Roll No. _____

NATIONAL UNIVERSITY OF MODERN LANGUAGES ISLAMABAD
DEPARTMENT OF COMPUTER SCIENCE
Open Ended Lab EXAMINATION – Spring 2025 – BSAI-VI (Morning)

Paper: Natural Language Processing	**Instructor: Dr. QuaratulAin**
Time allowed: 3 Hours	**Total Marks: 100**

Instructions: -

You are quired to upload the following files:

1. Separate python source files for each question, where the files names must be yourRollNo_q1, yourRollNo_q2, yourRollNo_q3. Submit in a zip folder.
2. A pdf file which must contain each question no., each task description, screenshot of output, and the explanation if asked in the task/question. The pdf file name must be yourRollNo_OELNLPS25
3. Clearly mention the question/task number before attempting it.
4. Since NO LATE SUBMISSION will be accepted at all, therefore try to complete and upload your work well before time.
5. You will present your work, and a viva will be conducted in the upcoming theory class this week.

Implementation: 90 Marks

Viva: 10 Marks

Tools: Students should use Python with libraries like NLTK, spaCy, or scikit-learn.

Time Allocation: Medium tasks (5 marks) should take ~10–15 minutes each, and difficult tasks (15 marks) ~30–40 minutes each, totaling ~3 hours for the exam.

QUESTION 1 [CLO-1, P-3, PLO-2] [Medium Tasks (5 marks each)]

Marking Scheme (5 marks each)

- **3 marks: Correct implementation of the core task.**
- **1 mark: Proper use of libraries and syntax.**
- **1 mark: Accurate output or explanation.**

Task 1: Tokenization

Scenario: You are analyzing customer reviews for a product. Given the text: "I love this product! It's amazing and works great.", tokenize the sentence into individual words.

Task: Write a Python code snippet using NLTK to tokenize the text and print the tokens.

Task 2: Morphemes in the Word

Scenario: You are studying the structure of words in a linguistic analysis project. Identify the morphemes in the word "unhappiness".

Task: Manually break down "unhappiness" into its morphemes and explain each part briefly.

Task 3: Stemming

Scenario: You are preprocessing text data for a search engine. Given the words ["running", "runner", "runs"], apply the Porter Stemmer to standardize them.

Task: Write a Python code using NLTK to stem the words and print the stemmed forms.

Task 4: Lemmatization

Scenario: You are building a chatbot that needs to understand word meanings. Given the words ["better", "running", "mice"], lemmatize them to their base forms.

Task: Write a Python code using spaCy or NLTK to lemmatize the words and print the results.

Task 5: Overstemming

Scenario: You notice that a stemming algorithm is overstemming words, reducing "organization" and "organize" to the same stem "organ". Explain the concept of overstemming and provide an example of how it can affect text analysis.

Task: Define overstemming and give a specific example with two words that might be overstemmed, explaining the impact.

Task 6: Part-of-Speech Tagging
Scenario: You are analyzing a sentence for a grammar checker: "The cat sleeps peacefully."
Task: Write a Python code using NLTK to perform POS tagging on the sentence and print the tagged output.

Task 7: Filter Out Non-Alphabetic Characters
Scenario: You are cleaning text data from social media: "Hello @user! How are you? 123 #happy".
Task: Write a Python code to filter out all non-alphabetic characters and print the cleaned text.

Task 8: Check for Profanity/Offensive Language
Scenario: You are moderating a forum and need to flag offensive content. Given the text: "This is a great day, damn it!", check if it contains profanity.
Task: Write a Python code using a simple profanity filter (e.g., a predefined list) to detect offensive words and print a message indicating whether profanity is found.

Task 9: One Hot Encoding
Scenario: You have a dataset with categorical labels ["cat", "dog", "bird"] for a classification task.
Task: Write a Python code using scikit-learn to perform one-hot encoding on these labels and print the encoded matrix.

Task 10: Bag of Words
Scenario: You are analyzing movie reviews: ["good movie", "bad acting", "good plot"].
Task: Write a Python code using scikit-learn to create a Bag of Words representation and print the feature matrix.

Task 11: TF-IDF
Scenario: You have documents: ["the cat", "the dog", "the cat and dog"]. Compute the TF-IDF scores.
Task: Write a Python code using scikit-learn to calculate TF-IDF values and print the matrix.

Task 12: N-Grams
Scenario: You are analyzing a sentence for text analysis: "I love to code".
Task: Write a Python code using NLTK to extract bi-grams (2-grams) and tri-grams (3-grams) from the sentence and print them.

QUESTION 2 [CLO-2, P-6, PLO-5] [15-Marks][Difficult Task]

Marking Scheme (15 Marks)

- **8 Marks**: Correct implementation of all steps (tokenization, N-gram generation, filtering, frequency computation, ranking).
- **4 Marks**: Optimization and edge case handling (e.g., efficient data structures, cleaning punctuation).
- **3 Marks**: Quality of analysis and explanation of how N-grams contribute to summarization, including the impact of filtering.

Advanced N-Gram Analysis for Text Summarization (15 Marks)
Scenario: You are working on a text summarization project for a news aggregation platform. The goal is to identify the most frequent and meaningful phrases in a news article to generate a concise summary. Given a news snippet: "The government announced new policies on climate change. Climate change policies will impact industries significantly. Industries must adapt to new regulations.", you decide to use N-grams (specifically bi-grams and tri-grams) to extract key phrases. However, not all N-grams are meaningful—some may include stop words (e.g., "the government") or irrelevant combinations. Your task is to generate, filter, and rank N-grams to identify the most significant phrases for summarization.
Task: Write a Python code to perform the following steps:

1. Tokenize the given text into words.
2. Generate bi-grams (2-grams) and tri-grams (3-grams) using NLTK.

3. Filter out N-grams that contain stop words (use NLTK's stop word list) to focus on meaningful phrases.
4. Compute the frequency of the filtered N-grams.
5. Rank the N-grams by frequency and select the top 3 bi-grams and top 2 tri-grams.
6. Analyze the selected N-grams and briefly explain (in 2–3 sentences) how they could be used to create a summary of the text.

Optimize the code by ensuring it can handle larger texts efficiently (e.g., use appropriate data structures like dictionaries or Counter).

QUESTION 3 [CLO-2, P-6, PLO-5] [15-Marks] [Difficult Task]

Marking Scheme (15 Marks)

- **8 marks:** Correct implementation of all required steps.
- **4 marks:** Handling edge cases or additional analysis.
- **3 marks:** Clear explanation and justification.

Comparative Analysis of Bag of Words and TF-IDF

Scenario: You are evaluating text representation techniques for a document classification task. You have three documents: ["good movie", "bad acting", "good plot bad movie"]. Compare the Bag of Words and TF-IDF representations to determine which might better capture the importance of terms for classification.

- **Task**:
    1. Write a Python code using scikit-learn to compute the Bag of Words and TF-IDF matrices for the documents.
    2. Print both matrices and analyze which method highlights the term "bad" more effectively given its varying importance across documents.

Provide a brief justification (2–3 sentences) for which method you would choose for this task and why.
