Code Report: Fine-tuning GPT-2 Model for Text Generation

1. Data Preprocessing - The code begins by importing necessary libraries, including pandas, numpy, re, and transformers.
- It reads a CSV file containing review data and preprocesses it by sampling 1200 random rows, removing duplicates, and handling missing values.
- Text cleaning is performed using regular expressions and NLTK libraries to convert text to lowercase, remove URLs, non-alphabetic characters, extra whitespaces, and stopwords.

2. Model Initialization and Fine-tuning
- The GPT-2 tokenizer and model are initialized using the `GPT2Tokenizer` and `GPT2LMHeadModel` classes from the transformers library.
- The model is moved to the available CUDA device if CUDA is available, otherwise, it is moved to CPU.
- Training arguments are defined, specifying output directory, number of training epochs, batch size, and logging settings.
- The `Trainer` class from the transformers library is used to fine-tune the model on the training dataset. Training progress is logged, and checkpoints are saved at regular intervals.

3. Model Saving
- After training, the fine-tuned model is saved to the specified directory using the `save_pretrained` method.

4. Model Loading and Evaluation
- The saved model is loaded from the checkpoint directory.
- Test data is assumed to be in a pandas DataFrame format and is preprocessed similarly to the training data.
- Text generation is performed for each input in the test dataset using the fine-tuned GPT-2 model. The generated text is decoded and printed.

5. Code Enhancement:
- The code is modularized by encapsulating repetitive tasks into functions, such as data preprocessing, model initialization, and text generation.
- Default parameter values are used where appropriate, providing flexibility and readability.
- Unnecessary context managers (`model.eval()` and `torch.no_grad()`) are removed from the text generation loop for better code organization.

6. Overall Summary
- The code efficiently fine-tunes a GPT-2 model for text generation tasks using a dataset of reviews.
- It demonstrates good practices in data preprocessing, model training, and evaluation.
- Modularization and code enhancements improve readability, maintainability, and reusability.