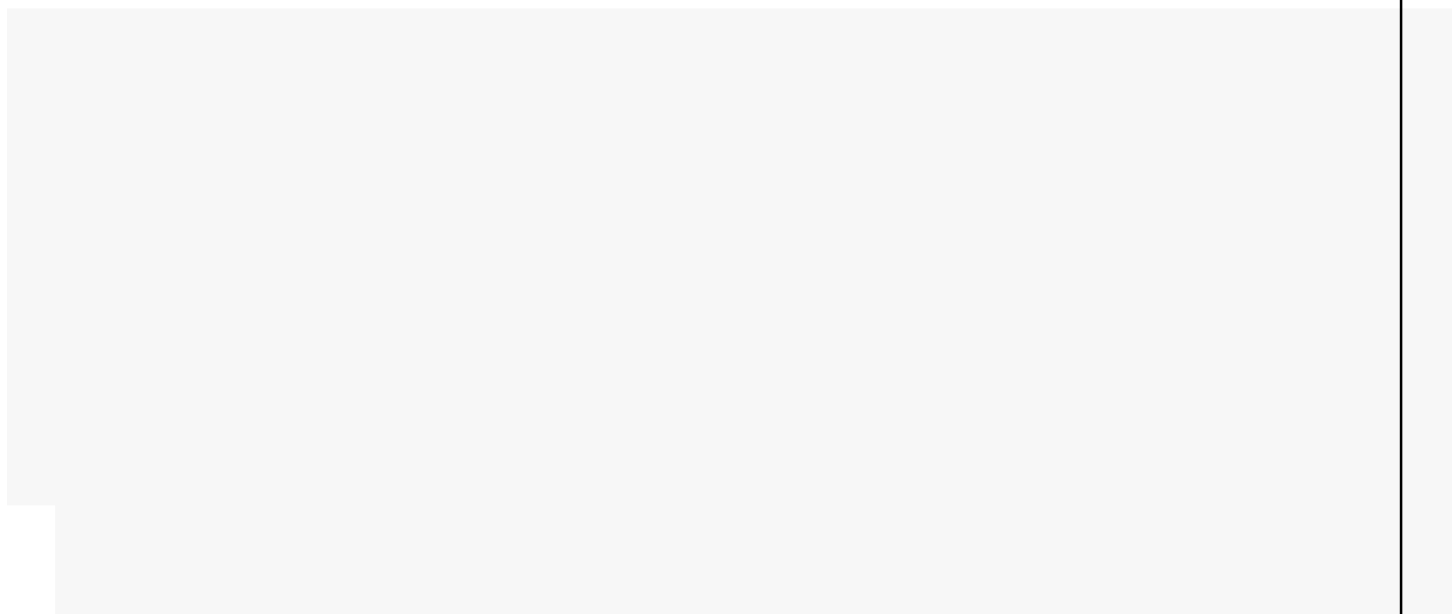
Use the Stanford named Entity recognizer to extract entities from the documents. Use It programmatically and output for each document which named entities it contains and of Which type.

**Signature of the Faculty**

**WEEK– 7**

Date:

Choose any corpus available on the internet freely. For the corpus, for each document, count how many times each stop word occurs and find out which are the most frequently occurring stop words. Further, calculate the term frequency and inverse document frequency as The motivation behind this is basically to find out how important a document is to a given query. For e.g.: If the query is say: “The brown crow”. “The” is less important. “Brown” and “crow” are relatively more important. Since “the” is a more common word, its tf will be high. Hence we multiply it by idf, by knowing how common it is to reduce its weight.



Signature of the Faculty

**Week- 8**

**Date:**

**a. Write the python code to perform sentiment analysis using NLP**

**Signature of the Faculty**

**WEEK– 9**

Date:

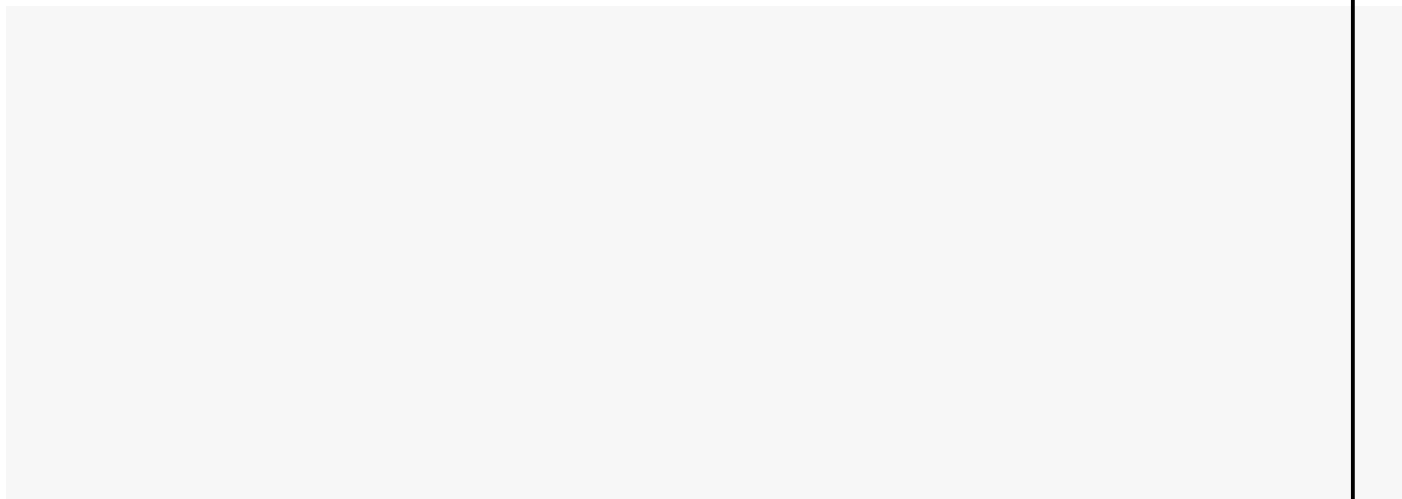
1. Write the python code to develop Spam Filter using NLP

**Signature of the Faculty**

Week-10:

Date:

1. Write the python code to detect Fake News using NLP



Signature of the Faculty

