REPORT

Part 1: Implementation of Byte Pair Encoding (BPE) Algorithm

Objective:

The primary goal of this assignment is to implement the Byte Pair Encoding (BPE) algorithm, a widely used subword tokenization technique. The focus is on:

1. Generating a vocabulary from input sentences.
2. Iteratively merging token pairs to optimize tokenization.
3. Evaluating the effectiveness of the tokenization process.

Introduction:

In modern Natural Language Processing (NLP), tokenization serves as a critical preprocessing step. Byte Pair Encoding (BPE) stands out as a subword tokenization method that balances vocabulary size with representational flexibility. This assignment focuses on understanding and implementing the BPE algorithm and analyzing its application in tokenization.

Implementation Details:

1. Class Definition:
    - A Python class `BPE` was created to encapsulate all functionalities of the BPE algorithm. By structuring the code into a class, modularity and reusability were ensured.
2. Key Methods:
    - `get_vocab`:
        - This method generates the initial vocabulary by splitting words into characters. Each word is appended with an end-of-word token (`</w>`), ensuring distinction between different word boundaries. The frequency of subword units is calculated using the `Counter` class, which simplifies counting operations.
    - `get_stats`:
        - This method computes the frequency of adjacent token pairs in the vocabulary. By scanning through all word sequences, it identifies the most frequent pairs that are candidates for merging.
    - `merge_vocab`:

        - The most frequent token pair is merged into a single unit, and the vocabulary is updated. This step gradually reduces the total number of tokens by iteratively combining frequently occurring pairs.
    - `train`:
        - The training process involves iteratively merging token pairs for a specified number of steps (`num_merges`). This method outputs a log of merge operations, providing insights into the evolution of the subword vocabulary.
    - `tokenize`:
        - The tokenize function uses the learned merge rules to segment new input words into subword units, ensuring that the segmentation aligns with the training process.
    - `evaluate`:
        - This method calculates performance metrics, such as:
            - Accuracy: Measures how well the learned subword vocabulary captures the original words.
            - Coverage: Examines the percentage of input words that can be represented using the learned tokens.
            - Precision, Recall, and F1-Score: Provide a comprehensive evaluation of the tokenization quality.
3. Sample Input and Output:

    - Input Sentences:
        - Example sentences like "low lower lowest" and "newer wider tallest" were used to validate the algorithm.
    - Output Analysis:
        - The initial vocabulary contained characters like `l</w>`, `o</w>`, `w</w>`, etc. With each merge operation, frequent pairs such as (`l`, `o`) were combined into subwords like `lo</w>`.
    - Metrics Results:
        - Accuracy was evaluated based on how closely the learned vocabulary matched the original dataset.
        - Coverage showed that most input words were well-represented within the new token set.

Results and Observations:

- The BPE algorithm effectively created a compact subword vocabulary through iterative merging of frequent token pairs.
- Higher accuracy and coverage scores were observed as the number of merge operations increased, demonstrating the adaptability of the BPE approach.

Conclusion:

The implementation of the BPE algorithm showcases its capability to compress and tokenize textual data efficiently. This method is particularly advantageous for tasks involving large datasets or morphologically rich languages. By fine-tuning parameters like merge steps, the algorithm can be tailored to diverse datasets and applications.

Part 2: Preprocessing and Analysis of NLTK Movie Reviews Dataset

Objective:

The second part of the assignment focuses on preprocessing and analyzing the NLTK Movie Reviews dataset. The primary objectives include:

1. Cleaning and normalizing textual data for machine learning applications.
2. Evaluating the impact of preprocessing steps on dataset quality.
3. Gaining insights through coverage analysis and preparing the dataset for future tasks.

Dataset Overview:

The NLTK Movie Reviews dataset consists of 2,000 reviews, each labeled as either "positive" or "negative." This labeled dataset is widely used in sentiment analysis research and serves as a benchmark for NLP models.

Preprocessing Steps:

1. Tokenization:
    - Each review is split into individual words using the `word_tokenize` function. Tokenization ensures that the textual data is converted into a format suitable for computational analysis.
2. Stopword Removal:
    - Commonly occurring words like "the," "is," and "and" are filtered out using the NLTK stopword list. Removing these words reduces noise and focuses on more informative tokens.
3. Stemming:
    - Words are reduced to their root forms using the Porter Stemmer. For instance, "running" becomes "run," ensuring that variations of a word are grouped together.

Sample Processing:

- Raw Review: "This movie was absolutely fantastic! The acting was superb."
- Processed Review: ["movi", "absolut", "fantast", "act", "superb"]

- This transformation demonstrates how preprocessing condenses the review while retaining its essential meaning.

Coverage Analysis:

- Coverage measures the percentage of unique words retained after preprocessing.
- Observations:
    - While stopword removal and stemming reduce the overall token count, they preserve sufficient semantic content for downstream tasks like sentiment analysis.

Evaluation and Results:

- Preprocessing significantly improved the dataset's quality by removing noise and normalizing variations in word forms.
- The token reduction ratio was approximately 35%, indicating effective filtering without excessive information loss.

Conclusion:

The preprocessing pipeline successfully prepared the NLTK Movie Reviews dataset for NLP applications. The balance between token reduction and information retention ensures that the dataset remains both concise and informative, providing a solid foundation for tasks like text classification or sentiment analysis.

Overall Summary:

This report presents two foundational NLP tasks:

1. Part 1: Implementation of the Byte Pair Encoding algorithm for subword tokenization. By iteratively merging frequent token pairs, the algorithm effectively optimized vocabulary size while retaining representational power.
2. Part 2: Preprocessing of the NLTK Movie Reviews dataset. The application of tokenization, stopword removal, and stemming enhanced dataset quality, ensuring its readiness for machine learning models.

Both assignments underline the importance of preprocessing and tokenization in NLP workflows. Future directions include:

- Applying advanced tokenization techniques like WordPiece or SentencePiece to the movie reviews dataset.
- Exploring the impact of preprocessing variations on model performance in sentiment analysis.

These assignments highlight critical steps in building efficient and effective NLP pipelines, paving the way for advanced applications in text analysis, classification, and beyond.