Culinary Curator: An AI-Powered Recipe Assistant

Ved Naik
Department of Computer Engineering
Mukesh Patel School of Technology Management and Engineering,
NMIMS University
Mumbai, Maharashtra, India
ved.naik109@nmims.edu.in

Aanya Lari
Department of Computer Engineering
Mukesh Patel School of Technology Management and Engineering,
NMIMS University
Mumbai, Maharashtra, India
aanya.lari060@nmims.edu.in

Khushi Tejwani
Department of Computer Engineering
Mukesh Patel School of Technology Management and Engineering,
NMIMS University
Mumbai, Maharashtra, India
khushi.tejwani178@nmims.edu.in

Ishani Saha
Department of Computer Engineering
Mukesh Patel School of Technology Management and Engineering,
NMIMS University
Mumbai, Maharashtra, India

***Abstract*—In today's data-driven world, the intersection of artificial intelligence, natural language processing (NLP), and computer vision has given rise to innovative solutions in various domains. One intriguing application area is recipe generation, we introduce the "Culinary Curator," a novel culinary assistant that utilizes AI, NLP, and computer vision to simplify meal planning and promote healthier eating choices.**

Our study presents a comparative analysis of five distinct models for recipe recommendation and two models for ingredient recognition. Through rigorous evaluation and performance metrics, we offer insights into the effectiveness of these models. Our findings contribute to the selection of optimal models for enhancing culinary experiences. This research informs the development of AI-powered recipe systems and underscores the significance of model choice in achieving accuracy and efficiency.

Keywords—NLP, AI, Computer Vision, recipe recommendation, ingredient recognition, GPT, recipe generation

I. INTRODUCTION

In the contemporary data-driven landscape, the amalgamation of artificial intelligence (AI), natural language processing (NLP), and computer vision has catalyzed a wave of innovative solutions spanning a multitude of domains. Among these emerging applications, one particularly intriguing area is recipe generation. In response, we present the "Culinary Curator," a pioneering culinary assistant that harnesses the power of AI, NLP, and computer vision to streamline meal planning and encourage healthier dietary choices.

The act of home cooking has been a cornerstone of daily life, fostering cultural richness and reinforcing familial bonds. However, amid the rapid urbanization and increasing reliance on technology, the everyday question of what to prepare from the ingredients at hand has become a formidable challenge. The "Culinary Curator" has been conceived as a technological marvel to offer guidance to individuals seeking culinary inspiration, especially when confronted with the perennial dilemma of what to cook given the ingredients available. Moreover, it contributes to the realm of informed dietary choices by providing detailed nutritional analyses of recipes.

This study embarks on a comprehensive journey, marked by a comparative analysis of five distinct models designed for recipe recommendation and two models dedicated to ingredient recognition. The models in the recipe recommendation category include the BERT Model using Sentence Transformer, Word2Vec and TF-IDF, GENSIM (FastText), LSTM, and Logistic Regression. On the ingredient recognition front, we evaluated the MobileNetV2 and a custom Convolutional Neural Network (CNN) model. We have meticulously evaluated these models, dissecting their performance across a range of parameters. The objective is to gain insights into the strengths and weaknesses of each model, ultimately determining the most effective options for enhancing the culinary experience.

The findings of our study extend beyond mere comparative metrics. They offer an informed perspective on why certain models outperformed others, thereby serving as the basis for our final selection. As we navigate through the research, we emphasize not only the guidance it provides for developing AI-powered recipe systems but also the pivotal role of model selection in achieving accuracy, efficiency, and the delivery of a seamless culinary experience.

In the subsequent sections, we will delve into the intricate details of each model, the evaluation metrics, the performance comparisons, and the rationale for selecting the optimal models. Through this research, we aim to underscore the significance of employing the right models in crafting a culinary assistant that thrives in the contemporary era.

A. Motivation

Home cooking remains integral to daily life, fostering cultural connections and supporting family bonds. However, the fast-paced, technology-driven lifestyle of the present day has complicated a seemingly simple question: what to cook with available ingredients. This perpetual culinary challenge, influenced by individual preferences and ingredient variability, has significant implications for dietary choices and meal planning.

Recipe recommendation systems have emerged as a promising solution to simplify meal planning, offering personalized cooking suggestions by considering factors like dietary restrictions, cooking skills, time constraints, and personal preferences. Yet, most of these systems fall short in accounting for a crucial detail – the ingredients readily available to the user. Ingredient accessibility is influenced by factors such as personal inventory, local availability, and dietary requirements, rendering a realistic context to culinary decision-making. This real-world constraint is often overlooked by conventional recipe recommendation systems.

Indian cuisine, characterized by its diverse regional recipes and ingredients, is renowned for its richness and complexity. While existing recommendation systems cover a wide range of cuisines, they frequently lack a focus on Indian culinary traditions. The project, Culinary Curator, seeks to address this gap by providing a specialized experience tailored to users interested in exploring the rich tapestry of Indian cooking. This specialization extends beyond generic recipe recommendations to include ingredients and techniques unique to Indian cuisine.

Modern dietary awareness and health-conscious cooking practices have become paramount in contemporary society. Key to this awareness is understanding the nutritional content of the food we consume. Unfortunately, many existing recipe recommendation apps lack comprehensive nutritional analysis features, leaving users to manually calculate or estimate nutritional values, a time-consuming and error-prone process. Our research aims to bridge this gap by integrating accurate and detailed nutritional analysis into the recipe recommendation process. This empowers users to make informed dietary choices in alignment with their culinary preferences.

A distinctive aspect of this research project is its comparative approach. Rather than focusing solely on implementing a single model for ingredient identification and recipe recommendation, the project engages in a comprehensive comparative analysis of multiple models. By evaluating and contrasting these models, we aim to uncover their individual strengths and weaknesses. The findings from this comparative analysis provide valuable insights for the scientific community and culinary practitioners, guiding the selection of the most effective models for specific culinary tasks, whether recipe recommendation or ingredient recognition. This approach fosters the advancement of the discipline and steers future efforts in the right direction.

Through these multifaceted research endeavors, the Culinary Curator seeks to offer a comprehensive and practical approach to recipe recommendation, with a specific emphasis on Indian cuisine. Our goal is to empower users who wish to make the most of the ingredients they have on hand while exploring the diverse world of Indian recipes. in the text.

II. Related Work

A. Overview

Artificial intelligence (AI), natural language processing (NLP), as well as computer vision, are bringing about a revolution in the world of food and culinary by introducing innovative applications that cater to diverse aspects of the culinary domain. These advancements offer solutions to help users discover new recipes, cook healthier meals, and minimize food waste while transforming the way we approach cooking and meal planning.

One of the most popular applications of AI in the culinary domain is recipe recommendation systems. AI-powered recipe recommendation systems have gained prominence, offering personalized cooking suggestions to users. These systems recommend recipes to users based on their dietary restrictions, cooking skills, time constraints, and preferences using machine learning algorithms. For example, the recipe website Yummly uses AI to recommend recipes to users based on their past browsing history, saved recipes, preferred cuisines, allergens, etc.

Furthermore, deep learning approaches can learn intricate patterns of user-item interaction through nonlinear transformations, they are superior in recommender systems. They are very good at representation learning, which eliminates the need for feature engineering by hand. Deep learning algorithms are perfect for problems involving temporal dynamics in user behavior and item evolution since they can successfully simulate sequential data. Also, these methods are more flexible, allowing for the creation of hybrid recommendation models that are able to adjust to a variety of traits and variables [2].

For a recipe to be recommended, ingredients must be extracted from the recipe text. In order to enable computers to match recipes with accessible components, advanced NLP techniques are used to recognise and extract elements and quantities from recipes. Given the variety of ways components are mentioned in recipes, this process can be difficult to parse, tokenize, and recognise entities [3].

Another promising application of computer vision techniques for ingredient recognition. These systems can identify the ingredients in images and videos. This can be used to recommend recipes based on ingredients identified from the image [4] or identify the ingredients used to make a dish from the image of the dish [5]. For example, the app Fooducate to identify the ingredients in food images and provide users with nutritional information.

AI and NLP can also be used to generate recipes from natural language descriptions [6]. This can be used to help users create custom recipes that meet their specific needs and preferences. For example, the website RecipeAI generates recipes based on users' dietary restrictions, cooking skills, and ingredient preferences.

AI can also be used to develop virtual chef assistants. These AI-powered systems can help users with a variety of culinary tasks, such as meal planning, cooking, and shopping. For example, the Amazon Alexa virtual assistant can be used to find recipes, generate shopping lists, and control kitchen appliances.

In addition to these specific applications, AI and NLP are also being used to improve food safety and develop more sustainable food systems. For example, AI systems are being used to track and identify potential food hazards, perform nutrition analysis and eating behaviour analysis [7] and to develop new ways to reduce food waste [8].

By developing innovative applications, these technologies can help people to discover new and exciting recipes, cook healthier meals, and reduce food waste.

B. Previous Research

This review of the literature dives into the field of AI-driven recipe recommendation systems, examining the developments, approaches, and discoveries that help the culinary industry come up with creative solutions. To obtain a thorough grasp of the state-of-the-art in this fascinating topic, we will examine the essential elements, methods, and trends in AI-based recipe recommendations in this survey.

The method used in one study employed a collaborative filtering to make recipe recommendations according to user preferences [1]. The item-based and user-based techniques are examined. The item-based method uses measures like the Tanimoto Coefficient and Log-Likelihood to evaluate recipe similarity without taking user ratings into account. The user-based approach, on the other hand, experiments with various neighbourhood sizes and thresholds and assesses user similarity using metrics like Pearson Correlation and Euclidean Distance. Evaluation metrics include Root Mean Squared Error (RMSE) and Average Absolute Difference (AAD). The results indicate that user-based collaborative filtering yields more accurate recipe recommendations than item-based techniques when there is enough data on user interactions.

Two primary components of another approach used are the identification of food ingredients and the suggestion of recipes. Using a Flask-based web application, food ingredient recognition uses the ResNet-50 CNN model to recognise ingredients from user-uploaded photos. Users can receive customised recipe recommendations by selecting their prefered diet and diet type, and the Edamam API and JavaScript are used to generate the recommendations. A user-friendly platform for recipe recommendations based on ingredients is produced using this methodology [4].

A cutting-edge recommendation system for sophisticated recipe recommendations is called the Hierarchical Graph Attention Network (HGAT) [9]. It is made up of several essential parts that allow HGAT to manage intricate interactions between users, recipes, and ingredients. Type-specific transformations guarantee data type compatibility, whereas node-level attention creates relation-specific embeddings by encoding neighbour nodes' information. These embeddings are combined by relation-level attention to create thorough node representations. Several layers of these representations are combined to improve the final node embedding. Recommendation scores are produced by a score predictor using user and recipe embeddings. HGAT is a valuable addition to recommendation systems since it is trained with a ranking-based objective function.

Recipe generation is an additional facet of recipe recommendation. Using a sizable corpus of recipes, RecipeMC refines the GPT-2 language model with an emphasis on producing coherent and structured recipes [10]. NAME $\rightarrow$ INGR (recipe name to ingredients) and NAME+INGR $\rightarrow$ INST (recipe name and ingredients to instructions) are the two activities that make up the text creation process. Reward functions direct the text generating process by promoting consistent outputs, preventing redundancy, and following predetermined recipe structures. During text generation, Monte Carlo Tree Search (MCTS) is employed to explore and choose the next token. The efficacy of this approach in generating recipes that satisfy particular standards and address prevalent constraints in language models is assessed.

An article highlights how important it is to comprehend the model's ability to associate food visuals with their corresponding elements. Because of the intricacies of zero-shot retrieval and visual changes, content-based ingredient recognition is difficult [11]. By using deep architectures to simultaneously learn food categorization and ingredient recognition, this research tackles these problems. We then use the learned semantic labels and deep features to novel zero-shot recipe retrieval. This paper illustrates the unique zero-shot challenge in cooking recipe retrieval and demonstrates the viability of ingredient recognition through thorough testing on a large dataset of complex photographs of Chinese food.

Another research also delivers matching recipes and nutritional information based on real-time food photographs or image links that are entered. Rather than requiring embeddings, the methodology streamlines the procedure by using a Convolutional Neural Network (CNN) to produce image encodings for recipe retrieval. The authors included an API for retrieving dietary data. The system uses a K-nearest neighbor classifier (KNN) and DenseNet-121 for image processing to get the closest matching recipe for the user-provided food image [12].

The technique with Convolutional Neural Networks (CNNs) for deep multi-ingredient recognition is the main emphasis of another study [13]. It starts with contrasting two foundational architectures that were first intended for object recognition: InceptionV3 and ResNet50. These architectures are adapted for multi-label ingredient classification after being pre-trained on ILSVRC data. Several highly active outputs are possible by applying a sigmoid activation in place of the conventional softmax activation. By using this method, the issue is changed into a multi-label framework where each ingredient's binary output is predicted by the model. The binary cross-entropy loss function is used to optimise the model, encouraging it to match the target binary vectors—which show whether each ingredient is present in the input samples or not—with its predictions.

Another study explores the use of Factorization Machines (FMs) for context-aware recipe suggestion [14]. It uses techniques like Stochastic Gradient Descent, Alternating Least-Squares, and Markov Chain Monte Carlo for parameter estimation and integrates a variety of contextual information from food recipe datasets. The absence of regularisation parameters and learning rate makes MCMC unique. The datasets go through preprocessing and filtration. The goal of the experiments is to improve recommendation accuracy, particularly with sparse datasets. Model performance is evaluated by cross-validation using RMSE and MAE metrics, providing a thorough method for context-aware recipe suggestion for sparse food domain data.

Another approach uses convolutional neural networks (CNNs) to achieve deep multi-ingredient recognition [15]. The first section compares two basic architectures that were originally intended for object recognition: InceptionV3 and ResNet50. After pre-training using ILSVRC data, these designs are adjusted for the purpose of multi-label ingredient classification. To provide several highly active outputs, a sigmoid activation is utilised in place of the normal softmax activation. With this method, the issue is changed to a multi-label framework in which the model forecasts a binary result for every component. By using the binary cross-entropy loss function, which encourages the model to align its predictions with the target binary vectors—which show whether each ingredient is present in the input samples or not—the model is optimized.

III. Methodology

A. *Data Collection and Description*

1) Ingredient Recognition:

The first step in ingredient recognition is to collect the necessary data. The dataset used for this research is ‘Fruits and Vegetables Image Recognition Dataset’, which is available on Kaggle.

We have used a diverse collection of food-related images, including a variety of fruits and vegetables. The dataset encompasses a comprehensive selection of food items:

Fruits: banana, apple, pear, grapes, orange, kiwi, watermelon, pomegranate, pineapple, mango.

Vegetables: cucumber, carrot, capsicum, onion, potato, lemon, tomato, raddish, beetroot, cabbage, lettuce, spinach, soy bean, cauliflower, bell pepper, chilli pepper, turnip, corn, sweetcorn, sweet potato, paprika, jalepeño, ginger, garlic, peas, eggplant.

This rich dataset is organized into three main folders: "train," "test," and "validation," each containing a set of 100, 10, and 10 images, respectively, for different fruits and vegetables. These subfolders within the main categories host the images specific to each food item, making it a valuable resource for food recognition and classification tasks.

2) Recipe Recommendation:

The first step in preparing the data is collecting the necessary data. The dataset used for this research is the ‘6000+ Indian Food Recipes Dataset’, which is available on Kaggle. The dataset consists of 5939 rows and 15 columns.

B. *Data Cleaning and Preprocessing:*

1) Ingredient Recognition:

Overall, data preprocessing and cleaning are crucial to ensure that the dataset is in the right format and that it's ready for training a deep learning model. These steps helped to improve the model's performance and make the training process smoother and more efficient.

a) Image Resize:

Resized the images to a consistent size. In this code, the images are resized to (224, 224, 3) using target_size=(224, 224) when loading images with image.load_img.

b) Image Normalization:

Normalized the pixel values of the images to a range between 0 and 1 by dividing by 255. This was done with img / 255.0.

c) Data Augmentation:

Data augmentation is a technique to increase the diversity of the training dataset by applying random transformations to the images. The ImageDataGenerator was used to apply various augmentations, including rotation, width and height shifting, shearing, zooming, and horizontal flipping.

d) Class Label Encoding:

Ensured that class labels are properly encoded. In this code, class_mode='categorical' was used, which implies that the classes are one-hot encoded. While doing this, we made sure that the subdirectories in the dataset directory structure match the class labels.

e) Train-Validation Split:

We had a separate validation dataset, so we could split it from the training dataset.

f) Missing class_names:

The code attempts to map the class index to class names using class_names[class_index]. To use this, we defined a list of class names corresponding to the classes in our dataset.

g) Error Handling and Exception Handling:

It's important to include error handling and exception handling in the code to handle various issues that may arise during data preprocessing and loading. For example, to want to handle cases where the image file doesn't exist or where there's an issue with the dataset directory structure.

2) Recipe Recommendation:

Effective data cleaning and preprocessing is essential for ensuring the compatibility of the data, especially the ingredients, with the models/algorithms used. The following are the steps involved in preprocessing:

a) Removing/replacing Parenthesis:

Regular expressions (regex) was used to remove any text within parentheses (e.g., "(extra cheese)") in the "ingredients" column. Further, the opening and closing parentheses containing empty strings are replaced, effectively removing them.

b) Remove non-alphabet characters:

Words that contain non-alphabetic characters are filtered out, ensuring that only alphabetic words remain.

c) Remove measuring words:

Measuring words are eliminated and phrases (e.g., "teaspoon," "tablespoon") by filtering out words that are present in list of measuring words defined before.

d) Remove stop words:

Common English stop words are removed (e.g., "the," "and") using NLTK's corpus of stop words.

e) Remove accents:

A function is used to remove accents from characters. For example, it converts accented characters to their non-accented equivalents.

f) Remove extra spaces:

Extra spaces in the text were removed to ensure consistency and readability.

g) Eliminate punctuations:

Punctuation marks were removed to standardize the text.

h) Convert to lowercase:

All text was converted to lowercase to ensure case-sensitive analysis.

i) Tokenization:

It is used in Natural Language Processing to split paragraph or sentence into individual units or tokens, making it easier to work with and analyse [16]. Ingredient descriptions are split into tokens by hyphens or spaces. For example, if the ingredient description is “tomato-sauce”, the tokenization process separates it into two tokens: “tomato” and “sauce”.

j) Lemmatization:

It is also used in Natural Language Processing to break a word down to its root form [17]. It standardizes variations of word to their common root form, enabling comparison and processing of ingredients. For example, for the word "tomatoes", lemmatization reduces it to its base form "tomato".

C. Model Implementation for Ingredient Recognition

1) MobileNetV2:

MobileNetV2 stands as a notable neural network architecture meticulously crafted for the efficient execution of computer vision tasks, especially on resource-constrained devices. Its architectural prowess encompasses several key components: It is known for its efficiency, acts as a feature extractor.
It processes input images of fruits and vegetables and extracts meaningful features from them. By utilizing depthwise separable convolutions and linear bottlenecks, it efficiently captures essential information from the images while keeping the computational requirements in check.

The model used in the code is pre-trained on a vast dataset. This pre-training imparts it with a robust understanding of general visual features. In the context of fruits and vegetable recognition, these general features serve as an excellent starting point. Fine-tuning the model on a more specific dataset (in this case, fruits and vegetables) allows it to adapt and specialize in recognizing these particular items.

Despite its efficiency, MobileNetV2 is known for maintaining a high level of accuracy. In the context of this code, it ensures that the recognition of fruits and vegetables is not compromised, and the model can make precise predictions.

Model Architecture: It defines the number of classes, which is 36 in this case, representing different fruits and vegetables. It creates an instance of the MobileNetV2 model with pretrained weights. MobileNetV2 is a popular CNN architecture known for its efficiency and accuracy. The layers of the pretrained model are frozen (not trainable) to retain the learned features. A custom classifier is added on top of the base model. This custom classifier includes:

a. A global average pooling layer to reduce spatial dimensions.
b. A dense layer with 1024 units and ReLU activation.
c. A dropout layer with a 50% dropout rate to prevent overfitting.
d. A dense output layer with a softmax activation function to predict class probabilities.

The model is compiled using the Adam optimizer and categorical cross-entropy loss. The "accuracy" metric is also monitored during training.
Data augmentation is applied to the training dataset using an ImageDataGenerator. Data augmentation techniques include rescaling, rotation, shifting, shearing, zooming, horizontal flipping, and specifying how to fill missing pixels. This helps to reduce overfitting by increasing the size of our dataset.
Model Training: The model is trained using the fit method. It trains for 10 epochs on the training data.

2) Convolutional Neural Network:

A CNN is used as a second model for image classification, specifically for classifying different fruits and vegetables based on input images. The model architecture defined in the code consists of convolutional layers for feature extraction and classification layers for making predictions.
The CNN is trained on a dataset of fruit and vegetable images. During training, it learns to recognize patterns and features that distinguish one fruit or vegetable from another. After training, the model can be used to predict the class labels of new, unseen images, making it suitable for tasks like automated image classification and object recognition.
In this context, the CNN serves as a powerful tool for automating the identification of fruits and vegetables, which can have various applications, including quality control in agriculture, inventory management in grocery stores, and automated sorting in food processing industries.

The following are the steps to creating and training a custom CNN model for ingredient recognition:

a) Model Architecture Definition: A sequential CNN model is created using the Convolutional layers, max-pooling layers, and dense layers are added to create a feedforward architecture. Batch normalization is applied for better convergence.

b) Convolutional Layers: Convolutional layers are added. These layers are responsible for learning spatial features from the input images. The activation function used is ReLU (Rectified Linear Unit).

c) Max-Pooling Layers: Max-pooling layers are used to down-sample the feature maps. This reduces computational complexity and helps the model generalize better.

d) Flatten Layer: After convolution and pooling layers, a flatten layer is added to convert the 2D feature maps into a 1D vector, which can be fed into the dense layers.

e) Dense Layers: Dense layers are added for classification. These layers learn to make predictions based on the features extracted by the previous layers. ReLU activation is applied to these dense layers.

f) Dropout Layer: A dropout layer is included for regularization. It randomly sets a fraction of input units to 0 during each update, preventing overfitting.

g) Output Layer: The final dense layer has 36 units, corresponding to the number of classes (fruits and vegetables) in the dataset. The activation function is softmax, which provides probabilities for each class.

h) Model Compilation: The model is compiled using the 'Adam' optimizer, categorical cross-entropy loss function (appropriate for multi-class classification), and accuracy as the evaluation metric.

i) Data Augmentation: The training data is augmented with operations such as rescaling, rotation, shifting, shearing, zooming, and horizontal flipping.

j) Data Splitting: The dataset is split into training, validation, and test sets. The validation set is used for monitoring the model's performance during training.

k) Training the Model: The model is trained using the fit method. Training and validation data are provided, and the number of training epochs (36) is specified. The history of training is recorded.

D. Model Implementation for Recipe Recommendation:

1) Word2Vec:

The Word2Vec model, specifically the Continuous Bag of Words (CBOW) variant, plays a pivotal role in transforming ingredients into continuous vector representations in a high-dimensional space.

The Word2Vec model, specifically CBOW, is selected for training. CBOW focuses on predicting a target word based on the context words that surround it. In the context of our recipe recommendation system, it's used to learn the semantic relationships between ingredients.

The training process requires a corpus, which is a collection of ingredient lists represented as documents. Each ingredient list is essentially a list of words. This corpus is passed to the Word2Vec model for training.

The following parameters were tuned to suit our application:

sg=0: The parameter sg stands for "skip-gram." When set to 0, it signifies the utilization of the CBOW model. This choice is fundamental for the system to implement CBOW effectively.

workers=8: The workers parameter specifies the number of CPU cores allocated for training the Word2Vec model. Utilizing multiple cores can significantly speed up the training process.

window=get_window(corpus): The window parameter is set using the get_window function. This function calculates the average length of the documents (ingredient lists) in the corpus. The window size in CBOW represents the number of context words considered when predicting the target word. It's adjusted based on the document length to capture the appropriate context.

min_count=1: The min_count parameter is crucial for determining the minimum word frequency required for a word to be included in the vocabulary. In our system, words with a frequency of less than 1 are excluded from consideration.

vector_size=100: This parameter defines the dimensionality of the word vectors. In this case, each word is represented as a 100-dimensional vector. The vector size impacts the granularity and capacity of the model, and in this context, it is set to 100 dimensions.

a) Document Embeddings: This phase involves the calculation of document embeddings, which are crucial for comparing user input with the existing recipe dataset. The system attempted two approaches for generating these embeddings:

MeanEmbeddingVectorizer: Document Embeddings Calculation: This method calculates document embeddings by averaging the word embeddings of ingredients within each recipe. Essentially, it computes the average vector representation of the ingredients in a recipe, resulting in a single vector representing the entire recipe.

TfidfEmbeddingVectorizer: Document Embeddings Calculation: This method calculates document embeddings using Term Frequency-Inverse Document Frequency (TF-IDF) weighting. It assigns weights to words (ingredients) based on their importance in the document. Rare ingredients are given more significance.

Through this Word2Vec model training, our system equips itself with the ability to understand and represent ingredients in a continuous vector space. It captures the semantic relationships between ingredients, allowing for the comparison and recommendation of recipes based on ingredient similarity. The chosen parameters, such as the model variant and vector size, are pivotal in shaping the effectiveness of the trained model.

2) *Long Short-Term Memory:*

The 'parsed' ingredients are prepared in the preprocessing part, which cleans, tokenizes, and lemmatizes the individual words. This processed ingredient list is used as input for the LSTM model.

The Tokenizer class from the Keras library is used to tokenize the ingredients. This means converting each word into a unique integer. A vocabulary is created, mapping words to integer IDs. The size of this vocabulary is determined by the number of unique words in the ingredient lists.

a) Word Embeddings: An Embedding layer is created in a Keras Sequential model. This layer is responsible for mapping the integers (word IDs) to dense vectors of a fixed size. It's similar to Word2Vec, but in this case, it's learned within the LSTM model. By training the model to generate word embeddings, it learns to capture the semantic relationships between ingredients. For example, it might learn that "tomato" and "tomatoes" are related terms and should have similar embeddings. These word embeddings help convert ingredient text into a numerical format that the model can work with.

The following parameters were tuned to suit our application:

Input dimension: Size of the vocabulary

Output dimension: Size of the word embeddings

Input length: length of each input sequence, which is the maximum number of words in a recipe.

The Embedding layer is used to create word embeddings for all words in the ingredients. These embeddings capture the semantic relationships between words in the context of the recipes.

b) Document Embeddings: An LSTM layer is added to the Sequential model. This LSTM layer takes the word embeddings as input and is responsible for creating document embeddings (representations of entire recipes).

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that is particularly good at handling sequential data. In this case, it's used to understand the context and relationships between ingredients in a recipe.

The LSTM layer typically has a certain number of units or neurons, which can be adjusted to control the complexity and capacity of the model.

After processing the entire ingredient list through the Embedding and LSTM layers, the LSTM layer outputs a document embedding. This embedding represents the entire recipe, capturing the contextual information between ingredients.

The word and document embeddings serve as numerical representations of recipes and ingredients.

The following parameters were tuned to suit our application:

Number of LSTM Units: The number of LSTM units (also known as neurons or cells) determines the model's capacity to capture complex patterns.

Number of LSTM Layers: Single or multiple LSTM layers stacked on top of each other can be used to create a deeper network. Deeper networks can capture hierarchical features in the data but require more computational resources.

Activation Functions: This project employed the tanh function, which is the default activation function in Keras.

3) *FastText (Gensim):*

FastText is a popular word embedding model developed by Facebook's AI Research (FAIR) lab. It is an extension of the Word2Vec model but with some key differences and improvements. The primary goal of FastText is to learn continuous vector representations (embeddings) for words or sub-words in a text corpus.

FastText takes into account not only whole words but also sub-word units, such as character n-grams and sub-tokens. This sub-word information enables the model to understand complex or out-of-vocabulary ingredients in recipes..

It is context-aware, meaning it considers the surrounding words when learning word embeddings, which is vital for recognizing ingredient meanings within recipes.

Furthermore, FastText uses a hierarchical soft-max approach to speed up training and make it efficient for large ingredient vocabularies.

The model's training process encompasses the following elements:

a) Contextual Learning: FastText excels in understanding words in context, considering not just isolated tokens but their contextual surroundings. This is crucial for capturing the meaning of ingredients in recipes.

b) Semantic Significance: Through training, FastText generates dense vector representations for tokens based on their dataset usage, enabling it to comprehend relationships and similarities between words or ingredients. For instance, it discerns that "tomato" and "bell pepper" share a closer connection than "tomato" and "computer."

c) Word Embedding Mastery: The training fine-tunes model parameters by predicting the context of a target word. This process results in word embeddings rich in semantic content, valuable for various natural language tasks, including similarity assessments and recommendations.

d) Continuous Vector Space: FastText establishes a coherent vector space where tokens with akin meanings are positioned closer. This spatial arrangement facilitates precise similarity calculations, underpinning our project's recipe recommendations based on ingredient similarity.

The following parameters were tuned to suit our application:

ingredient_tokens: This is the tokenized list of ingredients in our dataset.

vector_size=100: It specifies the dimensionality of the embeddings. In this case, the embeddings are 100-dimensional vectors. A higher dimension allows for more detailed representations

e) Creating Query Ingredient Embedding: Once the model was trained, it was used to create embeddings for a query ingredient list. The query ingredients are first tokenized, just like the training ingredients. This ensures that each ingredient in the query is represented as a sequence of tokens. The code for our project calculates an embedding for the entire query ingredient list by averaging the embeddings of the individual ingredient tokens. For each ingredient token, the corresponding embedding is retrieved from the trained FastText model.

Finally, to create the query ingredient embedding, the embeddings of all the ingredient tokens are averaged. This process generates a single vector that represents the semantic content of the entire query.

The resulting query ingredient embedding can be used to compare the similarity between the query and the recipes in the dataset.

4) *BERT (Bidirectional Encoder Representations from Transformers)*

We chose to use a pre-trained BERT model, specifically " bert-base-nli-mean-tokens" from the Sentence Transformers library. BERT models are pre-trained on extensive text data, allowing them to understand the context and meaning of words in a given text.

BERT is a language model pre-trained on vast text data. It understands word meanings in context, considering words bidirectionally as it uses the powerful Transformer neural network architecture. It also consists of multiple layers for deep contextual understanding.

BERT generates context-aware word embeddings, unlike traditional static embeddings.

To process text effectively, BERT tokenizer was used. This tokenizer is designed to split text into smaller units, or tokens, making it suitable for BERT's input requirements. It can also manage special tokens used for structuring input.

a) BERT Sentence Embeddings: By leveraging the 'bert-base-nli-mean-tokens' model, we were able to generate high-quality sentence embeddings. These embeddings play a pivotal role in our project, as they are instrumental in capturing the underlying semantic meaning of both ingredients and recipes. By encoding text into rich, high-dimensional vectors, we can facilitate advanced semantic analysis, similarity assessment, and context-based search tasks, which are paramount in recipe recommendation and retrieval systems.

b) Indexing and Recipe Retrieval Functions: We created functions to map recipe names to their corresponding indices in the dataset and vice versa. These functions allow for efficient retrieval of recipes based on their names.

c) Generating Sentence Embeddings: We employed the BERT model to encode the parsed ingredients for each recipe and saved these embeddings in a file named 'sentence_embeddings.npy'. This step helps in precomputing embeddings for efficiency.

5) *Logistic Regression:*

Logistic Regression is a classification algorithm that's well-suited for tasks where the output is a category or class label, as in this case:

The Logistic Regression model is trained using the training set. It learns to make predictions based on the TF-IDF features of ingredients and the corresponding recipe names.

After training, the model can predict the most likely recipe name for a given set of ingredient features. It computes a

probability distribution over all possible recipe names and selects the one with the highest probability.
Then model assigns a recipe name to the input based on the class (recipe name) with the highest predicted probability.

TF-IDF stands for Term Frequency-Inverse Document Frequency. It's a technique used to convert text data into numerical features. The following steps demonstrates how it works for the logistic regression model:

Term Frequency (TF): This part measures the frequency of a word within a document (recipe in this case). It gives higher weights to words that appear more frequently in the document.
Inverse Document Frequency (IDF): This part evaluates the importance of a word across all documents (recipes). Words that are common across many documents get lower weights, while words that are unique to a few documents receive higher weights.
TF-IDF Score: The final score for a word in a specific document is the product of its TF and IDF scores. This means that words that are frequent in a particular recipe but rare across other recipes get higher scores, making them more distinguishing.
Vectorization: TF-IDF vectorization converts each recipe (document) into a numerical vector. Each dimension of this vector corresponds to a unique word (ingredient). The value in each dimension is the TF-IDF score of the ingredient in the recipe.
Training Set: This is used to train the Logistic Regression model. The model learns from this data by adjusting its parameters to make accurate predictions based on the input features.
Testing Set: This set is used to evaluate the model's performance. The model hasn't seen this data during training, so it's a good measure of how well it generalizes to new, unseen examples. By comparing the model's predictions on the testing set to the actual recipe names, we assessed its accuracy.

Finally, after implementing each model, we implemented a recommendation phase, in which, the system defines several functions to process and format recipe recommendations:
Model Loading: The function loads the trained model that was previously trained on a dataset of recipes. This model contains the knowledge of ingredient relationships and the query embedding.
Cosine Similarity Scores: The function calculates cosine similarity scores between the user's input embedding and the embeddings of all recipes in the dataset. This measures how closely the user's ingredient preferences align with each recipe.
Cleaning and Formatting: These functions are responsible for tidying up the recipe titles and ingredient lists, making them more user-friendly and presentable. This ensures that users receive well-structured recommendations.
Ranking and Extraction: The system implements a function to rank recipes based on similarity scores. It extracts the top N recommendations, presenting them to users. These recommendations include details such as recipe names, ingredients, URLs, cuisine, and their respective similarity scores.

E. Additional Features:

1) GPT-Powered Recipe Generator:

In today's data-driven world, artificial intelligence and natural language processing have found applications in various domains.
One intriguing area is recipe generation, where AI models like GPT-2 (Generative Pre-trained Transformer 2) can transform a list of ingredients into a full-fledged culinary recipe. Therefore, as an additional feature in our project, we enable users to use AI-generated recipes as well, using the ingredients input.
GPT-2, short for "Generative Pre-trained Transformer 2," is a state-of-the-art natural language processing model developed by OpenAI. It is a deep learning model that belongs to a class of models known as transformers. It is pre-trained on massive amounts of text data from the internet, enabling it to understand and generate human-like text.

Key Characteristics:
Language Generation: GPT-2 is a text generator. It can generate coherent and contextually relevant text based on the input it receives. This makes it useful for a wide range of natural language understanding and generation tasks. *Scalability:* GPT-2 comes in different sizes, ranging from a small model with fewer parameters to a very large model with hundreds of millions of parameters. The larger models have a better understanding of context and can generate more coherent text.
Contextual Understanding: GPT-2 can capture contextual relationships between words and phrases in a given text, which allows it to generate text that makes sense within a specific context.
Fine-tuning: While GPT-2 is pre-trained on diverse internet text, it can also be fine-tuned for specific tasks, making it adaptable to various applications such as text completion, translation, summarization, and more.

The following are the important components of our implementation:

a) Data Collection and Preparation: The first step is gathering a dataset that consists of recipes. This dataset contains recipe names, lists of ingredients, and cooking instructions. Data is a crucial component in training of GPT-2 in our project, as the model learns from the patterns and information within it. To ensure that the model isn't biased, the dataset is shuffled randomly.

b) Text Formatting: To effectively train a language model, text formatting is essential. The "print_recipe" function is used to display the details of a recipe. It helps visualize what the generated recipes will look like. The "form_string" function is responsible for combining the ingredients and cooking instructions into a single, coherent text format. This formatted text is what the model will process.

c) Data Split: The dataset is divided into two parts: a training set and a validation set. This is done to assess the model's performance during training and ensure that it generalizes well to unseen data. An 85-15% split is chosen, which reserves 85% of the data for training.

d) Custom Dataset Class: To prepare the data for GPT-2 training, a custom "RecipeDataset" class is created. This class is designed to process and encode the data. It produces input IDs and attention masks, which are necessary for training the GPT-2 model.

e) Collate Function: Batch processing is a crucial element of training efficiency. The "collate_fn" function is responsible for organizing the data into batches, making the training process more streamlined and memory-efficient

f) Training Configuration: Proper configuration of the training process is vital. This includes setting up training arguments such as the output directories, batch sizes, gradient accumulation strategies, the number of training epochs, and the strategy for saving model checkpoints.

g) Model Training: Training a sophisticated model like GPT-2 involves several steps. The code initializes an AdamW optimizer, which is a variant of the standard Adam optimizer, tailored for training deep learning models. Additionally, a learning rate scheduler is implemented to dynamically adjust the learning rate during training. The "Trainer" class from the transformers library is employed to manage the entire training workflow, encompassing data loading, model optimization, and checkpoint saving.

h) Model Saving: A crucial step post-training is to save the trained GPT-2 model. This is essential for preserving the model's fine-tuned parameters and ensuring that the knowledge gained during training is retained for future use.

i) Text Generation: After successful model training, the code is equipped to generate recipe instructions. A "create_prompt" function is introduced to format a list of ingredients into a proper prompt. With this prompt, the GPT-2 model is capable of generating detailed culinary instructions, making it an invaluable tool for cooking enthusiasts.

2) Nutritional Analysis using Edamam API

The integration of the Edamam API with our frontend represents a significant achievement in enhancing the user experience and providing valuable nutritional insights. With this connection, users can seamlessly access nutritional information directly through the frontend interface. This means that when a user inputs a recipe or food item, they receive instant, detailed data on its nutritional content. This information is presented in a user-friendly format, making it easy for individuals to make informed decisions about their dietary choices.

The integration also enables real-time updates, ensuring that users receive the most accurate and up-to-date nutritional information available.

By connecting the Edamam API to our frontend, we've created a valuable tool for users to achieve their health and wellness goals while using our application.

F. Project Setup

In this section, we provide a comprehensive overview of the initial setup and configuration of our project. This segment serves as a crucial starting point for readers and stakeholders to understand the foundation of our work.

By presenting a clear and concise account of our project's setup, we aim to make our work accessible and reproducible for both technical and non-technical audiences.

IV. RESULTS & ANALYSIS

A. Performance Metrics

1) Ingredient Recognition Models

Accuracy: This metric measures the overall correctness of the models' predictions by considering the ratio of correctly predicted instances to the total instances. It was used to assess the accuracy of ingredient recognition in images.

Precision: Precision is a crucial metric for image recognition and assesses the accuracy of the models' positive predictions. It measures how many of the instances classified as positive are true positives.

Recall: Recall evaluates the models' ability to identify all relevant instances within the dataset, measuring the ratio of true positives to the total number of actual positives.

F1-Score: The F1-Score, a harmonic mean of precision and recall, offers a balanced measure of the models' accuracy, considering both false positives and false negatives. It provides a comprehensive evaluation of performance.

2) Recipe Recommendation Models

Precision at K: Precision at K (where K = 10 in our case) evaluates the accuracy of the top K predictions made by the recommendation models. It focuses on the most relevant recommendations within the top K predictions, ensuring the quality of the top suggestions.

B. Comparative Analysis of Ingredient Recognition Models

1) Results

Custom CNN Model: The custom Convolutional Neural Network (CNN) model achieved an accuracy of 93%, indicating a strong overall performance. When examining precision, recall, and F1-scores for individual ingredient categories, the model demonstrated noteworthy results. It showcased particularly high F1-scores for some ingredients, underscoring its proficiency in recognizing these items. However, it faced challenges in the accurate identification of certain ingredients.

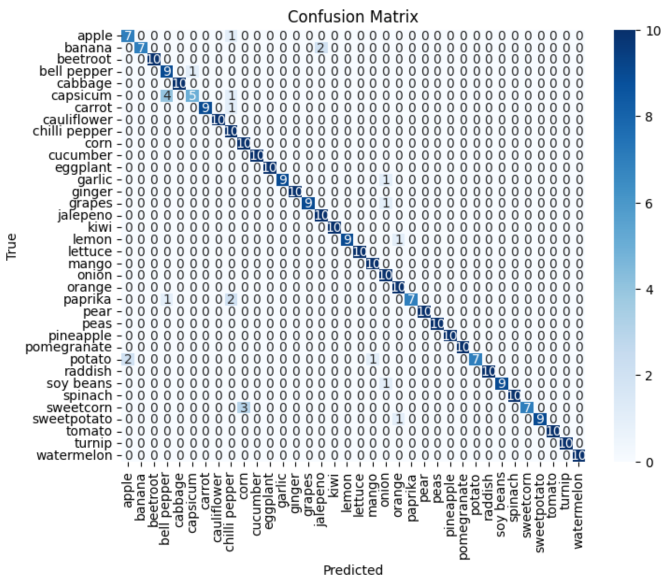

Figure 1: Confusion Matrix for CNN

MobileNetV2 Model: The MobileNetV2 model had an impressive accuracy of 95%. This model not only achieved high accuracy but maintained consistently strong precision, recall, and F1-score values. It excelled in recognizing a wide range of ingredients, with flawless F1-scores. The MobileNetV2 model demonstrated its robustness and proficiency in ingredient recognition.

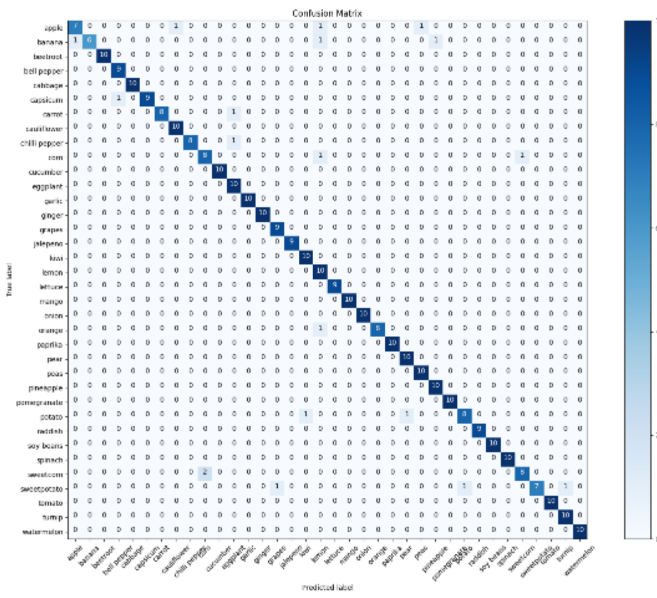

Figure 2: Confusion Matrix for MobileNetV2

2) *Model Comparison:*

We conducted a comprehensive comparison of our two image recognition models, focusing on accuracy, precision, recall, F1-scores and the confusion matrices for each ingredient category.

a) Custom CNN Model: Our custom Convolutional Neural Network (CNN) model showcases a commendable accuracy of 93%, reflecting its strong ability to identify a diverse array of ingredients. The model consistently delivers high precision, recall, and F1-scores, underscoring its proficiency. Notably, it excels in recognizing beetroot, cabbage, carrot, cucumber, and other ingredients, achieving impeccable F1-scores. However, it does face some challenges in the accurate identification of certain ingredients like apple and bell pepper, which is important to consider when assessing its performance.

b) MobileNetV2 Model: In contrast, the MobileNetV2 model emerges as the standout performer, surpassing the custom CNN model with an impressive accuracy of 95%. This model not only excels in recognizing ingredients but maintains consistently high recall and F1-score values. It demonstrates exceptional proficiency in recognizing a diverse range of ingredients, including beetroot, garlic, lemon, and more, with flawless F1-scores. The model's ability to generalize across different ingredients and maintain high accuracy is a testament to its robustness.

Comparative Insights: When comparing the two models, the MobileNetV2 model clearly outperforms the custom CNN model. It showcases a notable edge in terms of accuracy and generalization across various ingredients. The MobileNetV2 model consistently delivers superior precision, recall, and F1-scores. It presents a compelling case for its application in ingredient recognition tasks.
To gain a deeper understanding of the recognition capabilities of both models, we also included confusion matrices that provide valuable insights into the specific strengths and challenges encountered by each model when classifying ingredients.

C. *Comparative Analysis of Recipe Recommendation Models*

1) *Results*

In this section, we provide a comprehensive overview of the results achieved by various recipe recommendation models, with a primary focus on the precision at k (k=10) as the evaluation metric.

BERT Model using BERT Embeddings: This model, based on BERT embeddings, exhibited a modest performance with a precision at k of 0.1. The results were influenced by the model's ability to capture ingredient relationships and semantic meaning.

LSTM: The LSTM model, with a precision at k of 0.0, faced challenges in generating accurate recipe recommendations. Its ability to effectively capture ingredient relationships and context appeared to be less competitive when compared to other models.

GENSIM (FastText): Similar to the BERT model using BERT embeddings, the GENSIM model achieved a precision at k of 0.1. While it displayed reasonable performance, there remains room for improvement in recommendation accuracy.

Word2Vec and TF-IDF: Outperformed other models with a precision at k of 0.2. This approach, which combines Word2Vec embeddings and TF-IDF weighting, excelled in comprehending ingredient relationships and delivering precise recipe suggestions.

BERT Model using Sentence Transformer: This model secured the highest precision at k, achieving a score of 0.5. Leveraging sentence embeddings derived from BERT, it excelled in capturing the semantic meaning of ingredients and consistently provided recommendations that closely matched user preferences.

Logistic Regression: The Logistic Regression model yielded a precision at k of 0.0, indicating that it did not deliver accurate recipe recommendations. It struggled to encompass the complexity of ingredient relationships compared to more advanced models.

2) *Dataset Size and Its Impact:*

Our dataset comprises approximately 3,000 unique ingredients and 6,000 unique recipes. It's important to highlight that the ratio of unique ingredients to unique recipes in the dataset is almost 1:2, indicating that the amount of data may be insufficient for the recommendation models, especially those reliant on word embeddings, to achieve higher precision.

With a relatively small dataset, the models face limitations in effectively learning the semantic relationships and contextual understanding of ingredients. In particular, word embedding models require a substantial amount of data to capture the nuances of ingredient semantics, associations, and variations effectively. The relatively low number of recipes compared to the diversity of ingredients can result in a limited contextual understanding of ingredient relationships, making it challenging for the models to provide highly precise recommendations.

It's important to acknowledge that with a larger dataset, the models could potentially yield improved performance by having access to more varied and context-rich examples. In the context of our dataset, while some models demonstrated competitive performance, the dataset size remains a constraint that could be mitigated with an expansion in the number of recipes, enabling the models to better learn and comprehend ingredient relationships.

3) *Model Comparison:*

In this section, we offer an in-depth comparison of the various recipe recommendation models under consideration, using "precision at k" (k=10) as our primary evaluation metric. This metric enables us to assess how effectively these models provide recipe suggestions that closely align with users' preferences.

Model for Recipe Recommendation	**Precision at k (k=10)**
BERT Model using Sentence Transformer	0.5
Word2Vec and TF-IDF	0.2
GENSIM (FastText)	0.1
BERT Model using BERT Embeddings	0.1
LSTM	0.0
Logistic Regression	0.0

Table 1 Precision at k for all the Recipe Recommendation Models

BERT Model using Sentence Transformer: Emerging as the top-performing recommendation system, this model achieves an impressive precision at k of 0.5. Its remarkable ability to grasp the intricacies of ingredient semantics makes it the standout choice for ensuring highly accurate and user-centric recipe recommendations.

Word2Vec and TF-IDF: This model combination demonstrates commendable performance with a precision at k of 0.2, outshining several others. It exhibits a strong grasp of ingredient relationships, offering users precise and context-aware recipe recommendations.

GENSIM (FastText) and BERT Model using BERT Embeddings: Both models deliver a precision at k of 0.1, showcasing reasonable performance. However, there's room for improvement in terms of recommendation accuracy, suggesting the need for further model fine-tuning to capture intricate ingredient relationships effectively.

LSTM and Logistic Regression: Both models yield a precision at k of 0.0, indicating limited effectiveness in generating accurate recipe recommendations. These models face challenges in comprehending the complexities of ingredient relationships and might require additional enhancements to be considered viable options.

V. Discussion

For Ingredient Recognition, MobileNetV2 model emerged as the standout performer, surpassing the CNN model with an impressive accuracy of 95%. This model not only excelled in recognizing a diverse range of fruits and vegetables but maintained consistently high precision, recall and F1-score values- a testament to its robustness. The evaluation of the comparative analysis therefore led us to the decision to proceed with the MobileNetV2 model for its remarkable performance and robustness.

For Recipe Recommendation, the model comparison underscored the supremacy of the BERT Model using Sentence Transformer. This BERT Model, boasted a precision at k of 0.5. Its exceptional proficiency in understanding ingredient semantics and delivering highly precise recipe recommendations solidified its role as the optimal and final choice for our recipe recommendation system.

VI. Limitations

While our study presents promising results and a robust system for recipe recommendation and ingredient recognition, it is essential to recognize and discuss certain limitations that provide insight into areas for further research and development. These limitations encompass various aspects of our system, from ingredient recognition accuracy to the scope of dietary preferences. Understanding these constraints is vital for refining our system and guiding future work in the field of AI-powered culinary assistance.
In the following sections, we delve into each of these limitations, providing a comprehensive view of the challenges and opportunities for the Culinary Curator system.

1) *Ingredient Recognition Accuracy:*

While our ingredient recognition models, such as MobileNetV2 and the custom CNN, demonstrated strong performance, they may not achieve 100% accuracy. The recognition accuracy could be influenced by factors like the quality of user-uploaded images, variations in ingredient appearances, and occlusions.

2) *Limited Ingredient Dataset:*

The effectiveness of our ingredient recognition models heavily depends on the dataset used for training. If the training dataset lacks certain ingredients or contains limited variations, the system may struggle to accurately recognize less common or regional ingredients.

3) *Dietary and Allergen Preferences:*

While our system takes dietary preferences into account, it may not fully address complex dietary restrictions or allergies. Users with highly specific dietary requirements might not receive recommendations that align perfectly with their needs.

4) Real-Time Ingredient Availability:
Although our system considers the ingredients users have on hand, it doesn't have real-time access to local grocery store inventory or individual users' inventory. Therefore, ingredient availability is based on user input and might not always reflect real-world conditions.

5) User-Generated Content and Regional Coverage:
The system does not currently support extensive user-generated content, which could enhance the diversity of recipes and ingredients. Additionally, while we focus on Indian cuisine, the coverage of other global culinary traditions is limited.

6) Nutritional Data Accuracy:
The nutritional analysis component relies on a nutritional database, and the accuracy of the data is contingent on the quality and comprehensiveness of this database. Inaccuracies or missing data in the database could affect the precision of nutritional information provided.

7) Model Training Resources:
Training deep learning models requires substantial computational resources. Access to high-end GPUs or cloud computing may be a limitation for users who wish to replicate or further develop our system.

8) User Interface and Accessibility:
The current user interface and system accessibility could be improved for a more seamless and inclusive user experience. Usability and accessibility issues may impact certain users' ability to benefit fully from the system.

These limitations are important to acknowledge as they represent areas for potential future improvements and research. By addressing these challenges, the Culinary Curator can continue to evolve and offer an even more refined culinary experience.

VII. Conlusion

Culinary Curator marks a significant milestone in AI-driven culinary assistance, focusing on recipe recommendation and ingredient recognition. Our primary goal was a comprehensive evaluation of five recipe recommendation models and two ingredient recognition models. Notably, the BERT Model using Sentence Transformer for recipe recommendation and MobileNetV2 for ingredient recognition, emerged as standout performers.

The project's pillars encompass recognizing the importance of ingredient availability, exploring the world of Indian cuisine, integrating nutritional analysis, and providing a comparative analysis of various models, pushing culinary technology's boundaries.

In the vast landscape of culinary innovation, the Culinary Curator shines as a beacon of sophistication and practicality, offering tailored recommendations, nutritional insights, and a gateway to the captivating realm of Indian cooking.

Our journey forward holds limitless possibilities for the Culinary Curator, a stepping stone to a future where culinary technology is an indispensable tool for chefs of all levels. With solid foundations and identified paths for future development, the Culinary Curator is poised to adapt to evolving culinary needs.

VIII. Future Scope

Looking ahead, the Culinary Curator stands at the threshold of exciting possibilities and further enhancements. The journey of this research and the development of our AI-driven culinary assistant have illuminated the immense potential for culinary technology to transform the way we approach meal planning and cooking.While our project has already made significant strides in addressing contemporary challenges related to home cooking, this is merely the beginning of a culinary revolution.

In this section, we delve into the promising future scope of the Culinary Curator, encompassing several key areas for improvement and expansion that will not only enrich the user experience but also contribute to the broader culinary landscape. The following points outline the avenues of exploration that will take the Culinary Curator to new heights, ensuring its continued relevance and utility in the ever-evolving world of gastronomy.

Continuous refinement of the user interface and overall user experience is essential. This includes intuitive design, seamless navigation, and improved accessibility to make the Culinary Curator even more user-friendly. Advanced computer vision techniques and models to further boost the accuracy and capabilities of ingredient recognition can be explored. This may involve fine-tuning existing models or integrating new ones to handle a wider range of ingredients and complex scenarios, such as regional variations.

We may also implement a variety of language models for recipe generation. This diversity can cater to different user preferences, allowing for the creation of recipes in various styles, cuisines, and dietary requirements. Features that allow users to specify dietary restrictions or allergies to receive recipe recommendations tailored to their needs can be incorporated. This feature can significantly enhance the usability of the system for individuals with specific dietary requirements. Suggestions for ingredient substitutions, ensuring that users can adapt recipes to what they have on hand can be provided. This can be particularly helpful for users who may not have access to all ingredients listed in a recipe
.
To drive AI-powered culinary assistance, we must look not only to the present but also to the future. The Culinary Curator holds the promise of greater capabilities, expanded functionalities, and a deeper impact on how we understand and engage with food. As this culinary revolution unfolds, the Culinary Curator's journey has just begun, and the future looks exceptionally promising.

References

[1] M. B. Vivek, M. N. Manju, and M. B. V. Vijay, &quot;Machine Learning Based Food Recipe
Recommendation System,&quot; 2023.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay, &quot;Deep Learning based Recommender System: A Survey

and New Perspectives,&quot; IEEE Communications Magazine, vol. 57, no. 9, pp. 120-128, 2019.

[3] A. Dharawat and C. Doan, &quot;Ingredient Extraction from Text in the Recipe Domain,&quot; 2022.

[4] M. S. Rodrigues, F. Fidalgo, and A. Oliveira, &quot;RecipeIS—Recipe Recommendation System

Based on Recognition of Food Ingredients,&quot; 2023.

[5] R. Patil, N. Shirsat, K. Bumb, and A. Satpute, &quot;Identify Ingredients From Food Images and

Generate Recipe,&quot; 2023.

[6] H. H. Lee, K. Shu, P. Achananuparp, P. K. Prasetyo, Y. Liu, E.-P. Lim, and L. R. Varshney,

&quot;RecipeGPT: Generative Pre-training Based Cooking Recipe Generation and Evaluation System,&quot;

arXiv preprint arXiv:2007.07788, 2020.

[7] T. Theodoridis, V. Solachidis, K. Dimitropoulos, L. Gymnopoulos, and P. Daras, &quot;A Survey on AI

Nutrition Recommender Systems,&quot; IEEE Transactions on Emerging Topics in Computational

Intelligence, vol. 6, no. 1, pp. 81-96, 2022.

[8] H. Onyeaka, P. Tamasiga, N. Uju, T. Miri, U. C. Juliet, O. Nwaiwu, and A. A. Akinsemolu,

&quot;Using Artificial Intelligence to Tackle Food Waste and Enhance the Circular Economy: Maximising

Resource Efficiency and Minimising Environmental Impact: A Review,&quot; arXiv preprint

arXiv:2302.10659, 2023.

[9] Y. Tian, C. Zhang, R. Metoyer, and N. V. Chawla, &quot;Recipe Recommendation With Hierarchical

Graph Attention Network,&quot; arXiv preprint arXiv:2203.10795, 2022.

[10] K. Taneja, R. Segal, and R. Goodwin, &quot;Monte Carlo Tree Search for Recipe Generation using

GPT-2,&quot; arXiv preprint arXiv:2011.09522, 2020.

[11] J. Chen and C. Ngo, &quot;Deep-based Ingredient Recognition for Cooking Recipe Retrieval,&quot;

Proceedings of the 2016 ACM on Multimedia Conference, pp. 1373-1382, 2016.

[12] A. Mane, M. Bawankar, M. Goradia, and S. Chaudhari, &quot;Image to Recipe and Nutritional Value

Generator Using Deep Learning,&quot; arXiv preprint arXiv:2203.13713, 2022.

[13] M. Bolaños, A. Ferrà, and P. Radeva, &quot;Food Ingredients Recognition Through Multi-label

Learning,&quot; Lecture Notes in Computer Science, pp. 500-511, 2017.

[14] R. Maia and J. C. Ferreira, &quot;Context-Aware Food Recommendation System,&quot; Proceedings of the

2018 ACM International Conference on Information and Knowledge Management, pp. 2519-2522,

2018.

[15] B. P. Majumder, S. Li, J. Ni, and J. McAuley, &quot;Generating Personalized Recipes from Historical

User Preferences,&quot; Proceedings of the 28th ACM International Conference on Information and

Knowledge Management, pp. 2109-2118, 2019.

[16] J. Webster and C. Kit, &quot;Tokenization as the initial phase in NLP,&quot; 2023.

[17] D. Khyani and S. B. S, &quot;An Interpretation of Lemmatization and Stemming in Natural Language

Processing,&quot; 2023

[18] Kritik Seth, &quot;Fruits and Vegetables Image Recognition Dataset,&quot; Kaggle 2020

[https://www.kaggle.com/kritikseth/fruit-and-vegetable-image-recognition]

[19] Kanishk Jain, “6000+ Indian Food Recipes Dataset”, Kaggle 2020

[https://www.kaggle.com/datasets/kanishk307/6000-indian-food-recipes-dataset]