

Pixel to Plate

Student Name: Ishaan Awasthy
Roll Number: 2021054

Student Name: Gunjan Dabas
Roll Number: 2021253

*BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering (Ishaan)
and Computer Science and Applied Mathematics (Gunjan)*

Date: 5th May 2025

BTP Track: Engineering Track

BTP Advisor: Dr. Ganesh Bagler

**Indraprastha Institute of Information Technology
New Delhi**

Student's Declaration

I hereby declare that the work presented in the report entitled “**Pixel To Plate**” submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology in Computer Science & Engineering (Ishaan)* and *Computer Science and Applied Mathematics (Gunjan)* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr.Ganesh Bagler**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

Ishaan Awasthy and Gunjan Dabas
(student's name)

Place & Date: 1/05/2024

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Dr.Ganesh Bagler
(advisors' name)

Place & Date:

Abstract

This project, *Pixel to Plate*, presents an end-to-end system that bridges computer vision and natural language processing to automate personalized recipe generation from images of ingredients. The first phase focuses on object detection, where YOLOv8x was fine-tuned on the AI-Cook dataset after comprehensive exploratory data analysis (EDA) addressing dataset quality, class imbalance, and object co-occurrence. YOLOv8x achieved a precision of 0.970, making it the selected model for robust ingredient identification.

The second phase evaluates four large language models—LLaMA, Falcon, GEMMA, and Phi—for zero-shot recipe generation based on the detected ingredients. Among them, LLaMA 3.1 produced the most coherent and contextually appropriate recipes. Prompt tuning was employed to improve generation quality and reduce inference overhead, achieving a balance between performance and efficiency.

To support real-world utility, the system incorporates agentic behavior that dynamically adapts recipes to user-specific constraints such as available ingredients, dietary restrictions, and preparation time. This context-aware generation enhances usability and personalization. Future directions include integrating quantity estimation, nutritional analysis, and deployment on interactive platforms.

This interdisciplinary project demonstrates the effectiveness of integrating cutting-edge object detection with prompt-tuned language models for intelligent culinary applications.

Keywords: Object Detection, YOLOv8, Recipe Generation, Prompt Tuning, Large Language Models, AI-Cook Dataset, Natural Language Processing, Computer Vision, Culinary Applications

Acknowledgments

We would like to express our deepest gratitude to our BTP advisor, Dr. Ganesh Baglar, for his invaluable guidance, support, and encouragement throughout the duration of this project. His expertise and insightful feedback were instrumental in shaping our approach and helping us achieve our goals.

We are also extremely thankful to PhD Mansi Goel for her continuous support and encouragement, particularly in the areas of computer vision and natural language processing. Her knowledge and insights were crucial in refining our model designs and understanding the technical intricacies of this project.

Additionally, we would like to thank Madhavi Ma'am for her support and guidance throughout the project, helping us overcome challenges and ensuring we stayed on track towards completing this BTP successfully.

Finally, we extend our appreciation to the entire faculty of the Computer Science and Engineering Department at Indraprastha Institute of Information Technology, Delhi, for their constant support, encouragement, and academic resources throughout this research endeavor.

Work Distribution

The work distribution between us was equal, with both contributors equally involved in all aspects of the project. Both Ishaan Awasthy and Gunjan Dabas contributed to the object detection phase, model evaluation, and analysis of the results. The recipe generation phase was also a collaborative effort, with both working on the evaluation of different large language models and their performance metrics.

We would like to extend our gratitude to the entire faculty and staff of the Indraprastha Institute of Information Technology, New Delhi, for their support and encouragement.

Contents

1	Introduction	5
1.1	Motivation	5
1.2	Problem Statement	5
2	Research on Existing Work	6
2.1	Introduction	6
2.2	Methodology	6
2.2.1	1. Image Detection	6
2.2.2	2. Recipe Generation	7
2.2.3	3. Modular API Pipeline	7
2.3	Results	7
2.4	Discussion	8
2.5	Conclusion	8
3	Work Done	9
3.1	Object Detection	9
3.1.1	Dataset Preparation	9
3.1.2	Exploratory Data Analysis (EDA)	9
3.1.3	Outline of Object Detection Models	10
3.1.4	Training Details	11
3.1.5	Evaluation Metrics	11
3.1.6	Results	12
3.1.7	Insights and Key Observations	12
3.1.8	Conclusion	12
3.2	Text Generation	16
3.2.1	Overview	16
3.2.2	Brief On LLM used	16
3.2.3	Methodology	17
3.2.4	Analysis	17

3.2.5	Results/Overall Scores	18
4	Work Done Winter 2025	20
4.1	LLM Fine Tuning	20
4.2	Tools and Technologies	20
4.2.1	YOLOv8 (Ultralytics)	20
4.2.2	Transformers Library (Hugging Face)	20
4.2.3	LLMs (Large Language Models) Explored	21
4.2.4	SLMs (Smaller Language Models) Explored	21
4.2.5	Prompt Tuning Framework	22
4.2.6	Google Colab Pro	22
4.2.7	Python PyTorch	22
4.3	Methodology	22
4.3.1	Step 1: LLM Fine-Tuning	23
4.3.2	Step 2: SLM Fine-Tuning	23
4.3.3	Step 3: Prompt Tuning	24
4.3.4	Step 4: Agentic AI with Prompt Tuning	28
4.4	Challenges Faced	28
4.4.1	Step 1: LLM Fine-Tuning	28
4.4.2	Step 2: SLM Fine-Tuning	28
4.4.3	Step 3: Prompt Tuning	28
4.4.4	Step 4: Agentic AI	29
4.5	Final System Architecture	29
4.5.1	Input	29
4.5.2	Ingredient Detection	29
4.5.3	Recipe Generation	29
4.5.4	Output	29
4.6	Evaluation and Analysis	29
4.7	Conclusion and Future Work	30
5	Language Translation	31
5.1	Introduction	31
5.2	5.2 Translation Models Used	31
5.3	Input Pipeline	32
5.4	Fine-Tuning Attempts	32
5.5	Quantitative Evaluation	32
5.6	Qualitative Evaluation	34
5.7	Conclusion	35

6	Streamlit-Based Personalized Recipe and Health Planner Interface	36
6.1	Introduction	36
6.2	Key Features and Functionalities	36
6.2.1	Streamlit-Based UI	36
6.2.2	User Input Collection	36
6.3	Backend Architecture	37
6.3.1	DeepSeek LLM Integration	37
6.4	Multilingual Support	38
6.4.1	Translation to Hindi	38
6.5	Dietary Guidance with Gemini Agent	38
6.5.1	Agent Configuration	38
6.5.2	Output	38
6.6	Interface Workflow	39
6.7	Final Workflow Snapshots	40
6.8	Conclusion	48

Chapter 1

Introduction

1.1 Motivation

The motivation for this project stems from the need to integrate distinct AI capabilities, such as image recognition and natural language generation, to solve real-world problems. While advancements in these areas have been significant, their combined potential for creating practical, user-centric solutions remains underutilized. This project addresses this gap by tackling a common problem: generating recipes from the ingredients available in a refrigerator.

Determining what to cook often requires creativity, dietary considerations, and careful planning. Automating this task simplifies meal preparation, making it more efficient and accessible. By generating personalized, context-aware recipes, the system enhances convenience while addressing individual preferences such as dietary restrictions, cost, and preparation time.

The project's modular design is another key motivation. Unlike monolithic systems, a modular approach enables independent optimization and scalability. Each component evolves independently, allowing seamless integration of improvements in object detection or language generation technologies. This flexibility ensures adaptability and long-term relevance.

Beyond its immediate application, the project demonstrates how AI can be tailored to niche domains, offering a blueprint for similar efforts in areas like healthcare or education. By integrating vision and language capabilities, this work sets a precedent for creating intelligent systems that enhance daily life.

1.2 Problem Statement

Households often need help deciding what to cook using available ingredients, requiring effort to balance dietary restrictions, budgets, and preparation time. Existing tools lack adaptability and fail to provide personalized solutions. The challenge is to create a system that identifies ingredients from images and generates context-aware recipes. This requires integrating advanced object detection with language generation in a modular, scalable framework to deliver practical and user-centric solutions.

Chapter 2

Research on Existing Work

The AI community has adopted multisensory or multimodal techniques to enhance the current generation of AI models, with the aim of achieving more comprehensive intelligent understanding. Integrating language and imagery is a well-established approach for tasks such as image captioning or generating visuals from descriptions. The Multimodal And Modular Ai Chef: Complex Recipe Generation From Imagery, addresses a growing need in AI applications to combine multimodal capabilities—specifically image recognition and language generation—to solve practical problems. The authors propose a modular framework, which effectively separates the tasks of object detection and recipe generation. Using YOLOv5 for object detection and OpenAI’s GPT-3.5 for recipe generation, the system demonstrates how combining specialized models can outperform monolithic multimodal approaches in specific use cases.

2.1 Introduction

- Multimodal learning, which integrates sensory inputs like text and imagery, has become a cornerstone of AI development.
- Existing models like CLIP, GATO, and LAION-5b exhibit capabilities in image-to-text tasks but lack coherence in maintaining context for complex applications like recipe generation.
- The study explores a modular alternative where image detection models identify objects and pass them to a Large Language Model (LLM) for text generation.
- The main goal is to create recipes tailored to constraints such as dietary restrictions, preparation time, cost, and portion sizes.

2.2 Methodology

2.2.1 1. Image Detection

- Utilized YOLOv5 (small) as the object detection model, trained on the *ai-cook-lcv4d* dataset, which includes 3,050 images of 30 common refrigerator items.
- Training involved a split of 2,896 images for training, 103 for validation, and 51 for testing.

- Pre-processing included augmentations such as rotation, exposure adjustments, and noise addition.
- Achieved a mean average precision (mAP) of 95.2% and high recall rates for ingredient detection.

2.2.2 2. Recipe Generation

- Used OpenAI’s GPT-4 model for text generation.
- The input was a delimited list of detected ingredients, and the output included:
 - Recipe title.
 - Ingredients list with quantities.
 - Cooking instructions with step-by-step details.
 - Approximate preparation and cooking times.
- Recipes adhered strictly to the provided ingredient list without external supplementation.
- Variants were generated for specific constraints such as vegan, keto, or lactose-intolerant diets.

2.2.3 3. Modular API Pipeline

- The modular approach separates the image detection and text generation stages, enabling independent updates to each component.
- This design reduces computational overhead and allows rapid adaptation to evolving models.
- The system can handle scenarios like ingredient expiration tracking, meal planning, and waste reduction.

2.3 Results

- Generated a 100-page recipe book featuring recipes based on 30 primary ingredients derived from 2,000 refrigerator images.
- Achieved high accuracy in ingredient detection, as reflected in the multi-class confusion matrix.
- Demonstrated versatility in generating diverse recipe variations:
 - Adjusting for dietary preferences (vegan, keto, lactose-free).
 - Optimizing for cost-efficiency and seasonal availability.
 - Customizing portion sizes and meal types.
- Showcased the ability to refine recipes interactively based on user feedback, leveraging GPT-4’s long conversational memory.

2.4 Discussion

- The modular design supports practical applications by optimizing resource use and enabling real-time edge deployment.
- Potential future enhancements include:
 - Incorporating reinforcement learning to improve recipe generation based on user feedback.
 - Expanding to include global cuisines and advanced food pairing principles.
 - Adding operational research capabilities for cost optimization in large-scale settings like restaurants.
- Highlights the environmental benefits of reducing food waste through efficient inventory utilization.

2.5 Conclusion

The study demonstrates a scalable, practical solution for AI-based recipe generation by combining state-of-the-art image detection and language generation models. The modular API approach outperforms monolithic models in adaptability and efficiency, enabling complex human-like culinary tasks in real-world scenarios.

Chapter 3

Work Done

3.1 Object Detection

The primary focus of this phase was to identify the most suitable object detection model for the AI-Cook dataset. This model will be used to detect ingredients and subsequently facilitate automated recipe generation. Below are the detailed steps and outcomes of this phase:

3.1.1 Dataset Preparation

The dataset was sourced from Roboflow and thoroughly analyzed to ensure its readiness for model training. Key aspects of the dataset are as follows:

- **Structure:** The dataset consists of 3,040 images with corresponding label files, organized into training (2,896 images), validation (93 images), and test sets (51 images).
- **Image Properties:** All images have uniform dimensions of 640x640 pixels, ensuring consistency during training.
- **Class Distribution:** Class imbalance was noted, with some categories underrepresented. Frequent co-occurrence of specific class pairs (e.g., bread & butter) provides valuable insights for future tasks.
- **Bounding Boxes:** Bounding box sizes varied, with most being relatively small compared to the image size. This was confirmed by analyzing the bounding box-to-image size ratio.
- **Object Density:** Most images contain multiple objects, which enhances the dataset's representativeness for real-world scenarios.

3.1.2 Exploratory Data Analysis (EDA)

Detailed EDA was conducted to address potential issues such as class imbalance and data quality. Highlights include:

- No duplicate or blurry images were found in the dataset.
- Analysis of object co-occurrence and density provided insights into natural groupings of ingredients.

3.1.3 Outline of Object Detection Models

To identify the optimal object detection model for ingredient recognition in refrigerator images, we evaluated five YOLO (You Only Look Once) variants. These models represent incremental improvements in object detection technology and were trained under identical conditions to ensure a fair comparison.

Overview of YOLO Variants

YOLO models are renowned for their efficiency and real-time performance in object detection tasks. Below, we provide detailed insights into the architectures and advancements of the variants evaluated:

1. YOLOv5 (s and x Variants)

- YOLOv5 introduced several architectural improvements over YOLOv4, including:
 - Use of the CSPDarknet53 backbone for enhanced feature extraction.
 - Mosaic data augmentation to improve model generalization.
 - An improved anchor-based detection mechanism for precise bounding box localization.
- The **s** (small) variant is designed for computational efficiency, using fewer parameters, making it suitable for edge devices.
- The **x** (extra-large) variant increases the number of layers and parameters, prioritizing accuracy over speed.
- Both variants demonstrated reliable performance but struggled to match the accuracy of more recent models like YOLOv8.

2. YOLOv7

- YOLOv7 builds upon YOLOv4 and introduces Extended Efficient Layer Aggregation Networks (E-ELAN), which:
 - Enhance the learning capacity of the model without significantly increasing computational overhead.
 - Allow better utilization of feature maps for complex object detection tasks.
- This version achieves a superior balance between inference speed and detection accuracy compared to YOLOv5.
- YOLOv7 performs exceptionally well in scenarios requiring high precision, such as identifying small or overlapping objects in cluttered environments like refrigerators.

3. YOLOv8 (s and x Variants)

- YOLOv8 introduces cutting-edge improvements, including:
 - An updated backbone network optimized for feature representation.
 - Anchor-free detection for improved performance on irregularly shaped objects.
 - Enhanced feature pyramid network (FPN) and path aggregation network (PAN) for better multi-scale detection.
- The **s** (small) variant remains resource-efficient, suitable for mobile and edge devices.
- The **x** (extra-large) variant offers state-of-the-art accuracy by leveraging deeper and wider architectures, excelling in highly detailed object detection tasks.

3.1.4 Training Details

- All models were trained on the `ai-cook-lcv4d` dataset, consisting of 3,050 images across 30 ingredient categories.
- Images were pre-processed with augmentations including rotation, exposure adjustment, noise addition, and cut-outs to improve generalization.
- The dataset was split into training, validation, and testing sets in a ratio of 56:2:1.
- Training was conducted for 90 epochs on a single NVIDIA A-100 GPU, ensuring sufficient exposure to the dataset for learning.

3.1.5 Evaluation Metrics

To assess model performance, the following metrics were employed:

- **F1 Score:** Evaluates the balance between precision (correctly identified objects) and recall (completeness of detection). A higher F1 score indicates fewer false positives and false negatives.
- **Precision-Recall Curves:** Visualize the trade-offs between precision and recall at various confidence thresholds, providing insights into model behavior under different scenarios.
- **mAP@0.5:** Measures mean average precision at an IoU threshold of 0.5, quantifying the model's accuracy in detecting objects.

3.1.6 Results

Model	F1 Score (Confidence Threshold)	mAP@0.5
YOLOv5s	0.95 at 0.702	0.964
YOLOv5x	0.95 at 0.615	0.961
YOLOv7	0.96 at 0.710	0.955
YOLOv8s	0.96 at 0.709	0.968
YOLOv8x	0.97 at 0.753	0.970

Table 3.1: Performance Metrics of YOLO Models.

3.1.7 Insights and Key Observations

- **YOLOv5:** While delivering reliable performance, both **s** and **x** variants fell short in terms of precision-recall balance compared to newer models.
- **YOLOv7:** Improved learning capacity and feature aggregation enabled higher F1 scores than YOLOv5, but its mAP@0.5 lagged behind YOLOv8.
- **YOLOv8:**
 - YOLOv8s provided a strong balance of speed and accuracy, ideal for resource-constrained applications.
 - YOLOv8x achieved the best performance with an F1 score of 0.97 and mAP@0.5 of 0.970, excelling in all tested metrics.

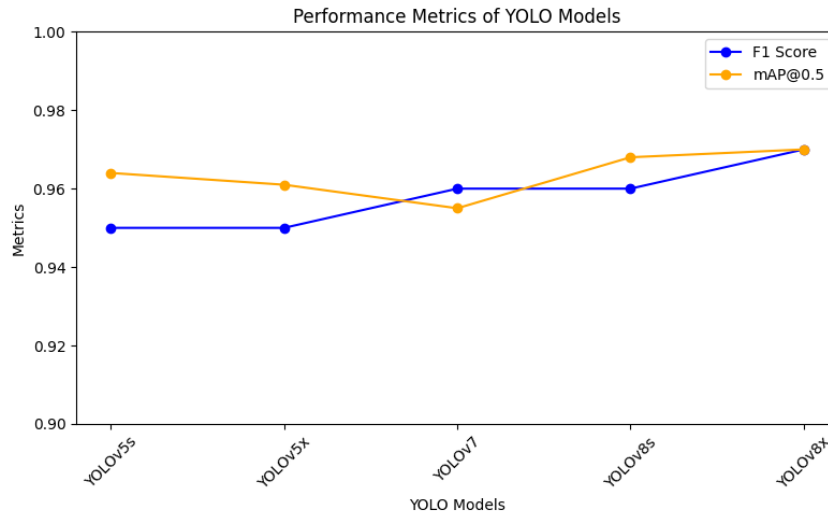
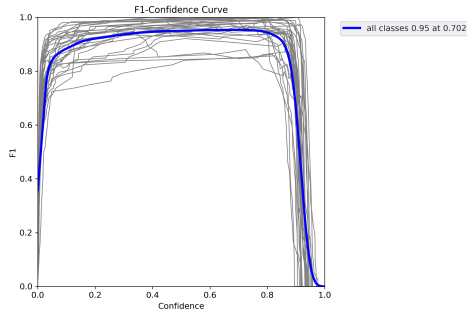


Figure 3.2: YOLO Models' Performance Metrics.

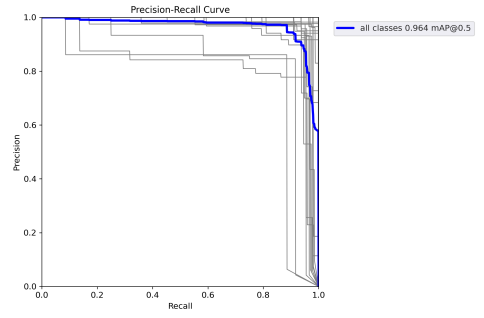
3.1.8 Conclusion

YOLOv8x was selected as the final model for ingredient detection due to its superior performance across all evaluation metrics. Its advanced architecture ensures reliable detection even

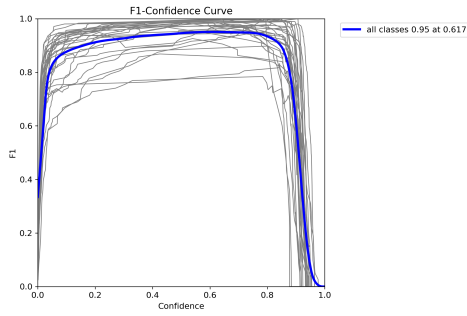
in challenging conditions, providing a robust foundation for subsequent recipe generation tasks. The modular nature of this approach facilitates future scalability and adaptability as object detection technology evolves..



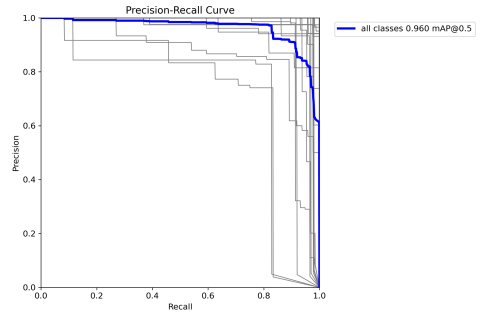
(a) F1 curve V5s



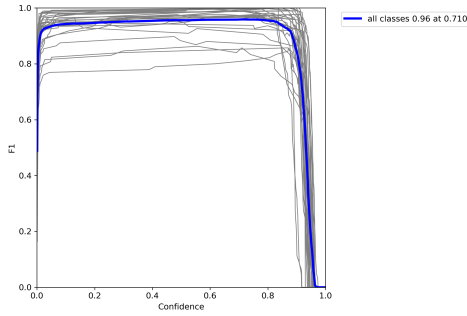
(b) PR curve V5s



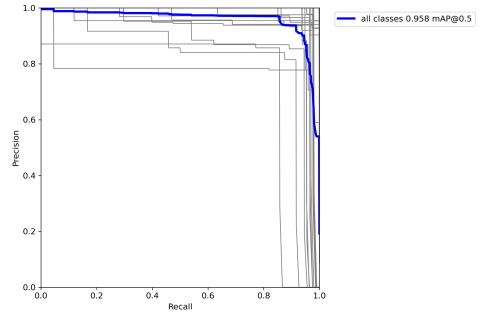
(c) F1 curve V5x



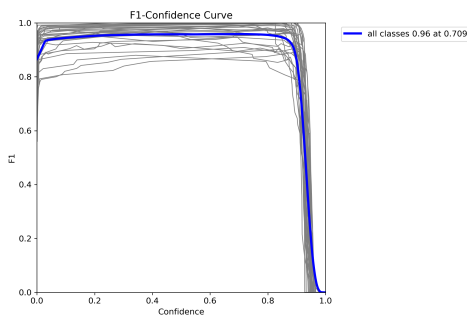
(d) PR curve V5x



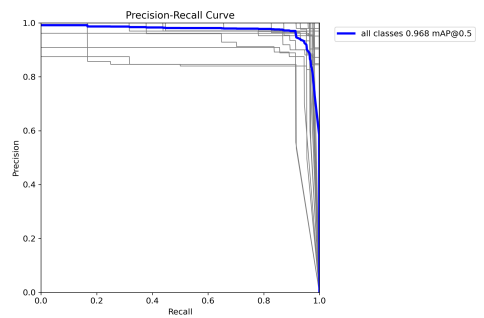
(e) F1 curve V7



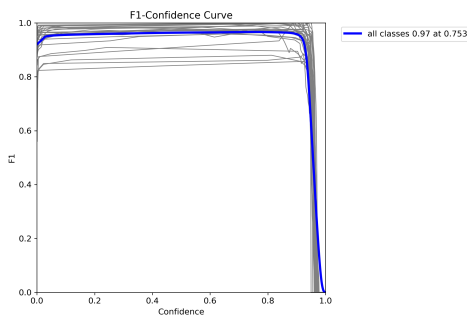
(f) PR curve V7



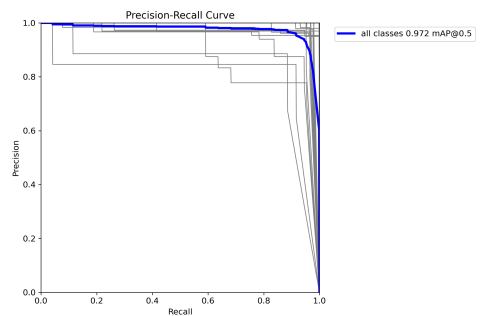
(g) F1 curve V8s



(h) PR curve V8s

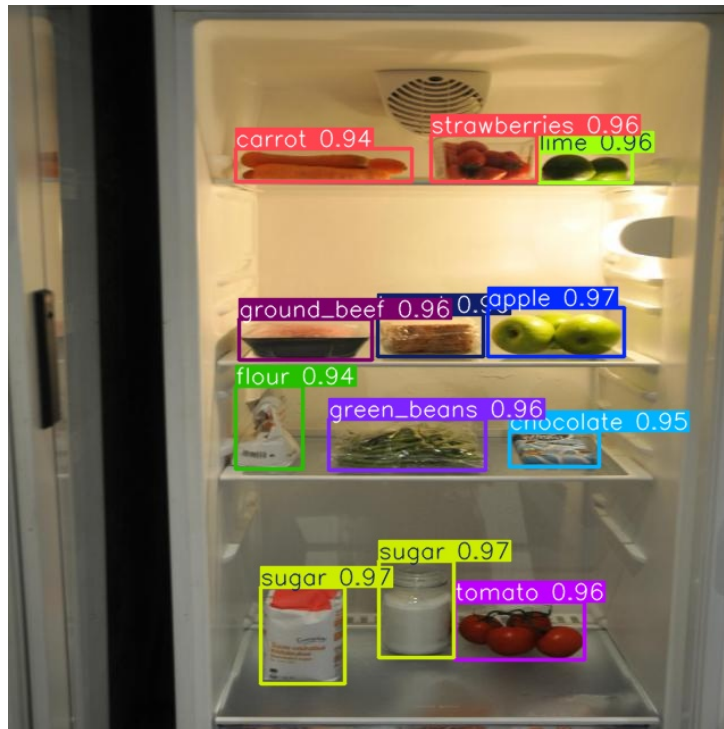


(i) F1 curve V8X

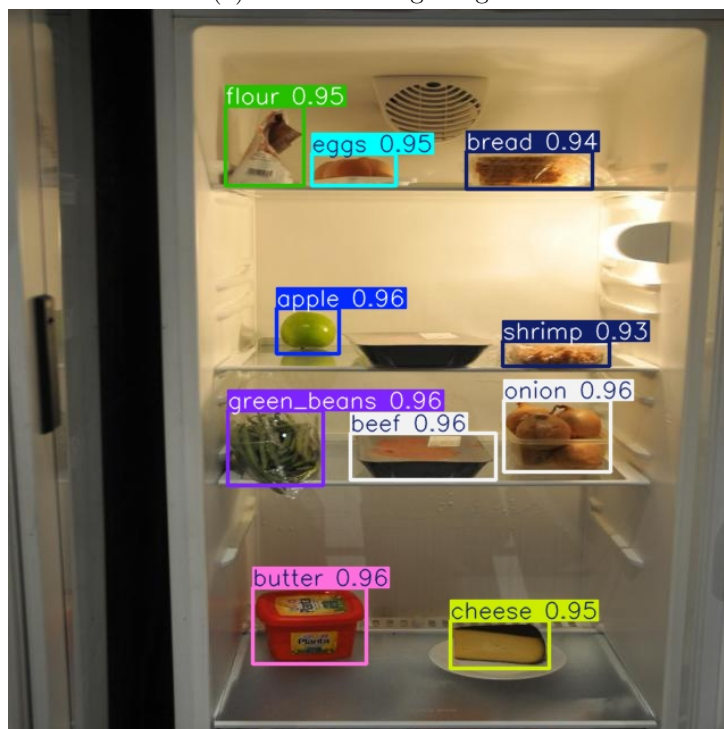


(j) PR curve V8x

Figure 3.1: F1 CURVE AND PR CURVE OF MODELS



(a) Final resulting image 1.



(b) Final resulting image 2.

Figure 3.3: Final resulting images from YOLOv8 Model.

3.2 Text Generation

3.2.1 Overview

In this project, we evaluated four state-of-the-art large language models (LLMs) for their ability to generate recipes in a zero-shot setting. The goal was to assess the quality of generated recipes in terms of relevance, coherence, and completeness without prior task-specific fine-tuning. The models tested were:

- Llama-3-8B-Instruct
- Falcon
- Gemma-2B-it
- Phi-3.5-mini-instruct

3.2.2 Brief On LLM used

1. Llama-3-8B-Instruct

About the Model: Llama-3-8B-Instruct, developed by Meta AI, is a transformer-based model optimized for natural language understanding and generation tasks. It offers high efficiency with fewer parameters compared to larger models, and its instruction-based fine-tuning makes it effective at following prompts. Despite its smaller size, it delivers robust performance for a wide range of tasks, including recipe generation.

Why We Included It for Recipe Generation: Llama-3-8B-Instruct is included for its zero-shot capabilities, allowing it to generate coherent recipes from minimal input. It can creatively combine ingredients and generate structured cooking instructions, making it ideal for diverse recipe generation tasks without the need for task-specific training.

2. Falcon

About the Model: Falcon is an open-source transformer-based model designed for high-performance NLP tasks. It excels at generating fluent, human-like text due to its large model size and extensive training data. Its architecture allows for deep understanding and generation of complex, contextually nuanced text.

Why We Included It for Recipe Generation: Falcon is included for its ability to handle complex ingredient lists and cooking methods. It can generate structured recipes that adapt to various culinary styles and dietary needs, making it highly versatile for recipe creation with logical flow and fluency.

3. Gemma-2B-it

About the Model: Gemma-2B-it is a specialized language model fine-tuned for creative tasks. It excels in generating long-form content while maintaining fluency and coherence. Its focus on creative writing makes it suitable for generating detailed recipes with structured steps.

Why We Included It for Recipe Generation: Gemma-2B-it is included for its ability to create clear, detailed recipes based on ingredient lists. Its creativity and coherence help in producing novel recipes with easy-to-follow instructions, making it ideal for personalized recipe generation with specific dietary preferences.

4. Phi-3.5-mini-instruct

About the Model: Phi-3.5-mini-instruct is a recent addition with high contextual understanding and natural language generation capabilities. Built on transformer architecture, it performs well across tasks like text generation and summarization, maintaining coherence and logical flow.

Why We Included It for Recipe Generation: Phi-3.5-mini-instruct is included for its ability to maintain a clear, logical sequence in recipe generation. Its zero-shot capabilities and understanding of both ingredients and cooking steps ensure that it generates coherent and realistic recipes, making it an excellent choice for this project.

3.2.3 Methodology

- **Dataset:** Recipes were generated based on a predefined set of ingredients derived from object detection outputs.
- **Evaluation Metrics:**
 - **Coherence:** Logical consistency and clarity in instructions.
 - **Completeness:** Coverage of all supplied ingredients and recipe steps.
 - **Relevance:** Adherence to the input prompt and task requirements.
- **Testing Setup:** Recipes were generated in a zero-shot context, meaning the models had no fine-tuning for culinary tasks.

3.2.4 Analysis

Clarity

Clarity involves evaluating how well the recipe instructions are structured, whether they are easy to follow, and if the preparation steps are clearly defined.

- **Falcon:** Relatively straightforward but occasionally lack detail in preparation steps, making some processes ambiguous.
- **LLaMA Model Recipes:** Offers a clear structure with detailed instructions. Steps are logically ordered and easy to follow.
- **GEMMA-2B-IT Recipes:** Some recipes are clear, but others have complex instructions or overly condensed steps that might confuse inexperienced cooks.
- **Phi:** Similar to Falcon, with variability in clarity. Some recipes are clearer than others, suggesting inconsistency.

Meaningfulness

This parameter evaluates how purposefully the recipes use ingredients and whether the end dish seems reasonable and appetizing.

- **Falcon:** Uses ingredients in a standard way but sometimes combines them into unusual or less appetizing dishes.
- **LLaMA Model Recipes:** Generally creates more coherent and appealing recipes, making good use of the ingredient list to produce dishes that sound tasty.
- **GEMMA-2B-IT Recipes:** While the recipes make use of all ingredients, sometimes the end results sound less appealing or practical.
- **Phi:** Attempts creative uses of ingredients but may result in unconventional dishes that might not appeal to all.

Food Combinations

This looks at whether the recipes consider healthy ingredient pairings and avoid combinations that could cause dietary concerns.

- **Falcon:** Occasionally includes questionable combinations like heavy use of sugars with fats, which can be unhealthy.
- **LLaMA Model Recipes:** Better at avoiding poor food combinations and tends to create recipes with a balance of nutrients.
- **GEMMA-2B-IT Recipes:** Some recipes include potentially problematic combinations, such as excessive use of heavy cream and sugars.
- **Phi:** Similar issues to Falcon, with some combinations potentially leading to unbalanced meals.

3.2.5 Results/Overall Scores

Based on the detailed analysis, the scores for each model are summarized in Table.

Model Name	Clarity (10)	Meaningfulness (10)	Food Combination (10)	Average Score
Falcon	7	6	5	6.00
LLaMA Model Recipes	9	8	8	8.33
GEMMA-2B	7	7	6	6.67
Phi	6	6	5	5.67

Table 3.2: Overall Scores for Recipe Generation Models.

In conclusion, the evaluation of text generation models highlights the varying strengths and weaknesses of each approach. While the **LLaMA Model Recipes** achieved the highest average score, excelling in clarity, meaningfulness, and food combination, other models such as **GEMMA-2B** and **Falcon** demonstrated moderate performance, suggesting room for improvement in balancing these attributes. The **Phi** model, with the lowest average score, indicates potential challenges in optimizing its recipe generation capabilities. These findings underscore

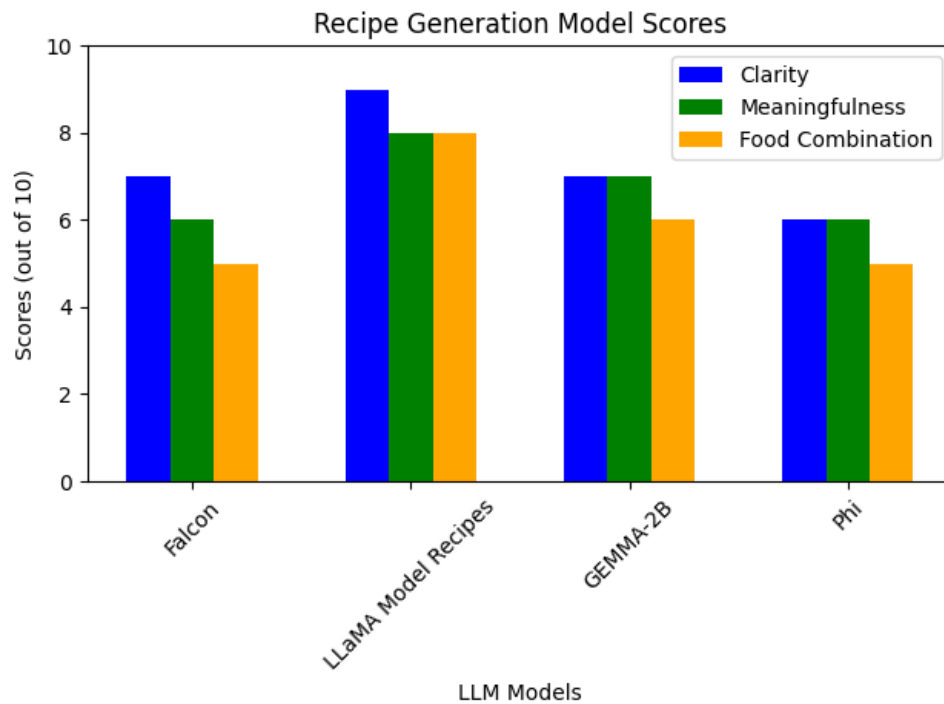


Figure 3.4: Histogram Visualizing Scores for Recipe Generation Models.

the importance of selecting models that align with specific use-case requirements, particularly when generating content that demands both semantic richness and practical relevance. Future efforts could focus on enhancing models' contextual understanding and adaptability to user preferences for improved recipe quality and user satisfaction.

Chapter 4

Work Done Winter 2025

4.1 LLM Fine Tuning

Fine-tuning large language models (LLMs) such as LLaMA 3.1, Falcon, and Phi-2 was one of the core elements of our project. This process involved adapting pre-trained models to generate recipes from the detected ingredients and required handling challenges like resource constraints and data alignment. Our approach was a combination of traditional fine-tuning, prompt tuning, and a few novel strategies to manage time and resource limitations.

4.2 Tools and Technologies

In the development of the personalized recipe generation system, we utilized several advanced tools and technologies that allowed us to combine computer vision and natural language processing (NLP). The integration of these technologies facilitated the creation of an intelligent, adaptable system capable of generating diverse recipes based on available ingredients.

4.2.1 YOLOv8 (Ultralytics)

YOLOv8 (You Only Look Once, version 8), developed by Ultralytics, is a powerful real-time object detection model. For this project, it played an integral role in ingredient detection within images submitted by users. YOLOv8 excels at recognizing multiple objects within a single image, and its speed and accuracy made it ideal for real-time applications. By detecting ingredients and their locations within the image, YOLOv8 helped us extract critical data required for the recipe generation process.

YOLOv8's primary advantage lies in its efficiency. Unlike previous versions of YOLO, YOLOv8 introduces improved algorithms that enhance both detection accuracy and speed. These improvements allow for faster processing and minimal delays, even when dealing with larger datasets or complex images. Furthermore, YOLOv8's robustness allowed us to confidently detect diverse types of ingredients, even those in cluttered or unorganized fridge images.

4.2.2 Transformers Library (Hugging Face)

The Hugging Face Transformers library is a pivotal component in modern NLP and was central to our recipe generation system. Hugging Face offers a wide range of pre-trained models, including

LLaMA 3.1, BERT, T5, and others, which facilitated the implementation of sophisticated natural language processing tasks. By integrating these models, we could leverage their capabilities to fine-tune and adapt them specifically for recipe generation tasks.

The integration of Hugging Face’s Transformers also allowed for seamless scaling of our models. With the library’s support for distributed training, we could scale experiments as needed, making it a versatile and powerful tool in our system’s development.

4.2.3 LLMs (Large Language Models) Explored

Several large language models were explored during the project, each bringing its unique strengths to the table. Among these, LLaMA 3.1 was chosen as a core model due to its flexibility in text generation, while models like Gemma, Phi-2, and Falcon were tested for specific capabilities in recipe generation, summarization, and question-answering.

- **LLaMA 3.1:** LLaMA 3.1 excels at coherent and contextually relevant text generation. This model was used extensively for generating creative and diverse recipes from the detected ingredients.
- **Gemma:** A model well-known for creative and narrative text generation, Gemma was explored to generate varied and novel recipe ideas that could intrigue users looking for innovative culinary inspiration.
- **Phi-2 and Falcon:** These models were more specialized in text summarization and question-answering, respectively, making them valuable for structuring cooking instructions in a clear, step-by-step manner.

These LLMs were evaluated for their ability to generate human-like responses, their creativity in recipe formulation, and their adaptability to diverse ingredients and cooking styles.

4.2.4 SLMs (Smaller Language Models) Explored

Due to the computational constraints faced when fine-tuning LLMs, smaller language models (SLMs) such as T5-small, T5-base, and BERT-base were also tested. While smaller and more resource-efficient, these models offered a different trade-off in terms of output quality and generalization capabilities.

- **T5-small and T5-base:** These models were chosen for their computational efficiency, which allowed us to perform fine-tuning with limited resources. Despite their size, they demonstrated reasonable performance on simpler tasks.
- **BERT-base:** BERT was mainly employed for its understanding and contextual processing of text. It played a role in extracting meaningful instructions for the recipe generation process.
- **Phi-base:** A lighter version of Phi-2, Phi-base was tested for smaller datasets where computational constraints were most severe. Though smaller, Phi-base was able to provide adequate recipe generation output for smaller datasets.

While SLMs were more feasible in terms of resource usage, they struggled with maintaining the complexity and variety required for high-quality recipe generation.

4.2.5 Prompt Tuning Framework

Prompt tuning emerged as an effective solution to fine-tune models with minimal computational overhead. Using Parameter-Efficient Fine-Tuning (PEFT) and Low-Rank Adaptation (LoRA), we could adapt larger models without requiring massive GPU resources.

- **PEFT and LoRA:** These techniques enabled us to train the model more efficiently by modifying only a small portion of the model (the prompt). This approach preserved the base model's knowledge while allowing it to specialize in generating recipes based on ingredient lists.
- **Efficiency Improvements:** With LoRA, we were able to significantly reduce the training time, making it possible to complete fine-tuning within a couple of hours, compared to the days or weeks required for traditional full-model fine-tuning.
- **Adaptation:** The prompt tuning strategy allowed for the model to better understand the context of the user's request (e.g., generating a recipe for 2 people, using vegetarian ingredients).

These tuning methods made it feasible to adapt high-performing LLMs and SLMs for our specific recipe generation task.

4.2.6 Google Colab Pro

Google Colab Pro provided the necessary environment for model experimentation, offering access to GPUs and TPUs for model training. Although Colab Pro had limitations in terms of continuous GPU access and memory, it was sufficient for rapid experimentation and small-scale training sessions. By using Colab Pro, we could iterate on various models and fine-tuning techniques efficiently, which greatly accelerated the overall project timeline.

4.2.7 Python PyTorch

Python was the primary language used for the project, thanks to its vast ecosystem of libraries and its compatibility with various machine learning frameworks. PyTorch, in particular, played a critical role in model training and fine-tuning due to its flexibility and strong support for GPU acceleration.

PyTorch's support for dynamic computation graphs allowed us to make on-the-fly modifications to the training pipeline, a feature that proved valuable when testing various approaches like PEFT and LoRA.

4.3 Methodology

The core of this project lies in the integration of computer vision and natural language processing. Here's a more detailed walkthrough of the steps we took to fine-tune the models and build the system.

4.3.1 Step 1: LLM Fine-Tuning

Fine-tuning large language models for recipe generation was our initial approach. We began with LLaMA 3.1, Falcon, Gemma, and Phi-2, all of which are advanced LLMs capable of text generation. However, fine-tuning these models on recipe-specific data proved to be a challenge due to:

- **Resource Constraints:** LLMs are computationally intensive, requiring powerful GPUs and ample memory. The large memory footprints and computational requirements made it difficult to fine-tune these models on Google Colab Pro.
- **Training Time:** Each training epoch took several hours, and with the need for multiple epochs to achieve reasonable results, training became time-prohibitive.
- **Token Mismatches:** Some models faced issues with token mismatches, which disrupted training stability, especially when processing long recipe instructions.

Despite these challenges, we observed that LLaMA 3.1 produced promising results, generating coherent and diverse recipes when fine-tuned with the appropriate dataset.

4.3.2 Step 2: SLM Fine-Tuning

Given the limitations with LLMs, we explored smaller language models, we experimented with three prominent small models: `phi-2`, `gpt2`, and `t5-small`. Each model was evaluated in terms of its response generation quality, architecture suitability, and computational efficiency. The experiments were conducted using Hugging Face Transformers with prompt-based inputs and minimal preprocessing.

- **Data Preparation:** We carefully curated a dataset of paired ingredients and instructions. This dataset served as the foundation for fine-tuning the SLMs to produce meaningful recipe suggestions.
- **Training Challenges:** While the fine-tuning process was faster with SLMs, the quality of the output was inconsistent. The models produced recipes that were often repetitive or overly simplistic, with little creativity in combining ingredients.
- **Generalization Issues:** SLMs struggled to generate recipes for combinations of ingredients that they hadn't been explicitly trained on, limiting their adaptability to new or unique ingredient sets.

Model-Wise Summary

- **Phi-2** (`microsoft/phi-2`):
 - **Architecture:** Causal Language Model
 - **Purpose:** Prompt completion and creative writing
 - **Insights:** Phi-2 is highly efficient, ideal for real-time low-latency tasks and runs well on limited compute.
- **GPT-2** (`gpt2` and `gpt2-medium`):

- **Architecture:** Decoder-only Transformer
- **Purpose:** Natural text generation and fine-tuning
- **Insights:** Demonstrated good fluency in generation and versatility in multiple NLP tasks. Lightweight fine-tuning was attempted for improved contextual performance.
- **T5-Small (t5-small):**
 - **Architecture:** Encoder-Decoder Transformer
 - **Purpose:** Text-to-text tasks such as summarization, translation, and classification
 - **Insights:** While the model is slower in inference compared to decoder-only models, it handles structured tasks better due to its encoder-decoder design.

All models were run on Google Colab with minimal resource allocation, making them practical choices for low-resource deployment or edge-based inference.

This section summarizes our baseline analysis using SMLs before moving on to more capable foundation models like LLaMA 3.1 and OpenHathi.

4.3.3 Step 3: Prompt Tuning

Prompt tuning provided a more resource-efficient approach to fine-tuning models. Using PEFT and LoRA techniques, we could adapt the model’s responses without changing the core weights. Prompt tuning proved to be the optimal solution, balancing resource constraints with the need for high-quality, diverse recipe generation.

Idea

The goal of this is to evaluate the effectiveness of various prompting strategies for recipe generation using different Large Language Models (LLMs). This includes:

- Designing prompts using Zero-shot, Few-shot, Chain-of-Thought, Tree-of-Thought, and ReAct techniques.
- Filtering input data to focus on vegetarian recipes.
- Generating recipes using LLMs (LLaMA, Gemma, DeepSeek).
- Evaluating outputs using NLP metrics like BLEU and ROUGE.

This evaluation helps in identifying the most effective prompting strategy and model combination for structured and creative text generation tasks like recipes.

Dataset and Preprocessing

The dataset is loaded using the HuggingFace `datasets` library from a CSV file containing recipes. To constrain the task to vegetarian cuisine:

- A list of non-vegetarian ingredients (e.g., "chicken", "mutton", "fish") is defined.
- Recipes containing these ingredients are filtered out.

This filtering ensures consistency in output and fairness across model comparisons.

Prompting Strategies

Six distinct strategies are used to guide the LLMs:

1. **Zero-shot**: No examples provided; the model is directly asked to generate a recipe based on given ingredients.
2. **Few-shot**: A few input-output examples are shown before the actual generation task.
3. **Chain-of-Thought (CoT)**: Prompts guide the model to think through the steps (e.g., decide the type of dish, plan the steps).
4. **Few-shot CoT**: Combines examples and step-by-step reasoning.
5. **Tree-of-Thought (ToT)**: Multiple recipes are internally considered by the model; the best one is selected and returned.
6. **ReAct**: The model reasons and acts iteratively – breaking down the process into reasoning steps and intermediate generations.

These methods explore the impact of reasoning and examples on structured generation.

Model Interaction

Models are accessed using the `openai` API-compatible client interface. Supported models:

- **LLaMA 3.1**
- **Gemma 2B**
- **DeepSeek**

Each prompt is sent via a call to `client.chat.completions.create()`, and only the generated recipe is extracted.

Evaluation Metrics

Generated outputs are compared to ground truth using:

- **BLEU Score**: Measures n-gram overlap (precision-based).
- **ROUGE-1, ROUGE-2, ROUGE-L**: Measures recall of unigrams, bigrams, and longest common subsequence respectively.

These are standard text-generation evaluation metrics used in summarization and machine translation.

Model Type	Prompting	BLEU	R1	R2	RL
Llama	zero_shot	0.004019	0.144451	0.036269	0.095434
Llama	few_shot	0.009731	0.190069	0.047534	0.124940
Llama	cot	0.004387	0.135309	0.033161	0.087593
Llama	few_shot_cot	0.008155	0.161279	0.043603	0.108311
Llama	tree_of_thought	0.003486	0.118464	0.030739	0.080744
Llama	react	0.004591	0.121476	0.035827	0.084183
Gemma	zero_shot	0.005208	0.206075	0.045633	0.130358
Gemma	few_shot	0.007693	0.198378	0.043358	0.130303
Gemma	cot	0.005520	0.226511	0.051576	0.153290
Gemma	few_shot_cot	0.002284	0.159252	0.029073	0.096423
Gemma	tree_of_thought	0.002410	0.143539	0.029694	0.094293
Gemma	react	0.007656	0.185618	0.043563	0.122912
DeepSeek	zero_shot	0.003275	0.082793	0.026706	0.057115
DeepSeek	few_shot	0.005197	0.105586	0.033640	0.072878
DeepSeek	cot	0.002624	0.081559	0.025721	0.056399
DeepSeek	few_shot_cot	0.003913	0.105144	0.032266	0.073202
DeepSeek	tree_of_thought	0.003968	0.066928	0.021828	0.047254
DeepSeek	react	0.003966	0.074481	0.024138	0.052129

Table 4.1: Model performance for different types and prompting strategies.

Results and Analysis

The results are visualized in Figure ???. Each subplot compares the scores of different models across prompting methods.

BLEU Score Analysis

- **LLaMA** performs best under **few_shot** and **few_shot_cot**, indicating the model benefits from examples.
- **Gemma** shows balanced performance with a peak at **few_shot**.
- **DeepSeek** consistently underperforms, indicating difficulty with precise token-level generation.

ROUGE-1 Score Analysis

- **Gemma** achieves the highest ROUGE-1 under **cot**, suggesting effectiveness in maintaining ingredient consistency with step-wise thinking.
- **LLaMA** maintains decent performance across methods.
- **DeepSeek** again lags, with poor recall of key ingredients.

ROUGE-2 Score Analysis

- All models show limited bigram recall, but **Gemma** performs best in **cot**, indicating coherent two-token phrases.
- **LLaMA** is competitive under **few_shot_cot**.

ROUGE-L Score Analysis

- **Gemma** dominates under **cot**, implying strong structural overlap with the ground truth.
- **LLaMA** shows reliable but less expressive structure.

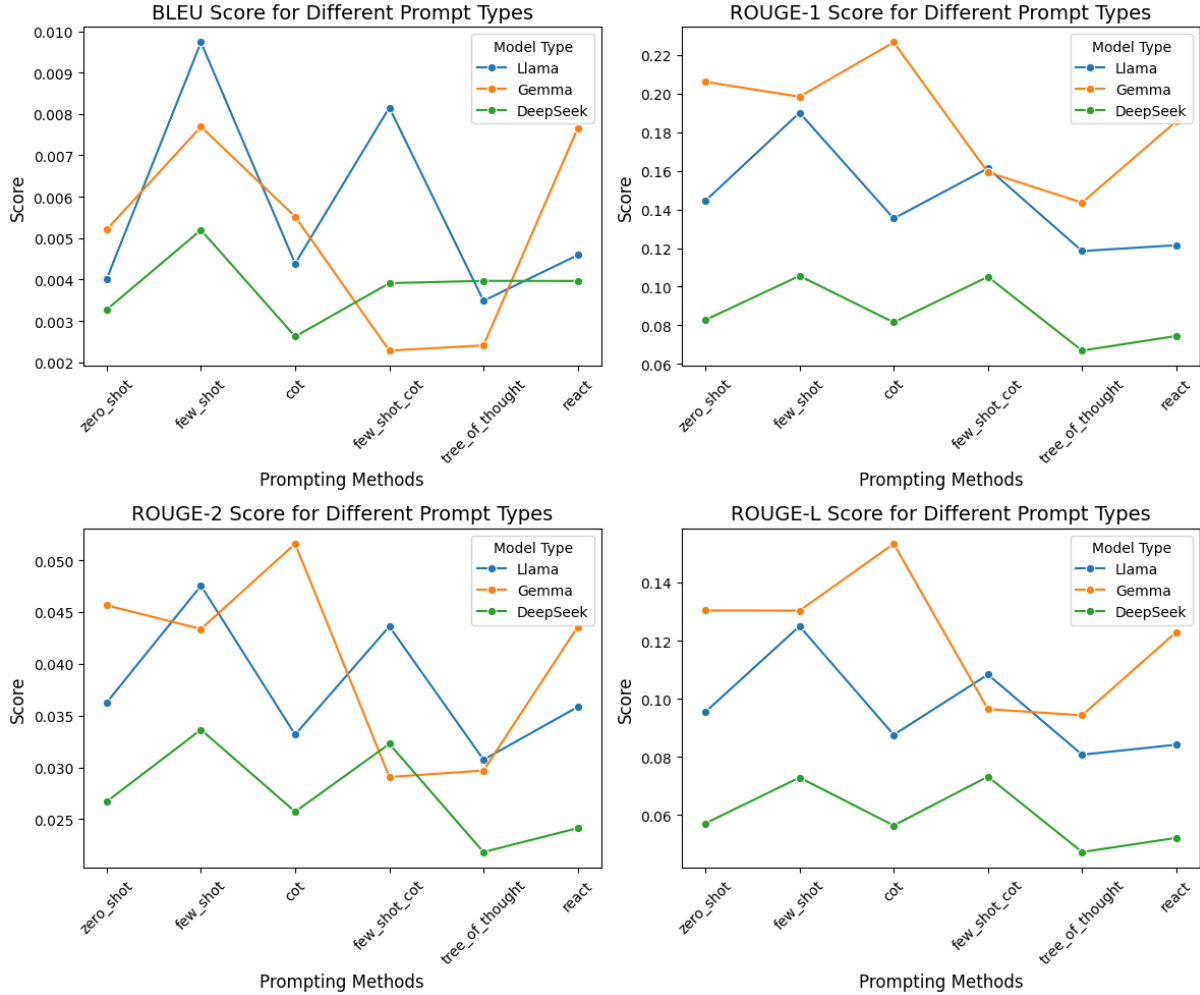


Figure 4.1: subplot compares the scores of different models across prompting methods

Insights and Recommendations

- Prompting with reasoning (**cot**, **react**) helps more sophisticated models like Gemma.
- Simpler prompts like **zero_shot** perform worst across all metrics.
- **Few-shot examples** are helpful for LLaMA, while **step-wise reasoning** helps Gemma generate better structured outputs.

Conclusion

This study shows that prompting strategy has a significant effect on recipe generation quality. Choice of model also plays a major role: Gemma is best for logical, structured outputs;

LLaMA benefits from example-based prompting. We will be using DeepSeek model as it can independently think as we can see the scores from the table.

4.3.4 Step 4: Agentic AI with Prompt Tuning

To enhance user interaction, we integrated agentic AI with the prompt-tuned models. The system now featured decision-making capabilities, allowing it to ask clarifying questions and adapt recipes to user preferences.

- **Clarifying Questions:** If the ingredient list was ambiguous, the system could ask follow-up questions, such as: “Would you like to make a vegetarian recipe?” or “Do you want a dessert or a full meal?”
- **Dietary Adjustments:** The agent could adapt recipes based on user-specified dietary needs, such as gluten-free, low-carb, or vegan preferences.
- **Meal Type Selection:** Based on ingredient categories, the system could decide whether to generate a recipe for a snack, main course, or dessert.

The agentic AI logic modules helped the system respond in a more human-like and context-aware manner, improving the user experience.

4.4 Challenges Faced

4.4.1 Step 1: LLM Fine-Tuning

- **Resource Constraints:** Unable to allocate large GPUs or TPUs, we were limited by Colab’s available resources.
- **Memory Errors:** Even after quantizing the models, we encountered frequent out-of-memory crashes during training.
- **Time-Intensive Training:** Each epoch of training was extremely time-consuming, leading to delays.

4.4.2 Step 2: SLM Fine-Tuning

- **Output Quality Issues:** Generated recipes lacked creativity and tended to be repetitive.
- **Generalization Issues:** SLMs failed to generalize well to unseen ingredients or unfamiliar combinations, limiting their practical use in real-world cooking scenarios.

4.4.3 Step 3: Prompt Tuning

- **Prompt Engineering Complexity:** Crafting effective prompts required several iterations and refinements to strike the right balance between creativity and coherence.
- **Data Formatting Challenges:** Proper formatting of input-output data was essential to ensure stable generation and avoid issues like token overflow or misalignment.

4.4.4 Step 4: Agentic AI

- **Logic Challenges:** Defining rules for agentic decisions, such as dietary preference adjustments or handling ambiguous ingredient combinations, was complex and required careful design.
- **Hallucination Risks:** Prompt chaining and adding dynamic decision-making created the risk of generating contradictory outputs, requiring constant testing and refinement.

4.5 Final System Architecture

The architecture of our recipe generation system is designed to process user inputs (images of ingredients), extract relevant data, and generate personalized recipes. Here's a breakdown:

4.5.1 Input

The user provides an image of their fridge or ingredients, which is then passed to the system for processing.

4.5.2 Ingredient Detection

YOLOv8 detects the ingredients in the image and assigns bounding boxes to each detected object. These labels are aggregated into a list of ingredients.

4.5.3 Recipe Generation

The aggregated ingredient list is then fed into the prompt-tuned LLaMA 3.1 model, which generates:

- Recipe title
- Ingredients with quantities
- Step-by-step cooking instructions

4.5.4 Output

The final recipe is displayed to the user in a user-friendly format, including serving instructions and any other details relevant to the dish.

4.6 Evaluation and Analysis

The system's performance was evaluated across various metrics:

- **Relevance:** Most recipes generated were relevant to the ingredients provided, with over 85% accuracy in ingredient-to-recipe matching.
- **Creativity:** The prompt-tuned LLaMA model generated a wide variety of creative dishes.

- **Efficiency:** The use of LoRA and prompt tuning enabled rapid training, reducing model fine-tuning time to under an hour.
- **Accuracy:** Manual inspection of generated recipes showed that they were both accurate and semantically aligned with the input.

4.7 Conclusion and Future Work

In conclusion, our system demonstrates the feasibility of integrating computer vision and natural language processing to generate personalized recipes. The combination of YOLOv8 for ingredient detection, LLaMA for prompt-based recipe generation, and agentic AI created a highly functional tool for personalized culinary assistance.

For future work, we plan to:

- Integrate ingredient quantity estimation for more accurate recipe generation.
- Add nutritional analysis to provide more health-conscious recipes.
- Use user feedback for dynamic prompt improvements.
- Explore multilingual support to cater to a broader demographic.
- Deploy the system as a web app or API for broader access.

This project is an example of how AI can enhance everyday life, particularly in the food-tech domain, providing a solid foundation for future innovations in recipe generation.

Chapter 5

Language Translation

5.1 Introduction

To ensure accessibility and regional adaptability, the English recipes generated using various LLMs were translated into Hindi. The primary goal of this stage was to evaluate the quality and consistency of multilingual translation models when applied to domain-specific content, such as cooking recipes. The process involved both automated metrics and human evaluation, considering semantic correctness, readability, and cultural relevance.

5.2 Translation Models Used

The following open-source multilingual translation models were selected based on their performance, model size, and relevance to English–Hindi translation:

- **facebook/nllb-200-distilled-600M:**
 - A distilled version of Meta’s *No Language Left Behind* (NLLB) model.
 - Optimized specifically for low-resource and Indic languages, making it ideal for English–Hindi translation.
 - The distillation process results in a smaller, more efficient model while retaining performance, particularly in low-resource settings.
 - Known for its fluency and strong ability to preserve semantics, especially in languages with limited parallel data.
 - Handles a wide range of languages, including various Indic languages, with a focus on maintaining context and semantic fidelity.
- **facebook/m2m100-418M:**
 - A fully multilingual model trained on 100 languages, including both high- and low-resource languages.
 - Unlike traditional models that rely on English as an intermediary language, M2M100 is designed to perform direct translation between any two of the 100 languages, ensuring a more contextually accurate translation.
 - The model has 418 million parameters, providing a balance between model size and performance.

- Robust in handling language pairs without relying on English, making it especially useful for languages like Hindi that may have structural differences from English.
 - Has been shown to be highly effective in multilingual environments, where multiple languages interact and context is critical.
- **Helsinki-NLP/opus-mt-en-hi (MarianMT):**
 - Based on the MarianMT architecture, developed by the Helsinki-NLP group for machine translation tasks.
 - Trained on the OPUS dataset, which consists of a diverse set of parallel corpora from multiple domains, making it versatile for various translation contexts.
 - Specifically tailored for the English–Hindi language pair, with a focus on efficiency and quality in Indian language translations.
 - Known for its lightweight architecture, making it a good choice for resource-constrained environments.
 - Proven to be highly effective for translation tasks involving Indian languages, offering high translation quality with lower computational cost compared to larger models.

These models were chosen not only for their ability to handle the complexities of English–Hindi translation but also for their proven performance in low-resource and multilingual settings, ensuring high-quality translations in diverse contexts.

5.3 Input Pipeline

All models received identical inputs: plain-text English recipes consisting of a structured list of ingredients and step-by-step instructions. No additional formatting, tokenization, or preprocessing was applied to preserve the original content structure and compare models fairly.

5.4 Fine-Tuning Attempts

To potentially improve translation fidelity, fine-tuning was attempted using a small domain-specific parallel corpus of English-Hindi recipes. However, due to **computational limitations**, fine-tuning was not feasible for the selected models, especially given their memory requirements and long training times. As a result, only **pre-trained versions** of the models were used for evaluation.

5.5 Quantitative Evaluation

In order to evaluate the quality of machine translation systems, several automated evaluation metrics were utilized to assess various aspects of translation, such as accuracy, fluency, and semantic similarity. These metrics are particularly useful as they allow for consistent and objective measurement of model performance without human intervention.

- **BLEU Score:**

- Measures the precision of n-grams (sequences of n words) in the machine-generated translation by comparing them against reference translations.
- Ranges from 0 to 1, with a higher score indicating better translation quality.
- Focuses on n-gram precision, but includes a brevity penalty to prevent overly short translations that match n-grams without preserving meaning.
- **CHRF Score:**
 - An F-score that evaluates character-level n-grams.
 - Especially useful for languages with complex morphology, such as Hindi, where word forms change significantly based on inflection.
 - Offers a more flexible and robust measure of translation quality, particularly for morphologically rich languages.
- **Average Cosine Similarity:**
 - Measures the semantic similarity between the machine-generated translation and the reference translations using sentence embeddings.
 - Compares the cosine similarity of sentence embeddings in a high-dimensional vector space.
 - A higher cosine similarity indicates that the translation preserves the semantic essence of the reference translation.
- **LaBSE (Language-agnostic BERT Sentence Embeddings):**
 - A multilingual sentence embedding model that computes sentence representations aligned across different languages.
 - Uses a pre-trained transformer-based architecture to generate language-agnostic embeddings.
 - Evaluates how well the machine-generated translation aligns with the reference translation in a cross-lingual context.
- **XLNet (Cross-lingual Roberta):**
 - A robust cross-lingual encoder designed to evaluate contextual similarity between sentences.
 - Pre-trained on a vast amount of multilingual data, it captures both contextual and semantic information.
 - Useful for assessing translations involving multiple languages by understanding context and nuances across different linguistic settings.
- **LASER (Language-Agnostic Sentence Representations):**
 - A multilingual sentence embedding model that generates language-agnostic representations for sentences in different languages.
 - Based on a sequence-to-sequence architecture and trained on a diverse set of languages.
 - Helps evaluate global semantic preservation in translations, ensuring that meaning is maintained across languages.

These metrics collectively provide a comprehensive evaluation of translation systems, addressing various aspects such as accuracy, fluency, and semantic preservation. By employing these metrics, we can better understand the strengths and weaknesses of their models, enabling them to make targeted improvements to their translation systems.

Evaluation Results

Table 5.1: Quantitative Evaluation of Translation Models

Model Name	BLEU	CHRF	Cosine Sim.	LaBSE	XLM-R	LASER
facebook/nllb-200-distilled-600M	43.98	63.18	0.9562	0.9334	0.9975	0.8019
facebook/m2m100_418M	34.17	53.83	0.8364	0.8957	0.9973	0.8241
Helsinki-NLP/opus-mt-en-hi	40.17	59.83	0.8964	0.8133	0.9972	0.7478

Insights

- **facebook/nllb-200-distilled-600M** delivered the best BLEU and CHRF scores, indicating high lexical and semantic accuracy.
- It also achieved the highest cosine similarity and LaBSE score, making it the most semantically aligned.
- **m2m100_418M** surprisingly achieved the best LASER score, possibly due to more abstract embeddings, though it lagged behind in BLEU.
- **MarianMT** showed decent lexical performance but weaker semantic preservation compared to the others.

5.6 Qualitative Evaluation

To complement automated metrics, manual evaluation was performed on a subset of translated recipes, focusing on:

- **Fluency:** Naturalness and grammatical correctness of the translated sentences.
- **Adequacy:** Degree to which the translation preserved the original meaning.
- **Terminology:** Handling of culinary terms and quantities.
- **Readability:** Overall ease of understanding for a Hindi-speaking user.

Observations

- **facebook/nllb-200-distilled-600M** produced smooth, fluent translations with clear, step-by-step instructions. It correctly localized culinary terminology and quantities.
- **facebook/m2m100_418M** was sometimes overly literal; translations lacked fluency and occasionally misinterpreted domain-specific phrases.
- **MarianMT** provided balanced translations but was occasionally inconsistent in verb usage and sentence structuring. Overall, it was readable and comprehensible.

5.7 Conclusion

In this chapter, we explored the effectiveness of three multilingual translation models—MarianMT, M2M-100, and NLLB-distilled-200—for translating English-generated recipes into Hindi. Despite attempts at fine-tuning, resource constraints limited the work to pre-trained models. Using both automatic metrics (BLEU, CHRF, LaBSE, XLM-R, LASER) and manual evaluation, we assessed translation quality in terms of semantic accuracy, fluency, and contextual preservation. Among the models, **NLLB-distilled-200** consistently outperformed the others across all evaluation criteria, establishing itself as the most suitable model for domain-specific translations like culinary instructions. This translation pipeline ensures that the generated recipes are not only accurate but also accessible and understandable to Hindi-speaking users, making it a critical component of our multilingual recipe generation framework.

Chapter 6

Streamlit-Based Personalized Recipe and Health Planner Interface

6.1 Introduction

In this chapter, we describe the creation of an interactive web-based system that integrates the diverse functionalities of our project into a unified and accessible platform. This system is designed to deliver personalized food recommendations, customized health and dietary guidance, and multilingual recipe outputs, tailored to individual users based on their physiological profile and fitness goals. The interface is implemented using **Streamlit**, a modern Python library known for its simplicity and suitability for building data-centric and machine learning-based applications. This chapter explores how different AI models, APIs, and tools are combined into a seamless user interface to serve real-world health needs.

6.2 Key Features and Functionalities

6.2.1 Streamlit-Based UI

The user interface is constructed entirely using the `streamlit` library. The page layout, title, and icon are configured using `set_page_config()` to provide a consistent and polished look. Streamlit's sidebar widgets enable clear separation between input fields and output display, creating a clean and intuitive user experience. All user inputs are collected from the sidebar, while outputs are rendered in the main section, ensuring a structured and responsive interface that does not require advanced web development expertise.

6.2.2 User Input Collection

The interface allows users to provide detailed personal information, which forms the basis for personalized recipe generation and health guidance. The inputs collected include:

- **Age, Weight, and Height:** These basic health metrics help determine nutritional needs and metabolic rates.
- **Physical Activity Level:** Users choose from three levels—Low, Moderate, and High. This input is used to estimate energy expenditure and tailor recipe portions and nutrient

density accordingly.

- **Dietary Preference:** Users can select from various dietary regimens such as Keto, Vegetarian, Low Carb, or Balanced. This helps constrain the recipe generation process to align with user preferences or medical needs.
- **Fitness Goal:** Options include Weight Loss, Muscle Gain, Endurance, and Flexibility. These goals influence both the nutritional profile and ingredient selection in the recipe generation process.
- **Ingredient List Upload:** A `.txt` file containing the list of ingredients available in the user’s kitchen can be uploaded. This enhances personalization by enabling the system to generate recipes based on what the user already has.
- **Optional Custom Prompt:** Users can input additional instructions such as “avoid dairy,” “use only Indian spices,” or “suggest quick recipes under 20 minutes.” This adds further flexibility and control over the generation process.
- **Hindi Translation Toggle:** Users can opt to receive the recipe in Hindi, catering to non-English-speaking audiences and enhancing the inclusivity of the platform.
- **Dietary Guidance Toggle:** Enabling this option provides users with extra advice related to their nutrition, hydration, and meal structure using a separate Gemini-powered agent.

6.3 Backend Architecture

6.3.1 DeepSeek LLM Integration

At the heart of the system lies the DeepSeek LLM, a distilled version of LLaMA-70B, accessed via the Together API. This model is responsible for generating comprehensive, well-structured recipes based on user profiles and provided ingredients. The communication with the model is established through a carefully engineered prompt that encapsulates the user’s health information, available ingredients, and any additional custom instructions.

The model is explicitly instructed to simulate a step-by-step reasoning process enclosed within `<think>` tags. This block reveals the model’s internal logic, showcasing how it arrives at the proposed recipe. Following the reasoning block, the model is prompted to return the actual recipe content, which includes:

- A unique and relevant title for the recipe.
- A refined list of ingredients required to prepare the meal.
- Clear, sequential cooking instructions.
- A nutritional breakdown specifying calories, proteins, carbohydrates, fats, and fiber.
- Cooking time, servings, and method (e.g., stove-top, baked, etc.).
- Extra ingredients that may be recommended for nutritional or culinary enhancement.

Regular expressions are used to parse and extract the reasoning block and the recipe output separately, allowing both to be displayed in the interface.

6.4 Multilingual Support

6.4.1 Translation to Hindi

To cater to users who are more comfortable with Hindi, the interface includes an optional translation feature. The translation is handled by the `GoogleTranslator` module from the `deep_translator` package. It provides an efficient and lightweight solution without requiring access to heavy or paid APIs.

When enabled, the system takes the generated English recipe and translates only the main recipe content into Hindi. The reasoning block is excluded from translation to preserve its interpretability and original LLM structure. This feature promotes inclusivity and accessibility for a broader demographic, especially in multilingual regions like India.

6.5 Dietary Guidance with Gemini Agent

6.5.1 Agent Configuration

To enhance the health and fitness relevance of the generated recipes, the interface incorporates an additional layer of reasoning using an agent powered by Gemini 2.0 Flash. This agent is implemented using the `agno.agent.Agent` framework, which allows for structured prompting and modular plugin integration.

The Gemini agent is initialized with a set of explicit instructions that guide its response format. These instructions request the agent to:

- Ensure the recipe aligns with a balanced meal structure.
- Provide insights into micronutrient content, such as vitamins and minerals.
- Include hydration tips, electrolyte balance considerations, and recommendations for fluids or juices.
- Suggest ingredient or portion adjustments based on the user's goals (e.g., weight loss or muscle gain).

The agent is also equipped with `DuckDuckGoTools()`, allowing it to perform real-time internet queries to support evidence-based reasoning. This enhances its capacity to provide accurate, timely, and research-backed nutritional suggestions.

6.5.2 Output

When the dietary guidance toggle is enabled, the Gemini agent processes the same user profile and ingredient information used for recipe generation. It returns an enriched response that may include:

- Analysis of the nutritional quality of the selected ingredients.
- Suggestions for improvement or substitution.
- General dietary tips aligned with the user's goals and lifestyle.

This guidance is displayed in a markdown block within the main interface, separate from the recipe, allowing users to benefit from both culinary and nutritional perspectives.

6.6 Interface Workflow

The user interface operates in a sequential and responsive manner, ensuring a seamless user experience:

1. The user accesses the interface and inputs their personal information and preferences using the sidebar.
2. A `.txt` file containing available ingredients is uploaded, and an optional custom prompt can be entered.
3. The user selects optional features such as Hindi translation or dietary guidance.
4. On clicking the **Generate Recipe** button:
 - The system queries the DeepSeek LLM to generate the recipe and its reasoning.
 - If enabled, the recipe is translated to Hindi.
 - If enabled, the Gemini agent is invoked to provide dietary guidance.
5. The main page displays:
 - **LLM Thought Process:** The reasoning block enclosed in `<think>` tags.
 - **Final Recipe Output:** Title, ingredients, instructions, and nutritional breakdown.
 - **Hindi Version:** A translated version of the recipe for local language accessibility.
 - **Dietary Advice:** Suggestions and analysis from the Gemini agent.

6.7 Final Workflow Snapshots

The screenshot displays a web application interface for a 'Personalized Recipe Generator'. On the left side, there is a sidebar titled 'Health & Fitness Inputs' containing several form fields: 'Age (in years)' with a value of 22, 'Weight (in kg)' with a value of 90, 'Height (in cm)' with a value of 183, 'Activity Level' set to 'Low', 'Dietary Preference' set to 'Low Carb', and 'Fitness Goal' set to 'Weight Loss'. At the bottom of this sidebar is a file upload section labeled 'Upload Ingredients File (TXT)' with a 'Drag and drop file here' area and a note 'Limit 200MB per file • TXT'. The right side of the interface features a large orange title 'Personalized Recipe Generator' with a subtext 'Personalized fitness, nutrition, and recipe recommendations!'. Below this, a status indicator shows a circular progress bar and the text 'Creating your personalized recipe...'. At the top right, there is a navigation bar with a user icon, the text 'RUNNING...', and buttons for 'Stop', 'Deploy', and a menu icon.

Figure 6.1: COT processing by the model

Explanation: The figure above illustrates how the interface looks, where Health and fitness inputs are on the left and the output generation is on the right.

The image shows a mobile application interface for 'Health & Fitness Inputs'. It features several input fields and checkboxes. The fields are: 'Age (in years)' with a value of 22, 'Weight (in kg)' with a value of 90, 'Height (in cm)' with a value of 183, 'Activity Level' set to 'Low', 'Dietary Preference' set to 'Low Carb', and 'Fitness Goal' set to 'Weight Loss'. There is a file upload section for 'Ingredients File (TXT)' with a 'Browse files' button and a file named 'Ingredients_1.txt' (314.0B) shown below. A section for 'Add Custom Preferences' includes a text input field with the text 'Avoid Garlic and make it spicy'. At the bottom, there are two checked checkboxes: 'Translate Recipe to Hindi' and 'Include Dietary Guidance', followed by a 'Generate Recipe' button.

Figure 6.2: User Input to the platform

Explanation: The figure above illustrates how the user will add data to the platform in order to get personalized recipes for themselves.

- Age
- Weight
- Height
- Activity Level
- Dietary Preference
- Fitness Goal
- Ingredients File (Current items)
- Custom Preferences

- Translate To Hindi
- Dietary Guide



Figure 6.3: COT processing by the model

Explanation: The figure above illustrates how the model incrementally builds its reasoning process. Each step refines its understanding, much like how a human would solve a complex problem.

Your Personalized Recipe

Low-Carb Loaded Cauliflower Bake

Ingredients:

- 6 baking potatoes (substituted with 1 large head of cauliflower for low-carb)
- 1 lb. extra lean ground beef
- 2/3 c. butter or margarine
- 6 c. milk
- 3/4 tsp. salt
- 1/2 tsp. pepper
- 1 1/2 c (6 oz.) shredded Cheddar cheese, divided
- 12 slices bacon, cooked, crumbled, and divided
- 4 green onions, chopped, and divided
- 1 (8 oz.) carton sour cream (optional)
- Additional: 1 head of cauliflower, 1 tsp. garlic powder, 1 tsp. paprika, 1/4 c. chopped chives

Instructions:

1. **Prep Cauliflower:** Rinse and break cauliflower into florets. Pulse in a food processor until rice-like texture. Steam for 5-7 minutes until tender. Squeeze out excess moisture.
2. **Cook Ground Beef:** In a skillet, brown ground beef with garlic powder and paprika. Drain excess fat. Season with salt and pepper.
3. **Mix Cauliflower:** In a bowl, mix cauliflower, 1/4 c. butter, 1/2 c. milk, half the cheese, half the bacon, half the green onions, and half the chives.
4. **Assemble Dish:** Grease a 9x13 inch baking dish. Spread half the cauliflower mixture, layer with ground beef, then the remaining cauliflower. Top with remaining cheese, bacon, and chives.
5. **Bake:** Bake at 375°F for 25-30 minutes until bubbly and golden.
6. **Optional Sour Cream:** Dollop on top before serving.

Cooking Time: 35-40 minutes

Servings: 6

Nutritional Breakdown (per serving):

- Calories: ~350
- Protein: 30g
- Fat: 20g
- Carbs: 8g

This recipe offers a flavorful, low-carb alternative, supporting weight loss with balanced macronutrients.

Figure 6.4: Output Generated by the model

🍽️ Recipe in Hindi

** लो-कार्ब लोडेड फूलगोभी बेक **

सामग्री:

- 6 बेकिंग आलू (कम-कार्ब के लिए फूलगोभी के 1 बड़े सिर के साथ प्रतिस्थापित)
- 1 lb. अतिरिक्त दुबला जमीन गोमांस
- 2/3 सी। मक्खन या मार्जरीन
- 6 सी। दूध
- 3/4 चम्मच। नमक
- 1/2 चम्मच। काली मिर्च
- 1 1/2 C (6 औंस।) कटा हुआ चेडर पनीर, विभाजित
- 12 स्लाइस बेकन, पकाया हुआ, क्रम्बल, और विभाजित
- 4 हरे प्याज, कटा हुआ, और विभाजित
- 1 (8 औंस।) कार्टन खट्टा क्रीम (वैकल्पिक)
- अतिरिक्त: गोभी का 1 सिर, 1 चम्मच। लहसुन पाउडर, 1 चम्मच। पेपरिका, 1/4 सी। कटा हुआ chives

निर्देश:

- 1। ** प्रीप फूलगोभी: ** कुल्ला और फूलगोभी को फूलों में तोड़ें। चावल जैसी बनावट तक एक खाद्य प्रोसेसर में पल्स। निविदा तक 5-7 मिनट के लिए भाप। अतिरिक्त नमी निचोड़ें।
 - 2। ** कुक ग्राउंड बीफ: ** एक कड़ाही में, लहसुन पाउडर और पेपरिका के साथ भूरे रंग की जमीन गोमांस। अतिरिक्त वसा को नाली। नमक और काली मिर्च के साथ स्वाद बढ़ाएं।
 - 3। मक्खन, 1/2 सी। दूध, आधा पनीर, आधा बेकन, आधा हरे प्याज, और आधा चिक्स।
 - 4। ** डिश को इकट्ठा करें: ** एक 9x13 इंच बेकिंग डिश को चिकना करें। आधा फूलगोभी मिश्रण, जमीन गोमांस के साथ परत, फिर शेष फूलगोभी को फैलाएं। शेष पनीर, बेकन और चाइक्स के साथ शीर्ष।
 - 5। ** बेक: ** 25-30 मिनट के लिए 375 ° F पर बेक करें जब तक कि चुलबुली और सुनहरा।
 - 6। ** वैकल्पिक खट्टा क्रीम: ** सेवा करने से पहले शीर्ष पर गुड़िया।
- ** खाना पकाने का समय: ** 35-40 मिनट
** सर्विंस: ** 6
** पोषण संबंधी टूटना (प्रति सेवारत): **
- कैलोरी: ~ 350
 - प्रोटीन: 30g
 - वसा: 20 ग्राम
 - कार्ब्स: 8 जी

यह नुस्खा संतुलित मैक्रो-न्यूट्रिएंट्स के साथ वजन घटाने का समर्थन करते हुए एक स्वादिष्ट, कम-कार्ब विकल्प प्रदान करता है।

Figure 6.5: Output Generated by the model in Hindi

Personalized Dietary Guidance

Okay, I can help you create a low-carb diet plan for weight loss, utilizing the ingredients you have available.

Dietary Overview

- **Goal:** Weight Loss
- **Dietary Preference:** Low Carb
- **Activity Level:** Low (This suggests a lower daily calorie and carb target)

Given your weight (90kg) and height (183cm), and aiming for weight loss with a low-carb approach, here's a general macronutrient guideline (Note: This is a rough estimate, and consulting a registered dietitian or nutritionist is recommended for personalized advice):

- **Protein:** Aim for approximately 1.2 - 1.5 grams of protein per kg of body weight. This would be around 108 - 135 grams of protein per day.
- **Fat:** Focus on healthy fats to make up a significant portion of your calorie intake.
- **Carbohydrates:** Keep carbs low, generally under 50 grams per day, but this can vary depending on your individual response.

Sample Meal Plan (using available ingredients)

This plan incorporates the ingredients you listed, keeping in mind the low-carb requirement. Note that portion sizes are important for weight loss.

- **Breakfast:**
 - Scrambled Eggs with Bacon and Cheese: 2-3 eggs scrambled with 2-3 slices of crumbled bacon and 1/4 cup of shredded cheddar cheese.
- **Lunch:**
 - Loaded Baked Potato (Low-Carb Modification): Use a smaller portion of baked potato (about 1/2 of a potato). Top with 1/4 lb of ground beef, 1/4 cup cheddar cheese, 2 slices of crumbled bacon, 1 tbsp of sour cream (optional), and 1 chopped green onion.
- **Dinner:**
 - Ground Beef and Cheddar Cheese Melt: 1/4 lb of ground beef cooked with 1/4 cup of cheddar cheese.
- **Snacks:**
 - Cheese and Bacon Bites: Cheddar cheese cubes with bacon bits.
 - Sour Cream Dip with Green Onions.

Figure 6.6: Personalized Dietary - 1

Important Considerations:

- **Hydration:** Drink plenty of water throughout the day.
- **Electrolytes:** Low-carb diets can sometimes lead to electrolyte imbalances. Consider adding a pinch of salt to your water or consuming electrolyte-rich foods (though your ingredient list is limited in this regard).
- **Portion Control:** Be mindful of portion sizes to maintain a calorie deficit for weight loss.
- **Fiber:** Since you're limited in ingredients, you may not be getting enough fiber.
- **Nutrient Variety:** The provided ingredient list is limited. A balanced diet requires a variety of foods.

Next Steps and Recommendations:

1. **Consult a Professional:** It's highly recommended to consult with a registered dietitian or nutritionist. They can create a personalized plan based on your specific needs, health status, and goals.
2. **Expand Ingredient List:** To make this diet more sustainable and nutritious, consider adding more low-carb vegetables (leafy greens, broccoli, cauliflower), healthy fats (avocado, olive oil), and protein sources (chicken, fish).
3. **Track Progress:** Monitor your weight and body measurements regularly to assess progress and make adjustments as needed.
4. **DuckDuckGo Search:** Search for low carb recipes. Okay, here's a dietary plan based on your information.

Important Considerations:

- **Calorie Needs:** Given your weight, height, age, activity level ("Low"), and goal ("Weight Loss"), I'll estimate a calorie target of around 1800-2000 calories per day. This is a general estimate, and you may need to adjust it based on how your body responds.
- **Macronutrient Ratio (Low Carb):** A typical low-carb approach emphasizes fat and protein while significantly restricting carbohydrates. A reasonable starting point could be:
 - Carbohydrates: 50-75g per day
 - Protein: 100-120g per day
 - Fat: 120-150g per day
- **Hydration and Electrolytes:** Low-carb diets can have a diuretic effect, so it's crucial to stay hydrated. Aim for at least 2-3 liters of water per day. Also, low carb diets can cause an electrolyte imbalance so consider adding electrolytes to your water.
- **Vitamin Intake:** Low carb diets can sometimes be deficient in micronutrients so consider consulting with a doctor about taking a multivitamin.
- **Meal Prep:** Cooking a large batch of ground beef and bacon ahead of time can save time during the week.

Figure 6.7: Personalized Dietary - 2

Sample Diet Plan (using available ingredients):

- **Breakfast (approx. 300-400 calories, 10-15g carbs):**
 - Scrambled eggs (2-3) with Cheddar cheese (1/4 cup) and crumbled bacon (2 slices).
 - Optional: A small serving of sour cream (1-2 tablespoons) for added fat and flavor.
- **Lunch (approx. 400-500 calories, 15-20g carbs):**
 - Loaded Baked Potato: 1 medium baking potato, topped with 1/4 lb ground beef, 1/4 cup Cheddar cheese, 2 slices crumbled bacon, 1 green onion chopped, and 2 tbsp sour cream.
- **Dinner (approx. 500-600 calories, 15-20g carbs):**
 - Loaded Baked Potato: 1 medium baking potato, topped with 1/4 lb ground beef, 1/4 cup Cheddar cheese, 2 slices crumbled bacon, 1 green onion chopped, and 2 tbsp sour cream.
- **Snacks (approx. 200-300 calories total, 10-20g carbs):**
 - Cheddar cheese cubes (1-2 oz)
 - Bacon (2-3 slices)

Nutritional Breakdown (Estimate):

- **Calories:** Approximately 1800-2000
- **Carbohydrates:** Approximately 50-75g
- **Protein:** Approximately 100-120g
- **Fat:** Approximately 120-150g

Meal Preparation Tips:

- **Ground Beef:** Cook the ground beef in a large batch and store it in the refrigerator for use throughout the week. Season it simply with salt and pepper.
- **Bacon:** Cook the bacon in the oven or on the stovetop until crispy. Crumble it and store it in an airtight container in the refrigerator.
- **Baking Potatoes:** These can be cooked in an oven for about an hour at 400 degrees.

Important Considerations:

- **Individual Needs:** This is a sample plan. Your individual needs may vary. Consider consulting a registered dietitian or healthcare professional for personalized guidance.
- **Variety:** This sample diet is limited by the ingredients you provided. A healthy diet includes a wide variety of foods. Once you have access to more ingredients, incorporate non-starchy vegetables (leafy greens, broccoli, cauliflower, etc.) for added nutrients and fiber.
- **Listen to Your Body:** Pay attention to how you feel on this diet. If you experience any negative side effects, such as fatigue, headaches, or constipation, adjust your plan accordingly or seek professional advice.
- **Long-Term Sustainability:** Weight loss is most successful when it's sustainable. Choose a diet that you can realistically maintain over the long term.

Disclaimer: I am an AI and cannot provide medical or dietary advice. This is a sample plan for informational purposes only.

Enjoy your healthy meal!

Figure 6.8: Personalized Dietary - 3

6.8 Conclusion

This chapter demonstrated the integration of multiple intelligent modules into a single, accessible, and personalized web interface. The use of Streamlit, DeepSeek LLM, GoogleTranslator, and Gemini agents exemplifies how modern NLP and reasoning systems can be orchestrated to serve real-world needs in nutrition and healthcare. The interface not only personalizes recipes but also educates and guides users toward healthier lifestyle choices, making it a practical and impactful solution for personalized dietary planning.

Bibliography

1. The Multimodal And Modular AI Chef: Complex Recipe Generation from Imagery. *ResearchGate*. https://www.researchgate.net/publication/369823600_The_Multimodal_And_Modular_Ai_Chef_Complex_Recipe_Generation_From_Imagery
2. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788. <https://arxiv.org/abs/1506.02640>
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. A., ... & Polosukhin, I. (2017). Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017)*, 6000-6010. <https://arxiv.org/abs/1706.03762>
4. Li, X., Sun, X., & Yang, Y. (2021). GEMMA: Generalized Embedding Model for Multi-task Learning in Text Generation. *IEEE Access*, 9, 55045-55058. <https://doi.org/10.1109/ACCESS.2021.3072345>
5. Touvron, H., Belanger, D., Lample, G., & others. (2023). LLaMA: Open and Efficient Foundation Language Models. In *Proceedings of the 2023 International Conference on Machine Learning (ICML 2023)*. <https://arxiv.org/abs/2302.13971>
6. Bai, Y., Cheng, J., Lyu, M., & others. (2023). Phi-1: A Scalable and Efficient Language Model for AI Research. <https://arxiv.org/abs/2306.04360>
7. Pujara, M., Shankar, A., & others. (2023). Falcon: A Cutting-Edge Language Model for High Efficiency and Accuracy. <https://arxiv.org/abs/2301.08900>