AI-Powered Code Optimizer Report

1. Project Overview

The AI-Powered Code Optimizer is an innovative tool designed to automatically analyze and optimize Python code using Large Language Models (LLMs). The project leverages a fine-tuned **Mistral-7B** model to provide developers with suggestions for enhancing code performance and clarity. It aims to streamline the code optimization process by automating the identification and improvement of inefficient or unclear code segments.

2. Key Features

The core functionalities and distinguishing aspects of the Code Optimizer include:

- **LLM-Powered Optimization**: Utilizes a fine-tuned Mistral-7B model to generate optimized Python code, focusing on both performance and clarity.
- **Efficient Fine-Tuning**: Employs **PEFT (LoRA)** for adapting the base model, ensuring minimal computational overhead during the fine-tuning process.
- **Quantized Inference**: Incorporates **4-bit quantization** (bitsandbytes) to achieve fast and memory-efficient model inference.
- **Interactive UI**: Provides a user-friendly **Streamlit** interface for real-time code optimization, allowing developers to input code and receive optimized suggestions instantly.
- **Modular Design**: Features a clean and professional directory structure, promoting easy extension and maintenance of the codebase.

3. Project Structure

The repository `Code-Optimizer` is organized into several key directories, each serving a specific purpose:

Directory	Description
`assets/`	Contains media assets and project presentations.
`data/`	Stores training datasets, such as `code_optimization_dataset.csv`.
`docs/`	Intended for detailed documentation and guides.
`notebooks/`	Houses Jupyter notebooks for experimentation, including `analysis.ipynb`, `application.ipynb`, and `fine-tuning.ipynb`.
`src/`	Contains the main source code of the application.
`src/app/`	Holds the Streamlit application logic, specifically `application.py`.
`src/training/`	Includes scripts for model fine-tuning, such as `fine-tuning.py`.
`tests/`	Dedicated to unit tests for the project.
`LICENSE`	Specifies the project's licensing (MIT License).
`README.md`	Provides an overview and getting started guide for the project.
`requirements.txt`	Lists all project dependencies.

4. Getting Started

Prerequisites

To run the Code Optimizer, the following are required:

- Python 3.8+
- CUDA-enabled GPU (recommended for training and inference)

Installation

1. **Clone the repository**:

```
git clone https://github.com/SeifEldenOsama/Code-Optimizer.git
cd Code-Optimizer
```

2. **Install dependencies**:

```
pip install -r requirements.txt
```

Usage

Running the Web Application

To start the Streamlit interface:

```
streamlit run src/app/application.py
```

Fine-Tuning the Model

To initiate the fine-tuning process:

```
python src/training/fine-tuning.py
```

5. Methodology

The project follows a structured AI development lifecycle for model training and deployment:

1. **Data Preparation**: Involves cleaning and formatting code pairs to be used for supervised fine-tuning of the LLM.
2. **Model Adaptation**: Applies Low-Rank Adaptation (LoRA) to the Mistral-7B base model, allowing for efficient adaptation with domain-specific data.

3. **Quantization**: Compresses the adapted model to 4-bit using `bitsandbytes` for efficient deployment and reduced memory footprint.
4. **Evaluation**: The fine-tuned model is tested on unseen code snippets to ensure the quality and effectiveness of the generated optimizations.

6. License

The AI-Powered Code Optimizer project is licensed under the MIT License. Details can be found in the `LICENSE` file within the repository.

7. Conclusion

This project presents a robust solution for Python code optimization, leveraging the power of fine-tuned LLMs. Its modular architecture, efficient training, and interactive interface make it a valuable tool for developers seeking to improve their code quality and performance. The use of modern techniques like PEFT (LoRA) and 4-bit quantization demonstrates a commitment to efficient and scalable AI application development.

Developed by Manus AI