INDIAN INSTITUTE OF INFORMATION TECHNOLOGY KOTTAYAM
Department of Electronics and Communication Engineering
Assignment I

ECS 421 Natural Language Processing

Submit on or before **27-02-2026** (hard copy and lms upload)

1. Consider a toy tag set $\{N, V, A\}$. The Hidden Markov Model (HMM) for POS tagging is defined using the following initial, transition, and emission probabilities.

   The initial probabilities of the tags (in the order N, V, A) are given as:

   $$P(N) = 0.4, \quad P(V) = 0.35, \quad P(A) = 0.25$$

   Tag-transition probabilities:

From \ To	N	V	A
N	0.3	0.5	0.2
V	0.4	0.4	0.2
A	0.5	0.3	0.2

   Word emission probabilities:

Tag	book	light
N	0.6	0.2
V	0.2	0.5
A	0.3	0.4

   Using the Viterbi algorithm, find the most probable part-of-speech (POS) tag sequence for the observed sequence:

   "light book light"

   Show the dynamic programming table and the backtrace clearly.

2. Consider the following three sentences:
   < s > The cat chased the mouse < /s >
   < s > The dog barked loudly < /s >
   < s > The bird flew away < /s >

   Assuming a bigram language model, what is the probability of the sentence:
   < s > The cat barked loudly < /s >?

   Assume a small probability of 0.001 for words/ngram not in the training data.

3. Compute the minimum edit distance to transform

   kitten → sitting

   using the edit operations insertion, deletion, and substitution. Assume insertion and deletion cost 1, and substitution cost 2. Find the minimum edit distance, align the two strings, and show the backtrace.

4. (a) Two corpora contain the same number of tokens. Corpus-A has 3,000 unique word types, whereas Corpus-B has 1,200 unique word types. Without calculating TTR explicitly, discuss which corpus is likely to be more lexically diverse and justify your reasoning.

(b) In a corpus following Zipf-like distribution, a word at rank 2 has frequency 500. Estimate the rank of a word whose frequency is approximately 100.

5. Write a Python program to perform basic text preprocessing on a given paragraph.

   Your program must perform the following operations:

   1. Convert the text to lowercase.
   2. Remove punctuation marks and numbers.
   3. Tokenize the text into words.
   4. Remove English stopwords.

   Expected Output:

   - Original text
   - Tokens before preprocessing
   - Cleaned tokens after preprocessing
   - Total number of tokens before and after preprocessing

6. Write a Python program using NLTK to compare stemming and lemmatization.

   Apply the following methods:

   - Porter Stemmer
   - WordNet Lemmatizer

   Use the following list of words:

   `running, studies, better, leaves, flying, wolves`

   Expected Output:

   - Display results in a table with the following columns:

     Word Stemmed Lemmatized

   - Write two observations comparing stemming and lemmatization results.

7. Implement the Minimum Edit Distance algorithm using dynamic programming.

   Transform the string

   `intention` → `execution`

   Use the following edit costs:

   - Insertion = 1
   - Deletion = 1
   - Substitution = 2

   Expected Output:

   - Dynamic programming matrix
   - Minimum edit distance value
   - Optimal alignment of the two strings

- Sequence of edit operations obtained using backtrace

8. Write a Python program to build a Bigram Language Model from a given corpus.

   Your program should:

   1. Compute unigram and bigram counts.
   2. Calculate bigram probabilities.
   3. Compute the probability of a given input sentence.
   4. Implement Laplace (add-one) smoothing.
   5. Compare probabilities with and without smoothing.

   Expected Output:

   - Unigram frequency table
   - Bigram frequency table
   - Sentence probability without smoothing
   - Sentence probability with Laplace smoothing
   - Probability values for unseen bigrams

9. Implement the Viterbi algorithm for Part-of-Speech (POS) tagging using a Hidden Markov Model.

   Given:

   - Initial tag probabilities
   - Transition probability matrix
   - Emission probabilities
   - Observation sequence (sentence)

   Write a Python program to:

   1. Compute the dynamic programming (delta) table.
   2. Store backpointer (psi) values.
   3. Perform backtrace.
   4. Find the most probable POS tag sequence.

   Expected Output:

   - Delta table
   - Backpointer table
   - Most probable POS tag sequence
   - Final probability of the best path