**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**