Task2: Advanced Text Preprocessing and Data Cleaning in Natural Language Processing (NLP)

Due Date: 12th October 2024
Total Marks: 10

Objective:

The primary objective of this task (composed of several sub-tasks) is to enhance students' understanding and programming skills in text preprocessing techniques commonly used in NLP. By breaking down the text cleaning process step by step, students will gain hands-on experience in applying different methods to clean, tokenize, and prepare text data for machine learning and deep learning models. These tasks will also encourage experimentation with custom cleaning techniques, regular expressions, and performance optimization, fostering a deeper understanding of real-world NLP workflows.

1: Apply the Provided Code to a New Dataset

Objective: Apply the given text-cleaning function (discussed in class and uploaded in Google Classroom) to a new dataset, such as movie reviews, social media comments, or product reviews.

1. **Task**: Load a different dataset, apply the clean_text() function, and analyze the results. Explain the cleaning process step by step as it applies to the new dataset.

2: Break Down the Cleaning Steps and Output

Objective: Understand each step in detail by applying them separately to small examples.

1. **Task**: Take an example sentence (e.g., "I bought 10 apples, and they were great!") and apply each line of the clean_text() function separately.
    - Show the output of each step.
    - Explain what happens in each transformation:
        - Convert to lowercase.
        - Tokenization and punctuation removal.
        - Remove numbers and stop words.
        - Apply POS tagging and lemmatization.
        - Remove single-letter words and join the text.
2. **Task Output**:
    - For example:
        - Original: "I bought 10 apples, and they were great!"
        - After lowercase: "i bought 10 apples, and they were great!"
        - After removing punctuation: ['i', 'bought', '10', 'apples', 'and', 'they', 'were', 'great']
        - After removing numbers: ['i', 'bought', 'apples', 'and', 'they', 'were', 'great']
        - After stop word removal: ['bought', 'apples', 'great']
        - After POS tagging and lemmatization: ['buy', 'apple', 'great']

- Final result: "buy apple great"

3: Investigate Stop Words and Modify the List

Objective: Understand the importance of stop words and their role in text cleaning.

1. **Task**: Modify the stop words list and observe how it affects the cleaning process.
    - Add or remove custom stop words.
    - Create a custom list of words to keep or remove, based on domain knowledge.
    - Compare the text output with different stop words removed.

4: Create a Custom Tokenizer

Objective: Experiment with different ways to tokenize text.

1. **Task**: Implement a custom tokenizer using regular expressions instead of the simple .split(" ") method.
    - Use a regex-based tokenizer to handle edge cases like contractions, punctuation, or special symbols.

5: Expand the Lemmatization Step

Objective: Work with custom parts of speech to expand the understanding of lemmatization.

1. **Task**: Use custom POS tagging or experiment with different lemmatization libraries.
    - Compare WordNetLemmatizer with other lemmatization techniques, such as Spacy's lemmatizer.
    - Experiment with different parts of speech and observe changes in the output.

6: Remove Different Types of Unwanted Tokens

Objective: Extend the text-cleaning process by targeting specific tokens for removal.

Task: Modify the text-cleaning function to remove additional unwanted tokens:

- Remove special characters (like emojis, symbols, etc.).
- Remove HTML tags or URL links from the text.
- Handle more specific cases by customizing the cleaning function.

7: POS Tagging Analysis

Objective: Analyze the effect of POS tagging on the lemmatization process.

Task: After POS tagging, observe how lemmatization behaves for different POS tags.

- Create a small set of sentences with varied parts of speech (noun, verb, adjective, adverb) and evaluate how the lemmatizer processes them.

8: Performance Evaluation

Objective: Evaluate the performance of the cleaning process using metrics like time complexity and effectiveness.

Task: Measure the time it takes to clean a large dataset using Python's time module to time each step of the clean_text() function.

- Analyze the computational cost of each step and optimize any inefficient processes.

9: Error Handling and Data Quality

Objective: Handle edge cases like missing data, null values, or improperly formatted text.

Task: Implement error handling in the clean_text() function to manage:

- Null values or empty strings.
- Very short sentences or very long sentences.
- Invalid or corrupt data.

10: Visualization and Frequency Analysis

Objective: Visualize the effect of text cleaning and perform frequency analysis on the cleaned data.

Task: After cleaning the dataset, visualize the frequency of words using word clouds or bar plots to show the most frequent words in the cleaned text.

- Analyze changes in word frequencies before and after cleaning.