**Section 1**

In the task assigned to me, I faced numerous challenges due to the limitations of the data provided. The issues primarily stemmed from inconsistencies in tagging conventions, poorly labeled data, the raw text format of the data, varying file lengths, and the presence of both title and body text in some reviews. These limitations significantly hindered the evaluation process, making the data less reliable and more complex to parse.

Key limitations of the data included:

- \*\*Inconsistent Naming Conventions\*\*: The tags varied significantly, making it difficult to evaluate consistently. For example, tags like “Zoom” and “Zoom mode,” “dvd player” and “player,” and “Universal remote control” and “remote” indicate a lack of standardisation.

- \*\*Poor Labeling\*\*: Some items were incorrectly labeled, such as labeling a “camera” as a “door.” These inaccuracies questioned the reliability of the labels as a source of truth.

- \*\*Raw Text Format\*\*: The data required manual parsing, which is more error-prone compared to structured formats that could be easily read by built-in parsers, like CSV.

- \*\*Complexity Added by Titles\*\*: The inclusion of titles along with the body text in some reviews added an extra layer of complexity to the data parsing process.

- \*\*Varying File Lengths\*\*: Files of different lengths presented a challenge, as I tended to prefer larger files for their more substantial data volume, which I believed would provide a more robust basis for prediction.

To handle these issues, I developed a comprehensive pipeline controlled by my `opinion\_miner\_controller` function. This pipeline is executed in the final two cells of my notebook, where the opinion miner runs and outputs various samples. The process includes several critical functions:

1. \*\*Read File\*\*: A pre-built parser manages data nuances, such as titles, annotations, and special string characters. It separates text data from tags and sentiments, organising it into a structured pandas dataframe.

2. \*\*Pre-Processing Controller\*\*: This function prepares the data for analysis by cleaning text, processing stop words, tokenising, lemmatising, and chunking nouns. Feature normalisation for machine learning occurs later in the pipeline.

3. \*\*Feature Extraction\*\*: Adjusts based on the pre-processing string type, a similarity threshold between product and features, and a choice between two feature extraction models.

4. \*\*Sentiment Analysis\*\*: The `sentiment\_controller` uses parameters like the sentiment classifier and the pre-processing review string type to apply either Vader\_blob classification or senti classification.

5. \*\*Further Processing\*\*: Includes creating feature table dictionaries and mapping dictionaries to align similar words (e.g., 'picture' and 'pic') to their correct tags for evaluation.

6. \*\*Output\*\*: Outputs include a confusion matrix, a metrics table showing precision, accuracy, recall, and the F1 score, and a feature table from the miner.

Additionally, outside the `opinion\_miner\_controller`, I developed functions to optimise the miner’s performance:

- \*\*Average Metrics\*\*: Averages evaluation metrics across three sample files.

- \*\*Sentiment Model Average Comparison\*\*: Compares the performance of two sentiment models.

- \*\*Noun Model Comparison\*\*: Shows differences between two noun extraction methods.

- \*\*Show Optimum String Variables\*\*: Identifies the best pre-processed strings for the miner.

- \*\*Sim Filter\*\*: Finds the optimal similarity parameter for feature extraction.

- \*\*Build ML Classifier and Evaluate ML Classifier\*\*: An alternative approach using a OneVsRestClassifier to both extract and classify features.

Overall, these tools and methodologies allow me to address the initial data limitations effectively, optimising the evaluation and analysis process to achieve reliable results.

**Section 2**

The approach to data parsing and pre-processing in the given system involves several meticulously designed steps to transform raw textual data into structured, analyzable formats. This multi-stage process begins with data extraction and continues through normalization and preparation for downstream analysis, such as sentiment analysis and machine learning.

\*\*1. Data Extraction (`read\_file` function):\*\*

- The initial step involves reading a file and separating reviews based on predefined structures (e.g., markers like asterisks to indicate metadata). This function handles titles by appending them to the reviews if indicated by a marker `[t]`.

- Reviews are split to separate the tags from the content using a delimiter `##`. If no delimiter is found, the tags are noted as empty, which ensures that every piece of text is accounted for without losing data integrity.

\*\*2. Handling Titles (`handle\_titles` function):\*\*

- Titles are integrated with their corresponding reviews when marked by `[t]`. This ensures that any additional context provided by titles is preserved within the main text, enhancing the quality and context of the data processed.

\*\*3. Normalizing Tags (`parse\_and\_normalise\_tags` function):\*\*

- Tags and sentiments associated with each review are extracted and normalized. Sentiments expressed as numeric values are averaged if multiple sentiments are listed for a single tag. This step provides a clean, consistent set of tags and sentiments for each review, facilitating accurate sentiment analysis.

\*\*4. Preprocessing Controller (`pre\_processing\_controller` function):\*\*

- This comprehensive function orchestrates several preprocessing steps:

- \*\*Tokenization:\*\* Splits the review into individual tokens while preserving compound phrases (e.g., 'credit\_card') which are crucial for maintaining the semantic integrity of certain terms.

- \*\*Soft Filtering:\*\* Applies a lighter filtration that includes conversion to lowercase and exclusion of non-alphabetic tokens unless they form part of a compound phrase. This step prepares the data for scenarios where nuanced information is essential.

- \*\*Hard Filtering:\*\* Removes stopwords along with handling case and non-alphabetic characters, focusing the dataset on content-rich words.

- \*\*Lemmatization:\*\* Converts words to their base forms, which helps in reducing the complexity of the dataset and improving the effectiveness of the analytical models.

\*\*5. Preservation of Semantic Relationships:\*\*

- Compound phrases are identified and preserved during tokenization, ensuring that critical relationships between words (e.g., adjectives and nouns they modify) are maintained. This step is crucial for tasks like feature extraction where the precise context of terms impacts the analysis outcome.

\*\*6. Final Structuring:\*\*

- The final output of the preprocessing steps includes multiple versions of the review text, each suited for different analytical purposes. This structured data is then ready to be fed into sentiment analysis modules or machine learning algorithms, providing a flexible foundation for various analytical approaches.

In summary, the approach to data parsing and preprocessing is robust, aiming to preserve as much contextual and semantic information as possible while preparing the data for complex analytical tasks. This methodology ensures that the insights derived from the data are based on a comprehensive and accurate representation of the original text inputs.