Final Project: Building a daily life assistant with an Instruction-Fine-Tuned GPT-2

> Author : Badr TAJINI - Large Language model (LLMs) - ESIEE 2024-2025

Project Summary

This final project focuses on creating a **basic AI assistant** capable of following everyday instructions, built on a **GPT-2** model that is **instruction-fine-tuned**. The goal is to practice **transforming a pretrained LLM** (like GPT-2) into a system that can respond effectively to various daily queries and tasks (e.g., scheduling reminders, giving recipe suggestions, generating short emails).

You will leverage their understanding of **pretraining**, **fine-tuning**, and, most importantly, **instruction-based fine-tuning** developed across the course (Chapters 5, 6, and 7). The project also draws on the methodology seen in Labs 6-7, where a dataset was prepared for supervised instruction fine-tuning using an **Alpaca-style** prompt format. By the end of this project, you will have a working prototype of a GPT-2–based assistant, demonstrating how **instruction fine-tuning** can dramatically improve the model's ability to follow user commands and prompts.

Learning Objectives

1. **Model Architecture & Pretraining**
   - Understand GPT-2's architecture and how it was originally pretrained to predict text sequentially.
   - Gain hands-on experience loading pretrained weights and preparing them for further training.
2. **Instruction Fine-Tuning**
   - Learn the process of **supervised instruction fine-tuning**, where the model is trained on <instruction, response> pairs.
   - Explore prompt engineering (e.g., **Alpaca-style** formatting) to maximize the model's instruction-following capabilities.
3. **Evaluation & Refinement**
   - Evaluate the fine-tuned assistant's responses for correctness, clarity, and coherence.
   - Adjust hyperparameters and training strategies based on metrics (e.g., validation loss, user satisfaction) and perform iterative improvements.
4. **Practical Application**
   - Design realistic use cases and daily tasks (e.g., to-do lists, note-taking, Q&A) that showcase the assistant's utility.
   - Integrate the final model into a simple demo application (**command-line interface** or **basic web UI**).

5. **Project Documentation & Presentation**
    - Produce a clear and concise technical report.
    - Optionally record a short demo video illustrating the assistant's performance in different scenarios.

Project Requirements

1. **Choice of Domain**: You can pick **any real-life context** to demonstrate the assistant (e.g., personal productivity, cooking tips, educational Q&A).
2. **Implementation Environment**:
    - Local training with GPU acceleration (e.g., PyTorch on a desktop or laptop with CUDA support).
    - Or cloud-based notebook environments (e.g., Google Colab, AWS EC2, Azure ML) if resources are available.
3. **Collaboration**: Pairs or small teams (**up to a maximum of 5, but I absolutely require traceability of who did what in the project**) are encouraged to promote idea exchange (solo projects remain acceptable).
4. **Deliverables**: A **technical report** describing the entire pipeline, a **demo video** (optional but recommended), the **training scripts**, and any relevant **dataset formatting scripts** in a private repository on Github.

Project Phases

Phase 1: Environment Setup & Base Model Initialization

- **Objective**: Prepare the development environment and confirm access to GPT-2's pretrained weights.
- **Tasks**:
    1. Install necessary libraries (e.g., Transformers, PyTorch, tokenizers).
    2. Load GPT-2 (small or medium variant) to verify baseline text generation capabilities.
    3. Set up any GPU/TPU environment if available (Colab, local GPU, or cloud instance).

Phase 2: Instruction Dataset Preparation

- **Objective**: Format and refine the **instruction–response** dataset for supervised fine-tuning.
- **Tasks**:
    1. Organize the dataset into **<instruction, response>** pairs using an **Alpaca-style** prompt format or similar.
    2. Ensure data cleanliness and consistency (spell-check, removal of duplicates, or mislabeled entries).
    3. Split the dataset into training, validation, and (optionally) test sets to measure progress.

Phase 3: Instruction Fine-Tuning

- **Objective**: Train GPT-2 on the curated instruction–response dataset.
- **Tasks**:
    1. Configure hyperparameters (learning rate, batch size, number of epochs) and run fine-tuning.

2. Monitor training and validation loss curves to detect overfitting or underfitting.
3. Periodically generate sample responses to subjectively assess improvements in instruction-following.

Phase 4: Evaluation & Iterative Improvement

- **Objective**: Evaluate the fine-tuned model and refine it based on performance metrics.
- **Tasks**:
    1. Use automatic metrics (e.g., perplexity on validation set) as a baseline.
    2. Conduct a **human evaluation**: manually test the assistant on different daily tasks to gauge quality.
    3. Adjust hyperparameters or data augmentation techniques if needed, and retrain or fine-tune further.

Phase 5: Deployment & Final Presentation

- **Objective**: Provide a polished demonstration and present the final assistant.
- **Tasks**:
    1. Wrap the fine-tuned model in a basic interface (**CLI script or minimal web app**).
    2. Prepare a short video showing the assistant’s responses to various instructions.
    3. Write a comprehensive report including your methodology, key findings, challenges faced, and future enhancements.

Evaluation Criteria

1. **Technical Depth**
    - Correct loading and fine-tuning of GPT-2.
    - Quality of prompt engineering and dataset curation.
2. **Model Performance**
    - Consistency, clarity, and usefulness of generated responses.
    - Improvement from the pretrained baseline to the instruction-fine-tuned model.
3. **Documentation**
    - Clarity of the final report (architecture, data preparation, training details, evaluation).
    - Completeness of the code repository (scripts, notebooks, environment files).
4. **Presentation**
    - Clarity and organization of the demonstration (recorded).
    - Ability to handle various daily-life instructions effectively and consistently.

Deliverables (in a private repository on Github)

1. **Technical Report**: A detailed document (**ReadMe**) explaining data preprocessing, training procedures, and results.
2. **Video Demonstration**: Showcasing the assistant’s capabilities across several user instructions.
3. **Training Code & Scripts**: Containing all relevant Python scripts or notebooks for fine-tuning, including environment setup (requirements.txt or conda environment.yml).

4. **Dataset and Prompt Templates**: A folder or reference to how you formatted your <instruction, response> pairs.

Deadlines

- **Final Submission for project (on Github)**: *(02/02/2025 at 23:59 Paris Time)*
  - Submit the final code, report, and any demonstration materials.
- **Final Submission for Labs (on Github)**: *(02/02/2025 at 23:59 Paris Time)*
  - Submit the final code, report, and any demonstration materials.

Weighting of the assessment

- **Project**: 50%
- **Labs**: 50%
- **Participation**: 10%

Identification

P.S.: Don't forget to mention: your name - degree course (filière) - academic year - school (for credits on your Github).

Delivery (traceability)

To ensure that your work can be traced (**for project and Labs**), please enter your full name and that of your teammates on the following Google form: Link.

Appendix

> Quick note : This project offers a **hands-on introduction** to **instruction fine-tuning** of LLMs. By transforming a GPT-2 model into a daily-life assistant, you will gain practical experience in **prompt engineering**, **finetuning strategies**, **evaluation** of model outputs, and the skill of adapting large pretrained models to **human-centered tasks**.

References :

- Awesome LLMs Fine-Tuning
- Fine-tuning large language models (LLMs) in 2024
- How to Fine-Tune LLMs in 2024 with Hugging Face
- Stanford Alpaca
- FLAN Dataset
- FLAN v2 Dataset