Project Report: AI Chatbot for Mental Health Support

A Guided AI/ML Project
July 29, 2025

Abstract

This report details the development of a conversational AI agent designed to provide empathetic emotional support. The project encompasses the entire machine learning lifecycle, from environment setup and model selection to fine-tuning, backend API development, and frontend user interface creation. A pre-trained language model, `microsoft/DialoGPT-small`, was successfully fine-tuned on the `empathetic_dialogues` dataset to adapt its conversational style. The project addresses real-world development challenges, including model instability and hardware constraints, culminating in a locally deployed application powered by a custom AI model.

1 Introduction

The primary goal of this project was to develop a conversational AI designed to provide basic, empathetic emotional support. The project followed a structured, hands-on learning path, beginning with foundational concepts and culminating in a deployed application. The core technical achievement was to fine-tune a pre-trained language model on a specialized dataset to adapt its conversational style to be more supportive and understanding. This report outlines the tools, steps, and key learnings from this process.

2 Tools Used

A combination of industry-standard tools and libraries was utilized to build this project:

- **Programming Language:** Python
- **AI/ML Libraries:** Hugging Face `transformers`, `datasets`, PyTorch
- **Backend API:** Flask
- **Frontend UI:** Streamlit
- **Development Environments:**
  - **Anaconda:** For local dependency and environment management.
  - **Kaggle Notebooks:** For GPU-accelerated model fine-tuning.
  - **Visual Studio Code:** For local application development and testing.

3 Steps Involved in Building the Project

The project was executed in a series of logical phases, from initial setup to a functional application.

1. **Environment and Version Control:** The project began by establishing a clean and isolated development workspace using an Anaconda virtual environment. A Git repository was initialized and linked to GitHub to ensure all code changes were tracked and managed professionally.

2. **Model Fine-Tuning:** This critical phase was conducted on Kaggle to leverage free GPU resources. After exploring several alternatives, the `microsoft/DialoGPT-small` model was selected. It was fine-tuned on the `empathetic_dialogues` dataset to teach it a more supportive conversational style. Key challenges, such as model collapse (producing gibberish) and hardware memory limits, were overcome by adjusting the learning rate and optimizing the training process to save only the best-performing model checkpoint.

3. **Backend Development:** A backend API service was created using Flask in an `app.py` script. This server loads the fine-tuned model and exposes a `/chat` endpoint to receive user messages and return the AI's response. A logging feature was also implemented to save conversation history to a file for later analysis.

4. **Frontend Development:** A user-friendly interface was built using Streamlit in a `frontend.py` script. This UI provides a simple chat window, manages the conversation history, and communicates with the Flask backend to create an interactive experience.

5. **Deployment and Testing:** Initial deployment attempts were made on cloud platforms like Replit and Render. However, the memory and disk space limitations of the free tiers were insufficient to host the large PyTorch and Transformers libraries, resulting in build failures. The project successfully pivoted to a local deployment model, running the Flask backend and Streamlit frontend concurrently on a local machine for end-to-end testing.

4 Conclusion

This project successfully achieved its objective of creating an empathetic AI chatbot. Through a structured, hands-on process, a base language model was successfully fine-tuned to exhibit a more supportive personality. Key real-world challenges, such as model instability and hardware limitations on cloud platforms, were encountered and systematically resolved, providing valuable engineering experience. The final result is a fully functional, locally hosted application that integrates a custom AI model with a modern web interface, fulfilling all initial project deliverables.