Project 23: Fine-Tuning LLMs for Recipe Recommendation and Instruction Following By Mehrdad

Goal:
The aim of this project is to **fine-tune a Large Language Model (LLM)** on recipe data to improve its ability to follow food-related instructions. You will generate instruction-response training data from the recipe dataset, fine-tune an LLM, and compare its performance with a non-fine-tuned baseline. A key part of the project is **detecting and analyzing hallucinations** (when the model generates incorrect or non-existent information).

Consider HUMMUS dataset

Specifications

1. Dataset Exploration

- Summarize and visualize key nutritional features: calories, protein, and sodium.
- Explore the distribution of duration to compare quick vs. long recipes.
- Identify the most common health_category values.
- List the top tags and frequent ingredients.
- Plot a simple correlation (e.g., protein vs. calories).

2. Instruction Dataset Generation

- Convert recipe data into **instruction-response pairs** that simulate user queries.
- Example formats:
  - **Instruction:** *"Suggest a healthy pasta recipe under 400 calories."* **Response:** *"Vegetarian Tomato Basil Pasta - 350 calories, 15g protein, ready in 20 minutes."*
  - **Instruction:** *"List three low-sodium chicken dishes."* **Response:** *"1. Lemon Garlic Chicken (180mg sodium)... 2. Grilled Herb Chicken (200mg sodium)... etc."*
- Include a variety of queries: ingredient filters, nutrition limits, preparation time, health category, or combinations.

3. Data Formatting for Fine-Tuning

- Structure the dataset in JSONL with fields:
- {
- "instruction": "Suggest a vegan dessert under 300 calories",
- "input": "",
- "output": "Fruit Salad with Citrus Dressing - 250 calories, 5g fiber, 15 min preparation."
- }
- Split into **training (80%)**, **validation (10%)**, and **test (10%)** sets.

4. Baseline (Non-Fine-Tuned LLM)

- Use a pre-trained model (e.g., GPT-2, LLaMA) without fine-tuning.
- Prompt it with test queries and record outputs.
- Example: Query *"Give me a high-protein vegetarian recipe under 20 minutes"*.
  - Baseline might return vague or incorrect answers.

5. Fine-Tuning the LLM

- Fine-tune a small/medium LLM (e.g., GPT-2, Mistral, LLaMA-2 with LoRA).

- Train on the generated instruction-response dataset.
- Track training/validation loss and document hyperparameters.

6. Evaluation Framework

- Use test queries to compare fine-tuned vs. baseline models.
- Evaluate outputs on:
  - **Relevance:** Does the answer match the query constraints?
  - **Factual Accuracy:** Do calories, protein, or nutrition values match dataset entries?
  - **Fluency:** Is the answer coherent and natural?
- Metrics: BLEU/ROUGE (text similarity), Precision/Recall (constraint satisfaction), human evaluation for fluency.

7. Hallucination Detection

- Define hallucinations as:
  - **Factual:** Model invents values not in dataset (e.g., wrong calorie counts).
  - **Recipe:** Model suggests a recipe not present in dataset.
- Automatically check outputs against dataset fields.
- Calculate hallucination rate = % of outputs with incorrect/made-up facts.

8. Hallucination Mitigation

- Strategies to reduce hallucinations:
  - Ground outputs in retrieved recipe data before generation.
  - Post-check outputs against structured fields and correct mismatches.
  - Add system-level constraints in prompts (e.g., *"Only use facts from dataset."*).

9. Error Analysis

- Compare error cases between baseline and fine-tuned models.
- Example:
  - Baseline → *"Vegan Chicken Curry under 200 calories"* (hallucination).
  - Fine-tuned → *"Tofu Stir-Fry - 190 calories, vegan, 18 min prep."* (valid).
- Identify frequent errors such as unrealistic calorie estimates or invented recipes.

10. Comparison Study

- Summarize differences between **baseline** and **fine-tuned**:
  - Baseline: more flexible, but prone to hallucinations.
  - Fine-tuned: more accurate, dataset-grounded, better at constraint-following.
- Provide tables/graphs of evaluation metrics.

12. Extensions (Optional)

- Try **parameter-efficient fine-tuning** (LoRA, adapters) to save resources.
- Explore **data augmentation** (paraphrased queries, synthetic instructions).
- Compare **RAG + fine-tuning** vs. **fine-tuning only** for hallucination reduction.