Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2025-26

Class:	**BE-AI&DS**	**Semester:**	**VII**
Course Code:	**CSDOL7011**	**Course Name:**	**NLP Lab**

Name of Student:	BARI ANKIT VINOD
Roll No. :	61
Experiment No.:	**4**
Title of the Experiment:	**Building Unigram, Bigram, and Trigram Language Models for Word Prediction**
Date of Performance:	
Date of Submission:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

CSDOL7011: Natural Language Processing Lab

Checked by

Name of Faculty :
Signature :
Date :

Aim: To build and evaluate N-gram language models (Unigram, Bigram, and Trigram) to analyze word-level dependencies in text.

Objective: To build N-gram language models for predicting words based on context and analyzing word-level dependencies.

Tools Required:

1. Python (Jupyter Notebook or Google Colab)
2. Nltk
3. collections (for frequency counting)
4. random (for sampling words)

Procedure:

1. Import libraries:
    a. Import nltk, collections.Counter, and any preprocessing modules.

2. Load a sample corpus:
    a. Use nltk.corpus.gutenberg, brown, or a custom dataset.
    b. Preprocess the text (lowercasing, tokenization, stopword removal optional).

3. Build N-gram models:
    a. Unigram: Compute frequency of each word.

b. Bigram: Count frequency of consecutive word pairs.
c. Trigram: Count frequency of word triplets.

4. Estimate probabilities:

For bigrams: $P(w_n|w_{n-1}) = \frac{\text{Count}(w_{n-1},w_n)}{\text{Count}(w_{n-1})}$

For trigrams: $P(w_n|w_{n-2}, w_{n-1}) = \frac{\text{Count}(w_{n-2},w_{n-1},w_n)}{\text{Count}(w_{n-2},w_{n-1})}$

5. Implement word prediction:
   a. Prompt user to input one or two words.
   b. Predict the next word using the most probable bigram/trigram.

6. Evaluate results:

Print predicted words or top-k probable next words.

Description of the Experiment:

This experiment introduces students to statistical language modeling using N-grams. By computing word sequence probabilities, students observe how simple models can predict the likelihood of a word given its context. This helps them understand foundational ideas behind more advanced models like RNNs or transformers.

Detailed Description of the NLP Technique:

N-gram Language Models:

An N-gram is a contiguous sequence of n words. N-gram language models estimate the probability of a word given its n-1 previous words.

Types:

Unigram Model (n=1): Assumes each word is independent.

Bigram Model (n=2): Considers one preceding word.

Trigram Model (n=3): Considers two preceding words.

Formula for Probability Estimation:

Unigram: $P(w_n) = \frac{\text{Count}(w_n)}{N}$

Bigram: $P(w_n|w_{n-1})$

Trigram: $P(w_n|w_{n-2}, w_{n-1})$

Challenges with N-gram Models:

1. Data sparsity: Many possible combinations may never occur in training data.
2. Smoothing: Techniques like Laplace Smoothing are used to assign non-zero probabilities to unseen N-grams.
3. Limited context: Longer dependencies can't be captured effectively.

Applications:

1. Word prediction
2. Spelling correction
3. Speech recognition

Conclusion:

The results demonstrate that Natural Language Processing techniques can effectively perform tasks such as word prediction, spelling correction, and speech recognition. The models were able to predict the next word in a sequence with good contextual accuracy, identify and correct common spelling errors, and accurately convert spoken input into text.

These outcomes show the power of combining linguistic rules with statistical and deep learning methods to process and understand human language. Overall, the experiment highlights how NLP models can improve communication between humans and machines by making text and speech interactions more intelligent and context-aware.