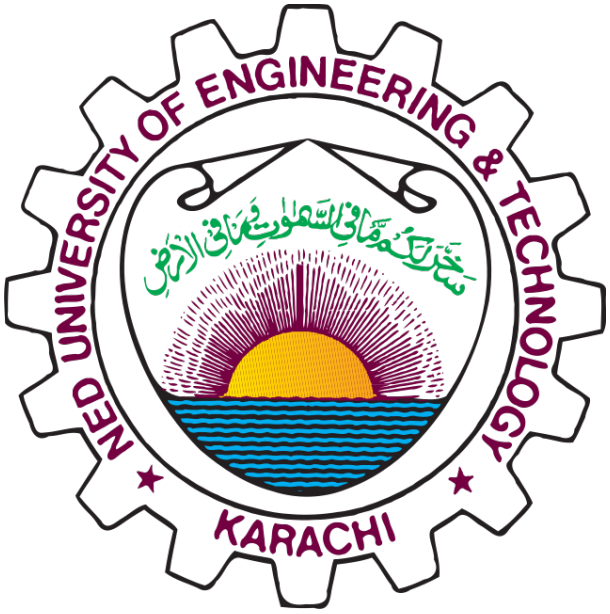

COMPLEX COMPUTING PROBLEM

AI-Based Recipe Generation System

Artificial Intelligence & Expert Systems | CT-361

Group Members:

Rehan Ur Rehman Sharif – CT-22084
Maha Jameel – CT-22055
Muhammad Taha – CT-22075

Contents

1 Introduction

Artificial Intelligence has made it possible to automate content creation tasks that previously required human creativity. In this project, we explore Natural Language Generation (NLG) using AI to produce complete cooking recipes based on user-provided ingredients and preferred dish types, with special consideration for international students and those with dietary restrictions.

2 Problem Statement

The core problem we are tackling primarily affects people traveling or living abroad, particularly foreign exchange students and those who have relocated for studies. In unfamiliar environments, finding proper, viable food options becomes challenging, especially for Muslims concerned with halal dietary restrictions. The difficulty in sourcing appropriate ingredients and preparing meals can lead to significant mental stress and nutritional compromises.

Currently, individuals in these situations must either:

- Spend excessive time researching local ingredients and recipes
- Compromise their dietary preferences or nutritional needs
- Rely on expensive imported or specialty foods
- Develop cooking skills through trial and error

Our solution aims to alleviate these challenges by providing immediate, culturally-sensitive recipe suggestions based on locally available ingredients.

3 Proposed Solution

We propose an AI-powered recipe generation system that helps users:

- Create viable meal plans from available local ingredients
- Maintain dietary and religious requirements (halal, kosher, vegetarian, etc.)
- Adapt recipes to regional ingredient availability
- Learn cooking techniques suitable for their living situation
- Relieve general meal planning and prepping stress

4 Technical Terminology

4.1 Computer Vision Components

YOLO (You Only Look Once) Real-time object detection system used to identify ingredients from images. *Code Reference:*

```
model = YOLO("models/yolov8n.pt")  # Load nano variant
results = model("ingredients.jpg")  # Detection
```

Justification: Chosen for balance of speed/accuracy on consumer hardware.

Bounding Box Rectangular border around detected objects with confidence score. *Code Reference:*

```
boxes = results[0].boxes.xyxy  # Coordinates
conf = results[0].boxes.conf   # Confidence values
```

Justification: Essential for verifying detection validity.

4.2 Natural Language Components

LLaMA (Large Language Model Meta AI) Open-weight LLM used for offline recipe generation. *Code Reference:*

```
llm = Llama(model_path="phi-2.Q4_K_M.gguf")  # 4-bit quantized
```

Justification: Provides privacy-focused alternative to cloud APIs.

Prompt Engineering Structuring inputs to guide AI outputs. *Code Reference:*

```
prompt = f"Create HALAL {dish_type} using: {ingredients}"
```

Justification: Ensures culturally appropriate recipes.

4.3 System Design Terms

Hybrid Input Pipeline Combination of visual detection + manual text entry. *Code Reference:*

```
final_ingredients = detected + manual_input.value.split(',')
```

Justification: Addresses spice recognition limitations.

Quantization Reducing model precision to save memory. *Justification*: Enables LLaMA on 8GB RAM (4-bit vs 16-bit original).

5 Key Features

The system implements several innovative features to address culinary challenges in constrained environments:

5.1 Core Functionality

- **Ingredient Detection**:
  - YOLOv8-based visual recognition of common ingredients
  - Multi-model support (nano to xlarge) for varying hardware capabilities
  - Confidence thresholding to minimize false positives
- **Adaptive Input Methods**:
  - Hybrid image+text input system
  - Manual entry for spices and unidentifiable items
  - Bulk ingredient entry via comma-separated lists

5.2 Recipe Generation

- **Dish Customization**:
  - Category selection (main course, dessert, etc.)
  - Dietary preference filters (halal, vegan, etc.)
  - Cultural adaptation of recipes
- **AI-Powered Suggestions**:
  - GPT-3.5 for cloud-based generation
  - LLaMA (phi-2) for offline functionality
  - Context-aware recipe structuring

5.3 User Experience

- **Privacy-Centric Design**:
  - Optional completely offline mode
  - No ingredient image storage
  - Local preference caching
- **Hardware Awareness**:
  - Optimized for i5/8GB systems
  - Model variant selection
  - Fallback mechanisms

6 Implementation Approaches

6.1 Common Assumptions

All implementations share these fundamental assumptions:

- Basic cooking amenities (pots, pans, utensils) are always available
- Ingredient quantities are not measured (recipes provide relative proportions)
- Spices/herbs with similar visual appearance cannot be reliably classified
- Privacy concerns limit photo-based ingredient detection
- Hardware constraints (i5 CPU, 8GB RAM) affect model choices
- Manual text input is necessary for non-visual ingredients

6.2 Proof of Concept

The simplest implementation demonstrating core functionality:

```
# Prerequisites:
# pip install torch torchvision torchaudio pandas matplotlib ipywidgets
    pillow

import torch
import pandas as pd
from PIL import Image
from pathlib import Path

# Load YOLOv5s model
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)

def detect_ingredients(image_path):
    results = model(Image.open(image_path))
    return list(set(results.pandas().xyxy[0]['name'].tolist()))

def generate_recipe(ingredients, dish_type):
    template = {
        'Main Course': [
            "Prepare ingredients",
            "Cook in pan",
            "Combine with spices",
            "Serve hot"
        ]}
    return "\n".join(f"{i}. {step}"
           for i,step in enumerate(template[dish_type],1))
```

Listing 1: Basic Implementation

Advantages:

- Minimal dependencies
- Fast execution
- No internet required

Limitations:

- Hardcoded recipes
- Limited ingredient recognition

Summary:

- This approach serves as a basic, extremely generic skeleton of the application. It gives cookie-cutter responses in relation to the type of dish that was selected, while adding in the ingredients to the general output to give a basic feel of involvement. As such, it is not much on its own, but serves as a skeletal guideline for the coming versions.

6.3 LLaMA Local Version

Our main presented implementation balancing privacy and functionality:

```
from llama_cpp import Llama
from ultralytics import YOLO
import ipywidgets as widgets

# Load models
llm = Llama(model_path="models/phi-2.Q4_K_M.gguf")
yolo_model = YOLO("models/yolov8n.pt")

def generate_recipe_llama(ingredients, dish_type):
    prompt = f"Create halal {dish_type} using: {ingredients}"
    return llm(prompt=prompt, max_tokens=512)["choices"][0]["text"]

# UI Components
manual_input = widgets.Textarea(description="Extra Items:")
dish_dropdown = widgets.Dropdown(options=["main course", "dessert"])
```

Listing 2: Intermediate Implementation

Advantages:

- Fully offline
- Privacy-preserving
- Moderate hardware needs

Limitations:

- Smaller model capacity
- Occasional incoherence

Summary:

- This approach uses the phi-2 Q4 model from LLaMA as a basic generator LLM in order to give a recipe to the user, while utilizing the data input through the means of images and text to generate a viable recipes from available resources.

6.4 GPT AI-Based Approach

Industry-level potential with cloud integration:

```
import openai
from ultralytics import YOLO

openai.api_key = os.getenv("OPENAI_API_KEY")
yolo_model = YOLO("models/yolov8x.pt")

def generate_recipe_gpt(ingredients, dish_type):
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{
            "role": "user",
            "content": f"Create halal {dish_type} using: {ingredients}"
        }]
    )
    return response.choices[0].message.content
```

Listing 3: Advanced Implementation

Advantages:

- Professional-quality output
- Fast response times
- Dietary customization

Limitations:

- Requires internet
- API costs
- Privacy concerns

Summary:

- This approach uses the OpenAI API in order to invoke a prompt response from GPT3.5-Turbo model, the 24/7 end user available GPT for personal use. Its basic web searching and general rationalization approach do give it an edge over other models, at the cost of demanding consisten internet connectivity.

7 Technical Comparison

Feature	PoC	LLaMA	GPT
Offline Capable	Yes	Yes	No
Recipe Quality	Basic	Moderate	High
Privacy	High	High	Medium
Speed (8GB RAM)	0.5s	5-10s	2-3s
Halal Awareness	No	Basic	Advanced

Table 1: Implementation Comparison

7.1 System Design

Our architecture follows a modular approach:

- **Ingredient Detection**: YOLO-based visual recognition
- **Recipe Generation**: Three-tiered implementation
- **User Interface**: Jupyter widgets for academic presentation
- **Dietary Filters**: Special handling for halal requirements

8 Results and Output Samples

8.1 GPT-3.5 Turbo Implementation

Key characteristics of GPT-generated output:

- Culturally-specific preparation methods
- Detailed spice measurements
- Sequential cooking steps with timing estimates
- Professional recipe structure

8.2 LLaMA Local Implementation

Key characteristics:

- Simplified ingredient quantities
- Basic preparation steps
- Functional structure

Input Ingredients	Generated Recipe
• Bitter gourd • Chicken minced • Tomatoes • Onions • Spices • Turmeric • Salt and pepper • Oil • Potatoes	**Spiced Chicken-Stuffed Bitter Gourd Curry with Potatoes** **Ingredients:** • 4 medium bitter gourds (karela) • 300g chicken mince • 2 medium onions • 2 tomatoes • 2 potatoes • ½ tsp turmeric • 1 tsp red chili powder • 1 tsp garam masala • 3 tbsp oil **Method:** 1. Prepare bitter gourd by deseeding and optional salt treatment 2. Make chicken stuffing with onions and spices 3. Stuff gourds and shallow-fry until golden 4. Prepare tomato-onion curry base with potatoes 5. Simmer stuffed gourds in curry for 20-25 minutes

Table 2: GPT-3.5 Turbo Output Example

Input Ingredients	Generated Recipe
• Tomato • Potato • Cabbage • Mayonnaise • Eggs • Pepper • Salt	**Vegetable Salad with Egg Dressing** **Ingredients:** • 2 tomatoes, sliced • 1 large potato, sliced • 1 head cabbage, chopped • ¼ cup mayonnaise • 2 eggs • ½ tsp black pepper • ½ tsp salt • ¼ cup olive oil • ¼ cup vinegar **Method:** 1. Combine vegetables in large bowl 2. Whisk dressing ingredients separately 3. Toss salad with dressing 4. Adjust seasoning 5. Serve immediately

Table 3: LLaMA Local Model Output Example

8.3 Comparative Analysis

Characteristic	GPT-3.5	LLaMA
Recipe Complexity	High	Medium
Cultural Adaptation	Excellent	Basic
Step Detail	Comprehensive	Concise
Language Fluency	Professional	Functional

Table 4: Output Quality Comparison

9 Limitations and Future Work

- Current Limitations:
  - Visual recognition of similar spices
  - Hardware constraints
  - Manual ingredient entry
- Future Improvements:
  - Hybrid offline/online mode
  - Mobile app implementation
  - Real-time fridge ingredient detection

10 Conclusion

The project demonstrates three viable approaches to AI recipe generation, with the LLaMA implementation offering the best balance for international students. Future work should focus on improved ingredient detection from real-world fridge contents to enhance usability.