Assignment-4

To clean and preprocess the 'Text' and 'Summary' columns from the dataset, we can follow a systematic approach using various text preprocessing techniques. Here's a step-by-step breakdown of the approach, methodologies, and assumptions for the problem:

Approach:

1. Data Loading: Load the dataset using Pandas library.
2. Data Cleaning: Handle missing values (if any) and ensure data integrity.
3. Text Preprocessing:
    - Remove special characters, punctuation, and symbols.
    - Remove numbers.
    - Remove extra whitespaces.
    - Remove URLs.
    - Handle case sensitivity.
4. Combining Text and Summary:
    - Combine the cleaned 'Text' and 'Summary' columns to create a single column for analysis.
5. Output:
    - Save the cleaned dataset to a new CSV file.

Methodologies:

1. Regular Expressions (Regex):
    - Utilize regex to remove specific patterns such as URLs, special characters, and numbers.
2. String Manipulation:
    - Use string methods to replace or remove certain substrings.
3. Pandas:
    - Leverage Pandas DataFrame operations to apply cleaning functions efficiently across the dataset.
4. Text Concatenation:
    - Combine the cleaned 'Text' and 'Summary' columns using string concatenation with a delimiter for better analysis.
5. Data Validation:
    - Ensure the integrity of the data after preprocessing to avoid loss of information or unintended changes.

Assumptions:

1. Text Encoding:
    - Assuming the text is encoded in UTF-8 format. We'll handle encoding issues accordingly.
2. Data Completeness:
    - Assuming there are no missing values in the 'Text' and 'Summary' columns after dropping NA values.
3. Cleaning Criteria:

- Assuming that removing special characters, punctuation, numbers, URLs, and extra whitespaces is sufficient for text cleaning. Additional cleaning steps may be required based on specific requirements or domain knowledge.

4. Text Combination:
    - Assuming that combining the 'Text' and 'Summary' columns with a delimiter ('TL;DR' in this case) provides meaningful context for analysis. Adjustments can be made based on the analysis goals.
5. Output Format:
    - Assuming the output CSV file will contain the cleaned dataset with the same structure as the original, maintaining data types and column names.

By following this approach, we can effectively clean and preprocess the 'Text' and 'Summary' columns of the dataset, making it suitable for further analysis or modeling tasks.

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	Combined
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a nfection that has been around a few ...	This is a nfection that has been around a few ...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wide...	Great taffy at a great price. There was a wide...

Approach:

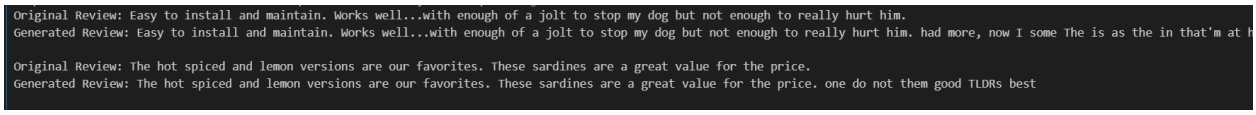
1. Initialization of GPT-2 Model and Tokenizer:
    - Initialize a GPT-2 tokenizer and model from Hugging Face's Transformers library to perform text generation tasks.
2. Data Splitting:
    - Divide the dataset into training and testing sets with a 75:25 ratio to train and evaluate the model's performance.
3. Custom Dataset Preparation:
    - Implement a custom dataset class to prepare the data for training. This class preprocesses the text data and tokenizes it using the GPT-2 tokenizer.
4. Fine-tuning GPT-2 Model:
    - Fine-tune the GPT-2 model on the review dataset to generate summaries. Use the TrainingArguments to configure training parameters such as the number of epochs, batch size, and output directory.
5. Evaluation Metrics:
    - Define a custom evaluation function to compute metrics such as loss, BLEU score, and ROUGE scores to evaluate the performance of the model on the test set.
6. Text Generation:
    - Generate summaries for the test reviews using the fine-tuned GPT-2 model. Pass the test reviews through the model to generate predicted summaries.

Methodologies:

1. Hugging Face Transformers:
    - Utilize the Hugging Face Transformers library to initialize the GPT-2 model and tokenizer, and for fine-tuning the model.
2. PyTorch Dataset:
    - Implement a custom PyTorch Dataset class to preprocess and tokenize the input text data for training.
3. Training Configuration:
    - Configure the training parameters such as the number of epochs, batch size, and optimizer settings using the TrainingArguments class.
4. Model Evaluation:
    - Evaluate the model's performance using standard NLP metrics such as BLEU score and ROUGE scores to assess the quality of the generated summaries.
5. Text Generation:
    - Use the fine-tuned GPT-2 model to generate summaries for the test reviews by passing the input text through the model and decoding the generated output.

Assumptions:

1. Model Selection:
    - GPT-2 model from Hugging Face is chosen as the model for text generation tasks due to its effectiveness in generating coherent text.
2. Data Splitting:
    - The dataset is divided into training and testing sets with a 75:25 split ratio to ensure sufficient data for training and evaluation.
3. Tokenization and Padding:
    - The input text is tokenized using the GPT-2 tokenizer and padded to a maximum length of 100 tokens to feed into the model.
4. Fine-tuning Parameters:
    - Assumed default values for fine-tuning parameters such as learning rate, batch size, and number of epochs unless explicitly specified.
5. Evaluation Metrics:
    - BLEU score and ROUGE scores are used as evaluation metrics to assess the quality of the generated summaries, assuming they provide meaningful insights into the model's performance.



Approach:

1. ROUGE Score Calculation:
    - Compute ROUGE scores on the test set to evaluate the model's performance in generating summaries. ROUGE (Recall-Oriented Understudy for Gisting

Evaluation) is a set of metrics used for evaluating the quality of generated summaries compared to reference summaries.

2. Comparison with Actual Summaries:
    - Compare each generated summary with its corresponding actual summary from the test set to assess the model's ability to generate meaningful summaries.

Methodologies:

1. ROUGE Scorer:
    - Utilize the rouge_score library to compute ROUGE scores for each predicted summary compared to its corresponding actual summary. This library provides metrics such as ROUGE-1, ROUGE-2, and ROUGE-L.
2. Tokenization:
    - Tokenize the test reviews using the GPT-2 tokenizer to generate summaries for evaluation.
3. Model Inference:
    - Pass the tokenized test reviews through the fine-tuned GPT-2 model to generate predicted summaries.
4. Comparison:
    - Compare each generated summary with its actual summary from the test set using the ROUGE scorer to calculate ROUGE scores.

Assumptions:

1. ROUGE Metrics:
    - Utilize ROUGE-1, ROUGE-2, and ROUGE-L metrics to evaluate the model's performance. These metrics measure the overlap of unigrams, bigrams, and the longest common subsequence between the generated summary and the actual summary.
2. Model Inference:
    - Assume that the fine-tuned GPT-2 model generates coherent summaries based on the input reviews.
3. Test Set Integrity:
    - Assume that the test set contains a representative sample of reviews with corresponding summaries for evaluation.
4. ROUGE Score Interpretation:
    - Consider higher ROUGE scores as indicative of better performance, indicating a higher degree of similarity between the generated summaries and the actual summaries.