

KaSangkap-Hunt (Filipino Chefbot)

Laguna State Polytechnic University – Santa Cruz campus

CMSC – 308 Programming Languages

Project Documentation

Submitted by: Salvahan, Umali, Hornilla, Luzande, Lachano

Course: BSCS3B - Final Year Project

Submitted to: Ms. Roxanne Garbo

Date: December 2, 2025

Table of Contents

- 1. Introduction**
 - 1.1. Project Overview
 - 1.2. Objectives
 - 1.3. Technology Stack
- 2. System Architecture**
 - 2.1. High-Level Overview
 - 2.2. Core Components
- 3. Methodology**
 - 3.1. Data Collection and Preprocessing
 - 3.2. Model Selection and Fine-Tuning
 - 3.3. Iterative Improvement
 - 3.4. Deployment
- 4. User Guide**
 - 4.1. System Requirements
 - 4.2. Step-by-Step Instructions to Run the Application
- 5. Results and Evaluation**
 - 5.1. Performance on Core Tasks
 - 5.2. Performance on Advanced Tasks
 - 5.3. Limitations
- 6. Conclusion and Future Work**
 - 6.1. Project Summary
 - 6.2. Potential Enhancements

1. Introduction

1.1. Project Overview

KaSangkap-Hunt is a conversational AI, or "Chefbot," specifically designed to be an expert in Filipino cuisine. It leverages a fine-tuned Large Language Model (LLM) to generate recipes, answer cooking-related questions, suggest ingredient substitutions, and adapt to modern cooking techniques. The project aims to create an interactive and intelligent tool for individuals interested in exploring the rich culinary heritage of the Philippines.

1.2. Objectives

- To fine-tune a pre-trained Large Language Model on a specialized dataset of Filipino recipes.
- To develop a model capable of generating coherent, accurate, and contextually relevant recipes.
- To build an interactive and user-friendly web interface for real-time interaction with the model.

- To implement robust training and deployment workflows that are reproducible and resilient to disconnections.

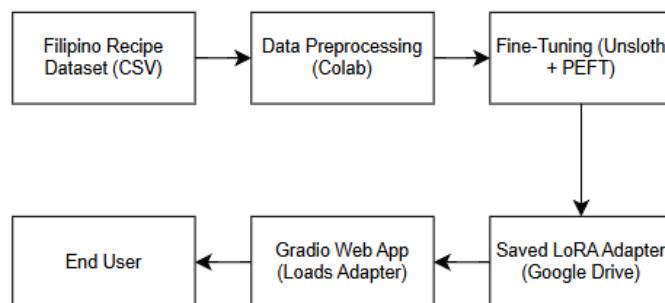
1.3. Technology Stack

- **Programming Language:** Python
- **Environment:** Google Colab (with T4 GPU)
- **Core AI/ML Libraries:**
 - **PyTorch:** The foundational deep learning framework.
 - **Transformers (Hugging Face):** For accessing pre-trained models and the training pipeline.
 - **Unsloth:** An optimization library for faster and more memory-efficient fine-tuning of LLMs.
 - **PEFT (Parameter-Efficient Fine-Tuning):** For utilizing the LoRA (Low-Rank Adaptation) technique.
 - **bitsandbytes:** For 4-bit model quantization, enabling the model to run on consumer-grade GPUs.
- **Web UI and Deployment:**
 - **Gradio:** A Python library for quickly building and sharing machine learning web applications.
 - **Google Drive:** For persistent storage of datasets, model checkpoints, and final trained models.

2. System Architecture

2.1. High-Level Overview

The system follows a standard fine-tuning and deployment pipeline. A pre-trained base model is adapted using a custom dataset. The resulting fine-tuned model is then loaded into memory and exposed to users through a web interface.



Flowchart

2.2. Core Components

1. **Dataset:** A CSV file (`filipino_recipes.csv`) containing over 2,000 Filipino recipes, with columns for recipe name, ingredients, and instructions.
2. **Base Model:** The `unsloth/mistral-7b-instruct-v0.3-bnb-4bit` model was used as the foundation. This is a 7-billion parameter instruction-tuned model, pre-quantized for memory efficiency.
3. **Fine-Tuned Model (LoRA Adapter):** The training process does not modify the base model. Instead, it creates a small, separate "adapter" (`recipe-chef-lora-filipino_v2`) that contains the specialized knowledge of Filipino cuisine. At runtime, this adapter is merged with the base model.
4. **User Interface:** A web app built with Gradio provides a real-time, interactive chat experience.

3. Methodology

3.1. Data Collection and Preprocessing

The primary dataset, `filipino_recipes.csv`, was sourced and cleaned. The ingredients and instructions columns, which used pipe (|) separators, were parsed into structured lists. To enhance the model's capabilities, the dataset was augmented with synthetically generated question-and-answer pairs, including:

- Requests for full recipes.
- Questions about cooking duration.
- Inquiries about ingredient substitutions, complete with realistic examples.

```

# --- SUBSTITUTION LOGIC ---
if ingredients:
    substitutions = {
        'pork': 'chicken for a lighter version, beef for a richer flavor, or oxtail which is traditional for Kare-Kare',
        'beef': 'pork, chicken, or oxtail sections for a different flavor profile',
        'chicken': 'pork for a richer taste, or firm tofu for a vegetarian option',
        'shrimp': 'fish fillet, squid rings, or small pieces of chicken',
        'fish sauce': 'soy sauce with a squeeze of calamansi, or simply salt to taste',
        'bagoong': 'fish sauce (patis) for a salty flavor, though the unique fermented taste will be different',
        'vinegar': 'calamansi juice or lemon juice for a citrus-based acidity'
    }
    ingredient_to_replace = random.choice(ingredients)
    found_sub = False
    for key, value in substitutions.items():
        if key in ingredient_to_replace.lower():
            training_data.append({
                "instruction": f"What can I use instead of {ingredient_to_replace} in {title}?",
                "input": "",
                "output": f"In {title}, a good substitute for {ingredient_to_replace} is {value}. The key is to match the texture and cooking time for the best results."
            })
            found_sub = True
            break
    if not found_sub:
        training_data.append([
            "instruction": f"Can I substitute {ingredient_to_replace} in {title}?",
            "input": "",
            "output": f"For {title}, you can often replace {ingredient_to_replace} with another ingredient that has a similar flavor and texture. It's always fun to experiment in the kitchen!"
        ])
# --- END OF SUBSTITUTION LOGIC ---

```

Code Snippet Section 2 Google Colab

3.2. Model Selection and Fine-Tuning

- **Model:** Mistral-7B was chosen for its strong performance and manageable size. The Unslot library was used to enable 2x faster training and 60% less memory usage, making it feasible to train on a free Google Colab T4 GPU.
- **Technique:** Parameter-Efficient Fine-Tuning (PEFT) with LoRA was employed. This approach freezes the weights of the large pre-trained model and trains only a small number of new, adaptable weights. This drastically reduces computational cost and prevents "catastrophic forgetting."

3.3. Iterative Improvement

The model was trained and evaluated in multiple cycles.

1. **Initial Training (1200 steps):** Revealed a weakness where the model would output placeholders (e.g., [alternative]) for substitutions.
2. **Data Enhancement:** The data preprocessing script was updated to provide real, high-quality substitution examples.
3. **Retraining (2400+ steps):** The model was trained for additional steps on the improved dataset. This fixed the substitution issue and resolved "looping" behavior on complex recipes like Pinakbet.

Write me a recipe for pinakbet

- 1 tablespoon canola oil
- 3 cloves Garlic (peeled, chopped)
- 1 medium onion (sliced)
- 500 Grams pork belly (cut into cubes)
- 2 cups water
- 4 pieces eggplant (quartered)
- 1/2 bundle long beans (sitaw) (cut into 2 inch lengths)
- 1/2 head cabbage (shredded)
- 1 bunch okra (halved)
- 1 Cup ampalaya (bitter gourd) (sliced)
- 1/8 teaspoon ground black pepper (or to taste)
- Salt (to taste)

Instructions:

1. In a large pot over medium heat, heat oil. Sauté garlic and onions until translucent. Add pork; cook until browned. Pour in water and bring to a boil. Lower heat, cover, and simmer for about 30 minutes or until meat is tender.
2. Add vegetables one by one in the order listed above, pushing them down into the broth as you add them. Simmer until just cooked through. Season with salt and pepper. Serve while hot.

Broken Pinakbet output

Chatbot

Write a recipe for Pinakbet

Ingredients:

- 1 tablespoon canola oil
- 2 cloves garlic, peeled and minced
- 1 onion, peeled and sliced thinly
- 1 pound pork belly, cut into cubes
- 1/4 cup shrimp paste
- 3 Roma tomatoes, chopped
- 1 bunch okra, ends trimmed and cut into 1-inch pieces
- 2 Chinese eggplants, ends trimmed and quartered
- 1 large bitter melon, ends trimmed and sliced into half moons
- 8 Thai chili peppers, stemmed and minced
- 1 cup water
- salt and pepper to taste

Instructions:

1. In a wide pan over medium heat, heat oil. Add garlic and onions and cook until softened.
2. Add pork and cook, stirring occasionally, until lightly browned.
3. Add shrimp paste and continue to cook, stirring regularly, for about 1 to 2 minutes or until the meat is evenly coated with the paste.
4. Add tomatoes and cook, mashing with back of spoon, until softened and release juice.
5. Add okra, eggplant, and bitter melon. Stir gently to combine.
6. Add chili peppers and water. Bring to a boil, skimming any scum that floats on top.
7. Lower heat, cover, and simmer for about 10 to 15 minutes or until vegetables are tender and liquid is reduced.
8. Season with salt and pepper to taste. Serve hot.

Fixed Pinakbet Output

3.4. Deployment

The final, trained model is deployed using a Gradio ChatInterface, which provides a user-friendly, ChatGPT-like UI. The application is hosted within the Google Colab environment, and ngrok is used to create a temporary, public URL for demonstration and sharing purposes.

4. User Guide

4.1. System Requirements

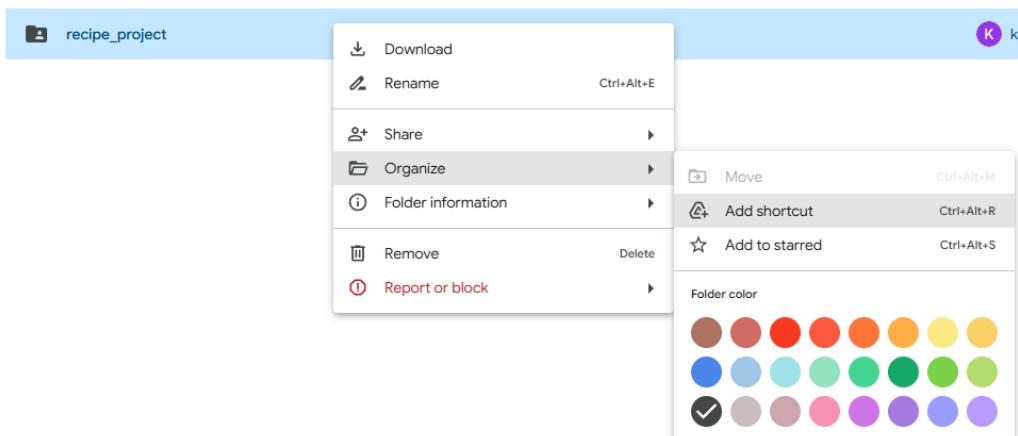
- A Google Account.
- A modern web browser (e.g., Chrome, Opera, Edge).
- An internet connection.

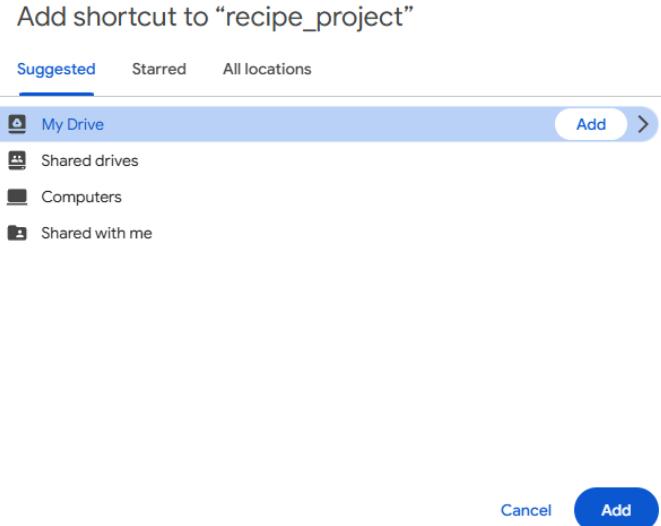
4.2. Step-by-Step Instructions to Run the Application

Goal: To launch the "KaSangkap-Hunt" web application. The entire process runs in the cloud on Google Colab, and no local software installation is required.

Step 1: Access the Project Files and Open the Notebook

1. Check the "Shared with Me" section in your Google Drive, you will see the recipe_project folder name there. (The author shared it with you)
2. **IMPORTANT:** Right-click on the recipe_project folder name at the top of the screen and select "Add shortcut to Drive". Select "My Drive" then Click "Add" in the pop-up. This step is essential for Colab to find the model files.





3. Now, open the Colab notebook directly using this link:

[Open the Chefbot Notebook in Colab](#)

Note: A pop-up may warn that the notebook was not authored by Google. Please click "Run anyway" to proceed.

Step 2: Configure the Colab Environment

1. Connect to a GPU Runtime: Once the notebook is open, go to the menu and select "**Runtime**" -> "**Change runtime type**".
2. In the "**Hardware accelerator**" dropdown, choose "**T4 GPU**" and click "**Save**". The notebook will automatically connect to the required hardware.

Step 3: Prepare the Environment and Launch the Application

This multi-step process will install the required software, connect to the project files, load the trained AI model into memory, and finally launch the web interface. **Please run the following cells in order.**

1. Install Dependencies (Section 1):

- Find the cell under **Section 1: REQUIRED DEPENDENCIES**.
- Click the "**Play**" button to run it. This will install all necessary libraries.
- **Note:** After installation, a pop-up may appear asking to "Restart session". If it does, click the "Restart session" button, and then proceed to the next step.

2. Mount Google Drive (Section 2):

- Find the cell under **Section 2: SETUP AND DATA PREPARATION**.

- Run this cell. It will prompt you to authorize access to your Google Drive. Follow the on-screen instructions. This allows the notebook to find the project files. (Allow all)

3. Load the Base Model (Section 4):

- Find and run the cell under **Section 4: LOAD MODEL WITH UNSLOTH**.
- This step downloads the original 7-billion parameter base model and may take a few minutes.

4. Load the Fine-Tuned Model (Section 9):

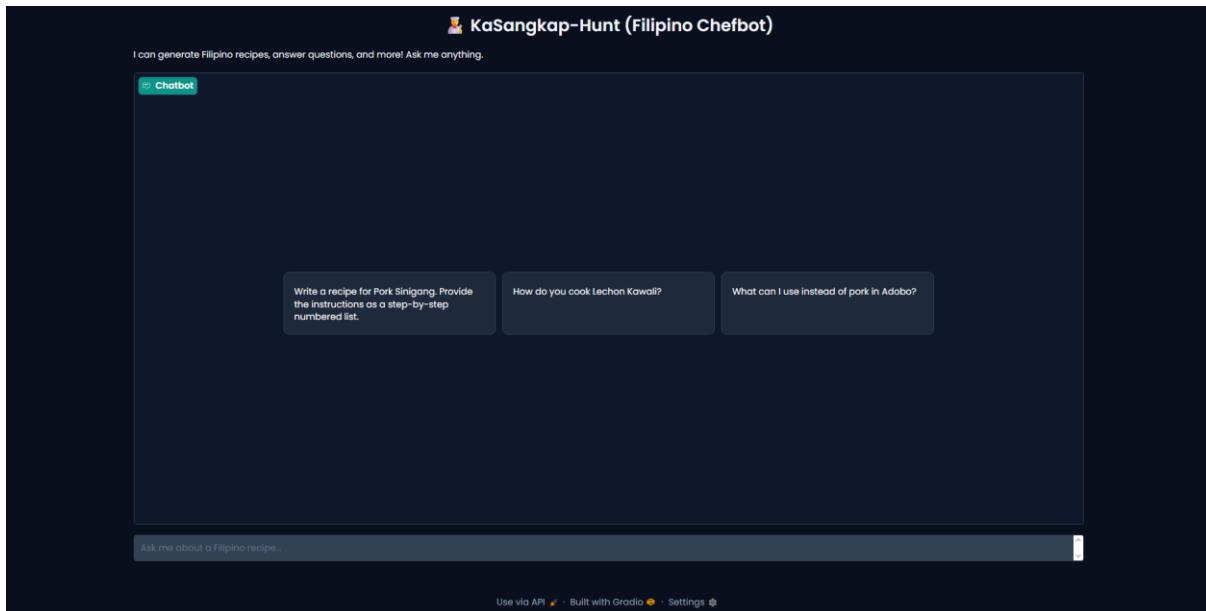
- Find and run the cell under **Section 9: INFERENCE (LOAD MODEL FROM DRIVE AND TEST)**.
- This crucial step loads the custom-trained "Filipino Chef" adapter from Google Drive and merges it with the base model. Wait for the message “ LoRA adapter loaded successfully.”

5. Launch the Gradio Web App (Section 10):

- Finally, scroll to the last cell under **Section 10: INTEGRATE WITH GRADIO UI**.
- Run this cell. After a few moments, the cell output will print a public URL that looks like <https://....gradio.live>.

Step 4: Use the Chefbot

1. Click on the public gradio.live URL.
2. The "KaSangkap-Hunt" web interface will open in a new browser tab, ready for you to interact with it.



KaSangkap-Hunt Chefbot User Interface via Gradio

5. Results and Evaluation

The final model demonstrates a high level of proficiency and generalization.

5.1. Performance on Core Tasks

The model can accurately and coherently generate recipes that were part of its training data, such as Adobo and Sinigang. The formatting is correct, and the instructions are logical.

Sinigang na Hipon recipe

Ingredients:

- 1 lb. shrimp head and shell removed (hipon)
- 2 cups kangkong leaves with stalks cut into 3 inch length
- 4 pieces long green chili siling pansiag
- 6 to 8 tablespoons sinigang mix or 10 pieces tamarind sliced
- 1 medium yellow onion wedged
- 2 medium tomatoes quartered
- 5 to 7 cups water
- 2 teaspoons fish sauce patis

Instructions:

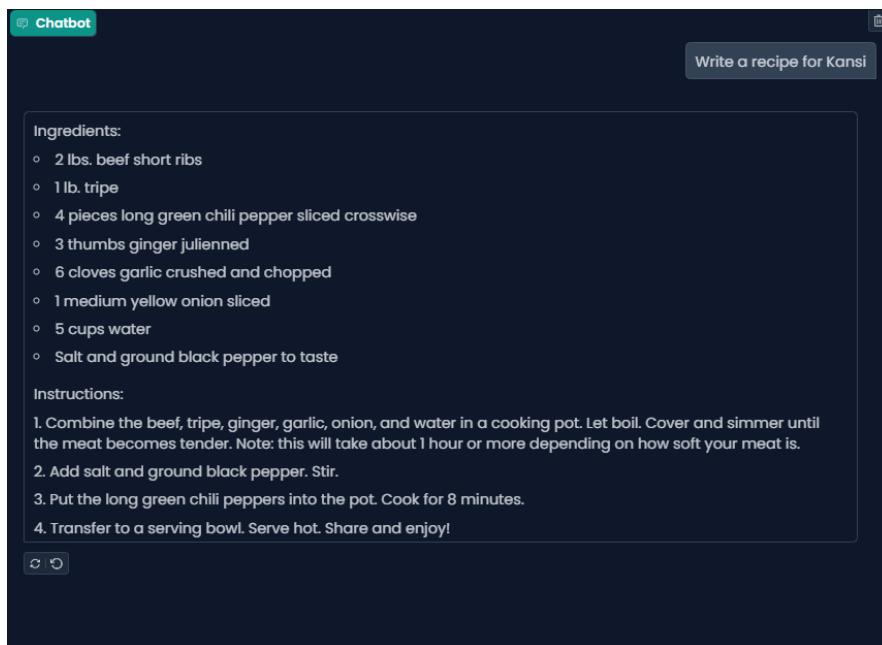
1. Pour the water in a cooking pot. Let boil.
2. Add the tomato, onion, and long green chili. Cover and continue to boil in low heat for 10 minutes.
3. Put the shrimps into the pot. Cook for 5 minutes.
4. Add the sinigang mix or tamarind. Stir. Continue to cook until the liquid reduces to half.
5. Put the kangkong leaves into the pot. Cover and turn the heat off. Let it stay covered for at least 5 minutes. Note: this will allow the residual heat of the soup to cook the kangkong properly.
6. Transfer to a serving bowl. Serve with fish sauce as condiment.
7. Share and enjoy!

Sinigang na Hipon Recipe prompt

5.2. Performance on Advanced Tasks

The model exhibits impressive generalization capabilities on prompts it was not explicitly trained for.

- **Problem-Solving:** When asked for a recipe using leftover ingredients, it correctly suggested a plausible dish and generated a simple, logical recipe.
- **Creative Adaptation:** It successfully adapted a traditional Adobo recipe for a modern air fryer and even generated a novel, creative dipping sauce to accompany it.
- **General Knowledge:** When prompted for "Kansi," a regional dish not in the dataset, it correctly inferred its nature as a hybrid of Bulalo and Sinigang and generated a remarkably accurate recipe.



Kansi Recipe (Regional Dish not in dataset)

5.3. Limitations

- **Knowledge Cutoff:** The model's knowledge is limited to its training data. It cannot generate recipes for dishes it has no information on (e.g., non-Filipino cuisine) without hallucinating.
- **Computational Latency:** Running on a free Colab GPU, response generation can take 10-30 seconds due to the model's large size and the hardware limitations.

6. Conclusion and Future Work

6.1. Project Summary

The KaSangkap-Hunt project successfully achieved its objectives. A 7-billion parameter language model was effectively fine-tuned on a custom dataset, resulting in a capable and creative AI assistant for Filipino cuisine. The final product is a functional, interactive web application that serves as a powerful demonstration of the potential of fine-tuned LLMs in specialized domains.

6.2. Potential Enhancements

- **Dataset Expansion:** The model's intelligence could be further enhanced by adding recipes for desserts, drinks, and regional variations.
- **Integration with Web Search:** The model could be given the ability to perform real-time web searches to answer questions about recipes not in its training data.
- **Persistent Deployment:** The application could be deployed on a permanent cloud service (like Hugging Face Spaces or a dedicated cloud server) for 24/7 availability.