NLP Assignment-1

Dr. Dheeraj

March 6, 2025

(Roll Nos: 1-5)

1. **Develop an NLP pipeline that performs:**
   a) Text preprocessing.
   b) Given a word and its possible meanings, select the correct sense based on a sentence.
   c) Compare the overlapping words in the sentence and the sense definition.
   d) Read the research papers.

Text Preprocessing

- **Input:** "The quick brown fox jumps over the lazy dog!"
- **Steps (Examples):**
  1. **Tokenization:** ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', '!']
  2. **Lowercasing:** ['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', '!']
  3. **Stopword Removal:** ['quick', 'brown', 'fox', 'jumps', 'lazy', 'dog', '!']
  4. **Lemmatization:** ['quick', 'brown', 'fox', 'jump', 'lazy', 'dog']
- **Output:** ['quick', 'brown', 'fox', 'jump', 'lazy', 'dog']

Word Sense Disambiguation

- **Input Sentence:** "He sat on the bank of the river."
- **Word to Disambiguate:** "bank"
- **Possible Meanings:**
    1. **bank (financial institution):** A financial establishment.
    2. **bank (river edge):** The slope beside a body of water.
- **Context Words:** "sat", "river"
- **Selected Sense: bank (river edge)**

Overlapping Words Comparison

- **Input Sentence:** "He sat on the bank of the river."
- **Word Definition:** "The slope beside a body of water."
- **Extracted Words from Sentence:** ["sat", "bank", "river"]
- **Extracted Words from Definition:** ["slope", "body", "water"]
- **Common Words:** "river" matches with "water" (semantic similarity)
- **Number of Overlapping Words:** 1
- **Similarity Score: 0.3 (depending on metric used)**

(Roll Nos: 6-10)

2. Develop an NLP pipeline that performs:

- **(c)** Generate Text N-Grams Without Using NLTK:
  - Implement n-gram extraction (bi-grams, tri-grams, etc.) from a given text.
- **(d)** Ignore stopwords and punctuation in the process.

Task (c): Generate Text N-Grams

Input:

- A given long text (e.g., `"The quick brown fox jumps over the lazy dog."`).
- The value of **N** (e.g., 2 for bigrams, 3 for trigrams).

Process:

- Tokenize the text manually (split into words).
- Ignore punctuation.
- Generate n-grams by forming consecutive sequences of words of size N.

Output:

- List of n-grams.
- **Example (bigrams)**: `[("The", "quick"), ("quick", "brown"), ("brown", "fox"), ...]`
- **Example (trigrams)**: `[("The", "quick", "brown"), ("quick", "brown", "fox"), ...]`

Task (d): Ignore Stopwords and Punctuation

Input:

- The same input text.
- List of stopwords (e.g., `["the", "is", "and", "over", "in", "on"]`).

Process:

- Remove stopwords and punctuation before generating n-grams.

Output:

- N-grams **without stopwords and punctuation**.
- **Example Before Stopword Removal:** `[("The", "quick"), ("quick", "brown"), ("brown", "fox")]`
- **Example After Stopword Removal:** `[("quick", "brown"), ("brown", "fox")]`

(Roll Nos: 11-15)

Task 3: Develop an NLP pipeline that performs:

1. Compute TF-IDF scores for each word in a document.
2. Return the top N keywords with the highest scores.

Step (c): Compute TF-IDF Scores

Input:

- A document (text data).
- Preprocessed text (tokenized, lowercased, stopwords removed).

Processing:

- Compute Term Frequency (TF):

$$TF = \frac{\text{Count of word in document}}{\text{Total words in document}}$$

- Compute Inverse Document Frequency (IDF):

$$IDF = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing the word}}\right)$$

- Calculate TF-IDF:

$$TF - IDF = TF \times IDF$$

Output: List of words with their TF-IDF scores.

Step (d): Extract Top N Keywords

Input:

- List of words with TF-IDF scores.
- Value of N (number of top keywords).

Processing:

- Sort words by TF-IDF scores in descending order.
- Select the top N highest-scoring words.

Output: A ranked list of top N keywords.

Example: TF-IDF Calculation

Input Document:
"Natural Language Processing (NLP) is a field of AI that focuses on the interaction between humans and computers using natural language. NLP techniques include tokenization, parsing, and named entity recognition." **Output (Top 3 Keywords)**

Keyword	**TF-IDF Score**
NLP	0.89
Natural Language	0.75
Processing	0.68

(Roll Nos: 16-20)

4. Develop an NLP pipeline that performs:

- (c) Implement extractive text summarization by scoring sentences based on:
  - Word frequency
  - Sentence length
  - TF-IDF scores
- (d) Find the most frequent POS (Part of Speech) tag in a given text.

(c) Extractive Text Summarization - Hint

Input:

- A large text document (e.g., an article, research paper, or long paragraph).

Processing:

- Tokenization (split text into sentences and words).
- Compute word frequency, sentence length, and TF-IDF scores.
- Rank sentences based on scores.
- Select the top-ranked sentences for the summary.

Output:

- A concise summary containing the most important sentences.

Example: Extractive Summarization

Input Text: *"Natural Language Processing (NLP) is a field of AI that focuses on the interaction between humans and computers using natural language. It enables applications such as text summarization, machine translation, and chatbots."*
Extracted Summary: *"NLP is a field of AI focusing on human-computer interaction. It enables text summarization, translation, and chatbots."*

(d) Most Frequent POS Tag - Hint

Input:

- A given text document or paragraph.

Processing:

- Tokenize text into words.
- Perform POS tagging using NLP libraries.
- Count occurrences of each POS tag.
- Identify the most frequent POS tag.

Output:

- The most frequent POS tag in the text.

Example: Most Frequent POS Tag

Input Text: *"The cat sat on the mat and looked at the window."*
POS Tags Count:

- Nouns (NN): 3
- Verbs (VB): 2
- Determiners (DT): 2

Output: Most frequent POS tag: **Noun (NN)**

(Roll Nos: 21-25)

5. Develop an NLP pipeline that performs:

1. **(c)** Apply **Byte Pair Encoding (BPE)** to compress text data and reduce vocabulary size.
2. **(d)** Given a BPE-encoded text, reconstruct the original words by merging subwords.

Byte Pair Encoding (BPE) for Text Compression

Step (c): Apply BPE to Reduce Vocabulary Size

- **Input:** A large text corpus (e.g., thousands of sentences).
- **Hyperparameter - Merge Operations:**
  - Defines how many times frequent subwords are merged.
  - Higher merges $\Rightarrow$ Larger subwords, smaller vocabulary.
  - Lower merges $\Rightarrow$ Smaller subwords, larger vocabulary.
- **Processing:**
  1. Tokenize text into individual characters or subwords.
  2. Identify and merge the most frequent adjacent subwords **N times**.
  3. Store the learned subword vocabulary.
- **Output:**
  - Compressed text with subword units.
  - Vocabulary of subwords.

Example of BPE Encoding

Original Sentence: `"learning NLP is useful"`
After 2 BPE Merge Operations:

- `"learn@@ ing NLP is use@@ ful"`

Reconstructing Original Words from BPE

Step (d): Decode BPE-encoded text

- **Input:** BPE-encoded text and learned merge rules.
- **Processing:**
  1. Reverse the BPE process.
  2. Merge subwords iteratively using learned rules.
- **Output:** Original words restored.

Example:

- **Input:** `"learn@@ ing NLP is use@@ ful"`
- **Output:** `"learning NLP is useful"`

6. (Roll Nos: 26-33)

6. Develop an NLP pipeline that performs:

(c) Find the most frequently occurring word in a text, excluding common stopwords.

(d) Find the most semantically similar sentence to a given input using TF-IDF.

Task (c): Most Frequent Word (Excluding Stopwords)

Input:

- A text document (long paragraph or multiple sentences).
- A list of stopwords (e.g., from NLTK or SpaCy).

Processing Steps:

- Tokenize text into words.
- Convert words to lowercase and remove punctuation.
- Remove stopwords.
- Count word occurrences and find the most frequent word.

Output: The most frequently occurring word (excluding stopwords).

Example for Task (c)

Input: *"The cat sat on the mat. The cat is happy. The dog is happy too."*
After Removing Stopwords: *"cat sat mat. cat happy. dog happy."*
Output: "cat" (Most frequently occurring word)

Task (d): Sentence Similarity using TF-IDF

Input:

- A collection of sentences (text corpus).
- A query sentence to compare.

Processing Steps:

- Compute TF-IDF vectors for all sentences in the corpus.
- Compute TF-IDF vector for the query sentence.
- Calculate cosine similarity between query and each sentence.
- Select the most similar sentence.

Output: The sentence with the highest similarity score.

Example for Task (d)

Corpus Sentences:

1. "The sky is blue and beautiful."
2. "The weather is amazing today."
3. "The sun is shining brightly in the sky."

Query Sentence: *"It is a bright sunny day."*
Most Similar Sentence: *"The sun is shining brightly in the sky."*

Summary

Key Points:

- Task (c): Identify the most frequently occurring word after removing stopwords.
- Task (d): Find the sentence most similar to a query using TF-IDF and cosine similarity.