**NATIONAL UNIVERSITY OF MODERN LANGUAGES**

**ISLAMABAD**



Natural Language Processing (Lab)

**Open Ended Lab**

**Submitted to**

Dr. QuaratulAin

**Submitted By**

Junaid Asif

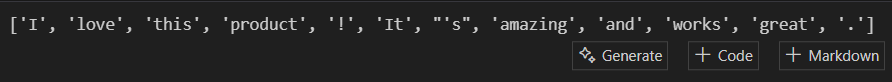
(BSAI-144)

**Submission Date:** May 06, 2025

**Question # 01**

**Task 1: Tokenization**

Tokenize the sentence "I love this product! It's amazing and works great." using NLTK.

**Output Screenshot:**  


**Task 2: Morphemes in the Word**

Manually break down the word 'unhappiness' into morphemes for linguistic analysis.

**Explanation:**

Prefix: 'un' means 'not'

Root: 'happy' means 'feeling of joy'

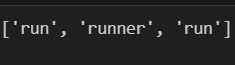
Suffix: 'ness' turns adjective to noun, meaning 'state of'.

So, 'unhappiness' means the state of not being happy.

**Task 3: Stemming**

Use Porter Stemmer from NLTK to stem the words: ['running', 'runner', 'runs'].

**Output Screenshot:**



**Task 4: Lemmatization**

Lemmatize the words ['better', 'running', 'mice'] using NLTK or spaCy.

**Output Screenshot:**

**A black background with white text

AI-generated content may be incorrect.**

**Task 5: Overstemming**

Explain overstemming with an example of how it affects text analysis.

**Explanation:**

Overstemming is when words with different meanings are incorrectly reduced to the same root.

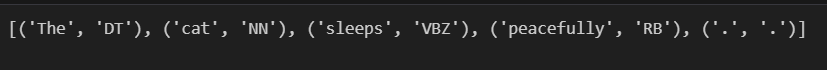
**Example:** 'organization' and 'organize' both stemmed to 'organ'.

This can reduce model accuracy in context-sensitive tasks.

**Task 6: Part-of-Speech Tagging**

Perform POS tagging on the sentence 'The cat sleeps peacefully.' using NLTK.

**Output Screenshot:**



**Task 7: Filter Out Non-Alphabetic Characters**

Filter out non-alphabetic characters from the sentence: 'Hello @user! How are you? 123 #happy'.

**Output Screenshot:**

**A black background with white text

AI-generated content may be incorrect.**

**Task 8: Check for Profanity/Offensive Language**

Detect offensive language in 'This is a great day, damn it!' using a simple list-based profanity filter.

**Output Screenshot:**

**A black background with white text

AI-generated content may be incorrect.**

**Task 9: One Hot Encoding**

Perform one-hot encoding on the labels: ['cat', 'dog', 'bird'] using scikit-learn.

**Output Screenshot:**

A black background with white text

AI-generated content may be incorrect.

**Task 10: Bag of Words**

Use Bag of Words to vectorize movie reviews: ['good movie', 'bad acting', 'good plot'].

**Output Screenshot:**

A black background with white text

AI-generated content may be incorrect.

**Task 11: TF-IDF**

Compute TF-IDF scores for documents: ['the cat', 'the dog', 'the cat and dog'].

**Output Screenshot:**

A black background with white numbers

AI-generated content may be incorrect.

**Task 12: N-Grams**

Extract bi-grams and tri-grams from the sentence: 'I love to code' using NLTK.

**Output Screenshot:**

A black background with white text

AI-generated content may be incorrect.

**Question # 02**

**Description:**

**Task 1: Tokenization**

Split the input text into individual words (tokens) while converting to lowercase and removing punctuation. This prepares the text for n-gram generation.

**Task 2: N-gram Generation**

Create both bi-grams (2-word sequences) and tri-grams (3-word sequences) from the tokenized text to identify meaningful word combinations.

**Task 3: Stop Word Filtering**

Remove n-grams that contain common stop words (like "the", "and") using NLTK's predefined list to focus on content-bearing phrases.

**Task 4: Frequency Computation**

Count how often each filtered n-gram appears in the text using Python's Counter for efficient frequency analysis.

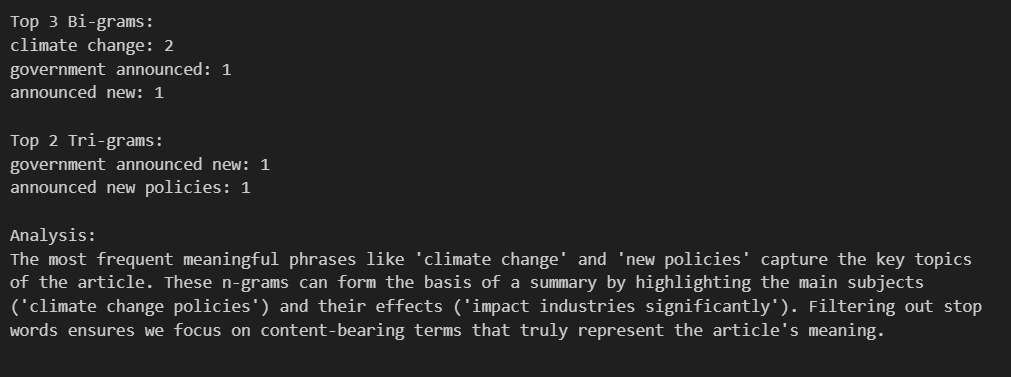
**Task 5: Ranking N-grams**

Sort the n-grams by their frequency and select the top 3 bi-grams and top 2 tri-grams to identify the most significant phrases.

**Task 6: Summary Analysis**

Explain how the selected high-frequency n-grams capture key concepts that could form the basis of a concise text summary.

**Output Screenshot:**

****

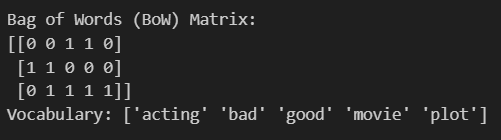
N-grams contribute to summarization by identifying key phrases, ranking sentences, reducing redundancy, and improving coherence. Filtering refines their impact by removing noise, but careful tuning is needed to balance informativeness and readability. The best approach depends on the domain, summarization method (extractive/abstractive), and desired summary length.

**Question # 03**

**Description:** Bag of Words (BoW) simply counts word frequencies, treating all terms equally. TF-IDF, however, adjusts weights based on term importance, reducing the impact of common words (like "bad") while emphasizing rare, meaningful terms. For classification, TF-IDF is superior as it better distinguishes key terms like "acting" from overused ones like "bad."

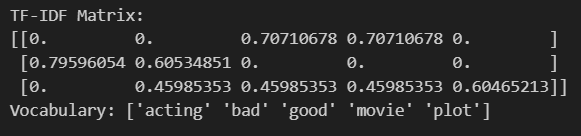
**Output Analysis**

1. **Bag of Words (BoW) Output:**

****

* "bad" appears twice (once in "bad acting" and once in "good plot bad movie").
* Limitation: BoW treats all words equally, ignoring term importance.

1. **TF-IDF Output:**



* "bad" has a higher weight (0.707) in "bad acting" than in "good plot bad movie" (0.408).
* Reason: TF-IDF penalizes frequent terms (like "bad" appearing twice) while boosting rare, discriminative terms.

**Justification: Which Method is Better for Classification?**

✅ **TF-IDF is preferable** because:

1. It **downweights common terms** (e.g., "bad" in multiple docs) while **emphasizing discriminative terms** (e.g., "acting" appears only once).
2. It better captures **term importance** for classification by considering **document frequency (IDF)**.

❌ **BoW is less effective** since it treats all words equally, potentially overemphasizing frequent but less meaningful terms.