README.md

Overview

This repository contains code for training a GPT-2 model for text generation tasks, specifically for summarization. The model is trained on a dataset of reviews, where it learns to generate summaries based on the input text.

Dependencies

- torch: PyTorch library for deep learning
- pandas: For data manipulation and analysis
- transformers: Library for natural language processing tasks, used here for GPT-2 model and tokenizer
- sklearn: Library for machine learning tasks, used for train-test splitting
- rouge: For evaluating the quality of generated summaries

Code Explanation

1. Importing Libraries

- Necessary libraries such as torch, pandas, transformers, sklearn, and rouge are imported.
- The required classes and functions from these libraries are imported to facilitate various tasks.

2. Data Preprocessing

- The provided dataset (`Reviews.csv`) is loaded into a pandas DataFrame.
- Null values are removed from the DataFrame.
- A new column named `training` is created by concatenating the lowercased `Text` and `Summary` columns along with a separator `"TL;DR"`. This column represents the input data for training the model.
- Only the first 5000 rows of the processed DataFrame are selected for training.
- Tokenizer and model are initialized using GPT-2's pre-trained weights.

3. Training Data Splitting

- The dataset is split into training and testing sets using `train_test_split` function from sklearn.

4. Dataset Encoding

- Custom Dataset class `EncodedDataset` is defined to encode the text data using the tokenizer.
- Padding and truncating are performed to make all sequences of equal length.

- The encoded sequences are converted into PyTorch tensors and stored as dataset.

5. Model Training

- A function `train_save_modal` is defined to train the model using the encoded dataset.
- The model is trained for a specified number of epochs using Adam optimizer.
- Model checkpoints are saved after training.

6. Text Generation

- Functions `select_top_tokens` and `generate_text` are defined to generate summaries using the trained model.
- Top-k sampling technique is employed for generating tokens.
- ROUGE scores are calculated to evaluate the quality of generated summaries.

7. Hyperparameter Tuning

- A function `train_with_evaluation` is defined to perform training with hyperparameter tuning.
- The function trains the model with different combinations of learning rates, batch sizes, and epochs.
- Validation loss is calculated to determine the best hyperparameters.

8. Hyperparameter Search

- A grid search approach is used to find the best hyperparameters based on validation loss.
- Different combinations of learning rates, batch sizes, and epochs are tested.
- The hyperparameters yielding the lowest validation loss are selected as the best.

Why

- **Data Preprocessing**: Data is preprocessed to remove null values and create suitable input data for the model.
- **Model Training**: The GPT-2 model is trained on the processed dataset to learn summarization.
- **Text Generation**: Functions are implemented to generate summaries based on the trained model.
- **Hyperparameter Tuning**: Hyperparameters such as learning rate, batch size, and epochs are tuned to optimize model performance.

Conclusion

This project demonstrates the process of training a GPT-2 model for text summarization tasks. By fine-tuning the model on review data, it can generate concise summaries, which can be beneficial for various applications such as text summarization, content generation, and more. The hyperparameter tuning process helps in improving the model's performance by finding the optimal configuration.