INDEX

PREFACE

Natural Language Processing form the backbone of modern computing, enabling efficient solutions to complex problems in areas like optimization, networks, and data processing. This lab manual for M.Tech CSE I Year I Semester equips students with hands-on skills to implement and analyze key techniques—from brute-force enumeration to sophisticated string matching and flow algorithms—fostering deep insight into performance trade-offs.

The experiments align with core texts like S. Sridhar's *Design and Analysis of Algorithms* and references such as Cormen et al.'s *Introduction to Algorithms*, emphasizing practical coding in C/C++ alongside theoretical rigor. Through these 10 structured labs, students will not only code solutions but also evaluate time/space complexities, preparing them for real-world applications in IT infrastructure and systems design.

ACKNOWLEDGEMENT

It has been a rewarding experience working on the Natural Language Processing Laboratory Manual. We would like to express our sincere gratitude to **Dr.K.Abdul Basith**, Professor and Head of the Department of Computer Science and Engineering, Marri Laxman Reddy Institute of Technology & Management, for his constant encouragement, valuable guidance, and continuous support in the preparation of this manual.

We are deeply indebted and gratefully acknowledge the support and motivation provided by the **Dr.P.Sridhar** Director, Marri Laxman Reddy Institute of Technology & Management, for granting us the opportunity and necessary resources to develop this laboratory manual.

Our heartfelt thanks to **Dr. R. Murali Prasad**, Principal, Marri Laxman Reddy Institute of Technology & Management, for his insightful suggestions, timely feedback, and academic guidance throughout the preparation of this document.

Finally, we extend our sincere appreciation to all the faculty members of the CSE Department whose encouragement, cooperation, and constructive inputs have played a significant role in helping us accomplish this work successfully.

COMPUTER SCIENCE AND ENGINEERING
NATURAL LANGUAGE PROCESSING LAB MANUAL

GENERAL INSTRUCTIONS

- Students are instructed to attend the Natural Language Processing laboratory on time. Late comers will not be entertained in the lab.
- Students must be punctual. Experiments conducted during the session will not be repeated for those who are absent or late.
- Students are expected to come prepared with the theory and procedure of the experiment scheduled for the day.
- The use of mobile phones inside the lab is strictly prohibited.
- Any damage or loss of system components such as keyboard, mouse, or other peripherals during the lab session will be the responsibility of the student, and a fine or penalty will be imposed accordingly.
- Students must update their lab records and observation books session-wise. Before leaving the lab, the student should get their observation book signed by the concerned faculty member.
- Lab records must be submitted in the next lab session to the respective faculty members in the staffroom for correction and return.
- Students should not move around or disturb others during the lab session.
- In case of any emergency, students must obtain written permission from the concerned faculty member before leaving the laboratory.
- Faculty members reserve the right to suspend any student from the lab session on disciplinary grounds.

COMPUTER SCIENCE AND ENGINEERING
NATURAL LANGUAGE PROCESSING LAB MANUAL

INSTITUTE VISION AND MISSION

Vision:

To be as an ideal academic institution by graduating talented engineers to be ethically strong, competent with quality research and technologies.

Mission:

- Utilize rigorous educational experiences to produce talented engineers
- Create an atmosphere that facilitates the success of students
- Programs that integrate global awareness, communication skills and Leadership qualities
- Education and Research partnership with institutions and industries to prepare the students for interdisciplinary research

COMPUTER SCIENCE AND ENGINEERING
NATURAL LANGUAGE PROCESSING LAB MANUAL

DEPARTMENT VISION AND MISSION

Vision:

To empower the students to be technologically adept, innovative, self-motivated and responsible global citizen possessing human values and contribute significantly towards high quality technical education with ever changing world.

Mission:

- To offer high-quality education in the computing fields by providing an environment where the knowledge is gained and applied to participate in research, for both students and faculty.
- To develop the problem solving skills in the students to be ready to deal with cutting edge technologies of the industry.
- To make the students and faculty excel in their professional fields by inculcating the communication skills, leadership skills, team building skills with the organization of various co-curricular and extra- curricular programmes.
- To provide the students with theoretical and applied knowledge, and adopt an education approach that promotes lifelong learning and ethical growth.

COMPUTER SCIENCE AND ENGINEERING
NATURAL LANGUAGE PROCESSING LAB MANUAL

PROGRAM OUTCOMES

PO No.	NBA Statement / Vital Features	No. of Vital Features
PO1.	**An ability to independently carry out research /investigation and development work to solve practical problems.** 1. Research problems in Computer Science and Engineering are clearly identified and defined. 2. Literature review highlights research gaps and suitable methods. 3. Experiments or simulations are conducted using appropriate tools. 4. Data collection, analyses, and interpretation systematically. 5. Innovative approaches are applied to engineering problem-solving. **6.** Results are validated against established theories and standards.	**6**
PO 2.	**An ability to write and present a substantial technical report/document** 1. Technical reports, dissertations, and papers are well-structured. 2. Referencing and academic integrity practices are properly maintained. 3. Content is presented with clarity, precision, and logical flow. 4. Oral communication and presentation skills are effectively demonstrated. 5. Digital tools are used for documentation and visualization. **6.** Research findings are communicated to both technical and non-technical audience.	**6**
PO 3.	**Students should be able to demonstrate advanced proficiency in Computer Science and allied emerging areas of Engineering.** i. In-Depth Technical Knowledge ii. Advanced Problem-Solving and Algorithmic Skills. iii. Hands-On Practical Expertise iv. Research and Innovation Aptitude **v.** Interdisciplinary Integration	**5**
PO 4.	**Students should be able to identify, analyze, and effectively solve complex real-world problems by applying advanced computing concepts, while considering solutions from a global perspective**	

	i. Ability to break down multifaceted problems into manageable parts and develop effective solutions using advanced computing techniques. ii. Analytical Thinking and Critical Reasoning iii. Advanced Computing Knowledge. iv. Capability to understand and model complex systems, considering interdependencies and dynamic behaviors within global contexts. v. Global and Societal Awareness. vi. Aptitude for developing novel approaches and innovative solutions to address complex challenges. vii. Ability to work effectively in diverse teams and integrate knowledge from multiple disciplines to solve global problems. viii. Skill in clearly presenting problem analyses, solutions, and their implications to technical and non-technical global audience.	**8**
PO 5.	**An ability to acquire and apply advanced technical knowledge, professional skills, and modern computing tools to develop sustainable solutions** i. Ability to continuously learn and apply cutting-edge technical knowledge in computing and related fields. ii. Proficiency with Modern Computing Tools. iii. Capability to design and implement solutions that are environmentally, and economically feasible. iv. Understanding of professional ethics and responsibility in creating technology solutions with long-term positive impact. v. Integration of Multidisciplinary Knowledge. vi. Adaptability to sustainable practices.	**6**
PO 6.	**An Ability to recognize the significance of lifelong learning and actively pursue continuous professional development by adapting technologies in emerging areas.** i. Self-Directed Learning. ii. Skill in quickly learning and integrating new technologies and tools as they emerge in the field. iii. Capability to assess the relevance and impact of emerging technologies and decide on their applicability. iv. Continuous Professional Development Planning. v. Ability to engage with professional communities, attend workshops, and collaborate to stay updated. vi. Habit of regularly reflecting on personal growth, learning experiences, and professional competencies to improve continuously.	**6**

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

COMPUTER SCIENCE AND ENGINEERING

NATURAL LANGUAGE PROCESSING LAB MANUAL

PROGRAM EDUCATIONAL OBJECTIVES

Sl. No.	PEOs Name	Program Education Objective Statements
1	PEO – 1	Graduates will achieve professional excellence and success in the field of Computer Science and Engineering by applying strong technical foundations and problem-solving skills to contribute effectively to industry, academia, and entrepreneurship.
2	PEO – 2	Graduates will demonstrate a commitment to lifelong learning by continuously enhancing their knowledge and skills through professional development and self-directed learning to effectively adapt to evolving global challenges.
3	PEO – 3	Graduates of the Computer Science and Engineering program will actively pursue advanced research, contributing to the development of solutions for complex problems and the generation of new knowledge to effectively address real-world challenges.
4	PEO – 4	Graduates will exhibit professionalism, effective communication, leadership skills, and ethical responsibility while working in multidisciplinary teams to deliver computing solutions that address societal needs and contribute to sustainable development.

COMPUTER SCIENCE AND ENGINEERING
NATURAL LANGUAGE PROCESSING LAB MANUAL

PROGRAM SPECIFIC OUTCOMES

PSO1: Applications of Computing: Ability to use knowledge in various domains to provide solution to new ideas and innovations.

PSO2: Programming Skills: Identify required data structures, design suitable algorithms, develop and maintain software for real world problems.

COMPUTER SCIENCE AND ENGINEERING
NATURAL LANGUAGE PROCESSING LAB MANUAL

Course Structure

Natural Language ProcessingLab will have a continuous evaluation during II semester for 40 sessional marks and 60 end semester examination marks.Out of the 40 marks for internal evaluation, day-to-day work in the laboratory shall be evaluated for 20 marks and internal practical examination shall be evaluated for 10 marks conducted by the laboratory teacher concerned.

The end semester examination shall be conducted with an external examiner and internal examiner. The external examiner shall be appointed by the principal / Chief Controller of examinations

COMPUTER SCIENCE AND ENGINEERING
NATURAL LANGUAGE PROCESSING LAB MANUAL

OBJECTIVE:

- To understand the fundamental concepts and techniques of Natural Language Processing, including linguistic structures and text representation.
- To develop skills in preprocessing textual data such as tokenization, stemming, lemmatization, and stop-word removal.
- To apply syntactic and semantic analysis techniques such as part-of-speech tagging, parsing, and morphology.
- To design and implement language models, including N-grams and probabilistic approaches for text prediction and analysis.
- To build and evaluate real-world NLP applications such as sentiment analysis, machine translation, chatbots, and information retrieval systems.

OUTCOMES:

Upon the completion of Operating Systems practical course, the student will be able to:

- Implement Python programs for text preprocessing tasks such as tokenization, stop-word removal, and stemming to prepare textual data for NLP applications like chatbots and search engines.
- Apply part-of-speech tagging and morphological analysis to extract syntactic information for tasks such as grammar checking and text parsing.
- Develop word analysis and word generation techniques for building applications like auto- completion and predictive text systems.

.

NATURAL LANGUAGE PROCESSING LAB (LAB - II)

M.Tech CSE I Year I Sem.

L	T	P	C
0	0	4	2

Prerequisites: Data structures, finite automata and probability theory

Course Objectives:

- To Develop and explore the problems and solutions of NLP.

Course Outcomes:

1. Implement Python programs for text preprocessing tasks such as tokenization, stop-word removal, and stemming to prepare textual data for NLP applications like chatbots and search engines.
2. Apply part-of-speech tagging and morphological analysis to extract syntactic information for tasks such as grammar checking and text parsing.
3. Develop word analysis and word generation techniques for building applications like auto- completion and predictive text systems.
4. Construct chunking and N-gram models, including smoothing techniques, to perform phrase detection and probabilistic language modelling for speech recognition and text prediction.
5. Evaluate the performance of NLP preprocessing and language modelling methods for real- world applications such as document summarization and sentiment analysis.

List of Experiments

Implement the following using Python

1. Tokenization
2. Stemming
3. Stop word removal (a, the, are,.)
4. Word Analysis
5. Word Generation
6. Pos tagging
7. Morphology
8. chunking
9. N-Grams
10. N-Grams Smoothing

TEXT BOOKS:

1. Multilingual natural Language Processing Applications: From Theory to Practice – Daniel M. Bikel and Imed Zitouni, Pearson Publication
2. Natural Language Processing and Information Retrieval: Tanvier Siddiqui, U.S. Tiwary

REFERENCES:

1. Speech and Natural Language Processing - Daniel Jurafsky & James H Martin, Pearson Publications

Experiment-1

Implement Tokenization using python

```
import nltk
text = "This is a sample sentence. Tokenization is important in NLP."
tokens = nltk.word_tokenize (text)
print ("Tokens:", tokens)
```

OUTPUT:

```
Tokens: ['This', 'is', 'a', 'sample', 'sentence', '.', 'Tokenization', 'is', 'important', 'in', 'NLP', '.']
```

Experiment-2

Implement Stemming using python

```
from nltk.stem import PorterStemmer
porter = PorterStemmer()
words = ["running", "runs", "ran", "runner", "easily",     "eating", "eats"]
stemmed_words = [porter.stem(word) for word in words]
print("Stemmed words:", stemmed_words)
```

OUTPUT:

```
Stemmed words: ['run', 'run', 'ran', 'runner', 'easili', 'eat', 'eat']
```

Experiment-3

Implement Stop Word Removal using python

```
from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

text = "This is a sample sentence with some stop words like a, the, and is."

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

word_tokens = word_tokenize(text)

filtered_words = [w for w in word_tokens if not w.lower() in stop_words]

print("Original tokens:", word_tokens)

print("Filtered tokens (stop words removed):", filtered_words)
```

OUTPUT:

Original tokens: ['This', 'is', 'a', 'sample', 'sentence', 'with', 'some', 'stop', 'words', 'like', 'a', ',', 'the', ',', 'and', 'is', '.'] Filtered tokens (stop words removed): ['This', 'sample', 'sentence', 'stop', 'words', 'like', ',', ',', ',', '.']

Experiment-4

Implement Word Analysis using python

```
from nltk.tokenize import word_tokenize
from collections import Counter
text = "This is a sample sentence. This sentence is a sample."
tokens = word_tokenize(text.lower())
word_frequency = Counter(tokens)
print("Word Frequencies:", word_frequency)
```

OUTPUT:

```
Word Frequencies: Counter({'this': 2, 'is': 2, 'a': 2, 'sample': 2, 'sentence': 2, '.': 2})
```

Experiment-5

Implement Word Generation using python

```
def simple_pluralize(word):
    if word.endswith('y'):
        return word[:-1] + 'ies'
    elif word.endswith('s'):
        return word + 'es'
    else:
        return word + 's'
words = ["cat", "dog", "fly", "bus"]
plural_words = [simple_pluralize(word) for word in words]
print("Original words:", words)
print("Pluralized words:", plural_words)
```

OUTPUT:

```
Original words: ['cat', 'dog', 'fly', 'bus']
Pluralized words: ['cats', 'dogs', 'flies', 'buses']
```

Experiment-6

Implement Part-of-Speech (POS) Tagging using python

```
import nltk

from nltk.tokenize import word_tokenize

text = "The quick brown fox jumps over the lazy dog."

tokens = word_tokenize(text)

pos_tags = nltk.pos_tag(tokens)

print("Tokens with POS tags:", pos_tags)
```

OUTPUT:

Tokens with POS tags: [('The', 'DT'), ('quick', 'JJ'), ('brown', 'JJ'), ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('lazy', 'JJ'), ('dog', 'NN'), ('.', '.')]

Experiment-7

Implement Morphology using python

```
from nltk.stem import WordNetLemmatizer

from nltk.tokenize import word_tokenize

lemmatizer = WordNetLemmatizer()

words = ["running", "runs", "ran", "runner", "eating", "eats"]

lemmatized_words = [lemmatizer.lemmatize(word, pos='v') for word in words] # Lemmatize as verbs

print("Original words:", words)

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

words_noun = ["corpora", "boxes", "children"]

lemmatized_nouns = [lemmatizer.lemmatize(word, pos='n') for word in words_noun] # Lemmatize as nouns

print("\nOriginal nouns:", words_noun)

print("Lemmatized nouns:", lemmatized_nouns)
```

OUTPUT:

Original words: ['running', 'runs', 'ran', 'runner', 'eating', 'eats'] Lemmatized words (verbs): ['run', 'run', 'run', 'runner', 'eat', 'eat'] Original nouns: ['corpora', 'boxes', 'children'] Lemmatized nouns: ['corpus', 'box', 'child']

Experiment-8

Implement Chunking (Shallow Parsing) using python

```
import nltk

from nltk.tokenize import word_tokenize

text = "The quick brown fox jumps over the lazy dog."

tokens = word_tokenize(text)

tagged = nltk.pos_tag(tokens)

grammar = "NP: {<DT>?<JJ>*<NN>}"

chunk_parser = nltk.RegexpParser(grammar)

chunked = chunk_parser.parse(tagged)

print("Chunked output:")

print(chunked)
```

OUTPUT:

Chunked output: (S (NP The/DT quick/JJ brown/JJ fox/NN) jumps/VBZ over/IN (NP the/DT lazy/JJ dog/NN) ./.)

Experiment-9

Implement N-Grams using python

```
from nltk.util import ngrams

from nltk.tokenize import word_tokenize

text = "This is a simple example of n-grams."

tokens = word_tokenize(text)

bigrams = list(ngrams(tokens, 2))

print("Bigrams:", bigrams)

trigrams = list(ngrams(tokens, 3))

print("Trigrams:", trigrams)
```

OUTPUT:

Bigrams: [('This', 'is'), ('is', 'a'), ('a', 'simple'), ('simple', 'example'), ('example', 'of'), ('of', 'n-grams'), ('n-grams', '.')] Trigrams: [('This', 'is', 'a'), ('is', 'a', 'simple'), ('a', 'simple', 'example'), ('simple', 'example', 'of'), ('example', 'of', 'n-grams'), ('of', 'n-grams', '.')]

Experiment-10

Implement N-Grams Smoothing using python

```
from nltk.util import ngrams
from nltk.tokenize import word_tokenize
from collections import defaultdict
def calculate_ngram_probabilities_with_add_one(text, n):
    tokens = word_tokenize(text.lower())
    all_ngrams = list(ngrams(tokens, n))
    ngram_counts = defaultdict(int)
    context_counts = defaultdict(int)
    vocabulary_size = len(set(tokens))
    for ngram in all_ngrams:
        ngram_counts[ngram] += 1
        if n> 1:
            context = ngram[:-1]
            context_counts[context] += 1
        elif n == 1:
            context_counts[()] += 1 # Empty tuple for unigrams
    ngram_probabilities = {}
    for ngram, count in ngram_counts.items():
        context = ngram[:-1] if n > 1 else ()
        ngram_probabilities[ngram] = (count + 1) / (context_counts[context] + vocabulary_size)
    return ngram_probabilities
```

```
text = "This is a simple example. This is another example."

bigram_probabilities=calculate_ngram_probabilities_with_add_one(text, 2)

print("Bigram Probabilities (with Add-one Smoothing):")

for ngram, prob in bigram_probabilities.items():

    print(f"{ngram}: {prob:.4f}")
```

OUTPUT:

Bigram Probabilities (with Add-one Smoothing): ('this', 'is'): 0.2000 ('is', 'a'): 0.2000 ('a', 'simple'): 0.2000 ('simple', 'example'): 0.2000 ('example', '.'): 0.2000 ('.', 'this'): 0.2000 ('another', 'example'): 0.2000 ('is', 'another'): 0.2000