Lab 4: Revision and Word Sense Disambiguation (WSD)

Date & Time: 29 July 2025, 12:45 PM

Question 1: Positional Index

Program Description:

Build a positional index for one or more documents to track the positions of each term in each document.

Logic / Algorithm:

1. Tokenize each document into words.
2. For each word and each document, record the positions (indices) where the word occurs.
3. Store in a dictionary of the form `{term: {docID: [pos1, pos2, ...], ...}, ...}`.
4. Provide lookup functionality to retrieve positional lists for given terms

Question 2: Word Matrix

Program Description:
Construct a term-document incidence matrix indicating presence (1) or absence (0) of terms in each document.

Logic / Algorithm:

1. Compile the set of all unique terms across the documents.
2. For each term and each document, check if the term appears.
3. Populate matrix entries with 1 for presence, 0 for absence.

Question 3: Linguistic Preprocessing

Program Description:
Perform tokenization, normalization, stemming, lemmatization, and frequency analysis on input text.

Logic / Algorithm:

1. Tokenize text into lowercase tokens.
2. Apply stemming (`PorterStemmer`) and lemmatization (`WordNetLemmatizer`).
3. Count token frequencies and sort by frequency.
4. *(Additional)* Compute edit distance between two user-specified words.

Question 4: Levenshtein Edit Distance

Program Description:
Visualize and compute the minimum edit distance between two words using dynamic programming.

Logic / Algorithm:

1. Initialize DP matrix of size `(len(a)+1) x (len(b)+1)` with base costs for insertions/deletions.
2. Fill matrix: cost 0 for matches, 1 for substitutions/insertions/deletions.
3. Trace back from bottom-right to reconstruct alignment and operations.
4. Summarize total distance and counts of matches, substitutions, insertions, deletions.

Question 5: POS Tagging with HMM

Program Description:
Train a simple Hidden Markov Model on a small corpus to perform POS tagging.

Logic / Algorithm:

1. Tokenize and POS-tag training sentences.

2. Estimate emission probabilities `P(word | tag)` and transition probabilities `P(tag_i | tag_{i-1})`.

3. Display probability tables for inspection.

Question 6: Word Sense Disambiguation (WSD)

Program Description:
Disambiguate senses of open-class words in a sentence using the Lesk algorithm and WordNet.

Logic / Algorithm:

1. Tokenize and POS-tag the input sentence; select open-class words (nouns, verbs, adjectives, adverbs).

2. Retrieve all synsets (senses) for each word from WordNet.

3. Apply the Lesk algorithm: choose the sense with maximum overlap between context and sense definition.

4. Present all candidate senses and the selected sense.