Culinary Companion

A Project submitted to Bharathiar University

In the partial fulfilment of the requirement for the degree of

MASTER OF COMPUTER SCIENCE

ARTIFICIAL INTELLIGENCE

To be awarded by Bharathiar University,coimbatore-641046

Submitted by

KAMAL KISHORE S (24CSER010)

LOGA BALAJI J (24CSER013)

RAHUL M (24CSER020)

Date : 04.10.2024

Under the guidance of

DR. P. B. PANKAJAVALLI , M.E., M. PHIL.,

Assistant Professor , Department of Computer Science

DEPARTMENT OF COMPUTER SCIENCE

BHARATHIAR UNIVERSITY

2024-2026

ABSTRACT

The **Culinary Companion** project presents an innovative approach to recipe generation by leveraging advanced technologies such as Optical Character Recognition (OCR), Recurrent Neural Networks (RNN), and text-to-speech capabilities. Designed to enhance the cooking experience, the system automatically identifies available ingredients through a camera, eliminating the need for manual input. Utilizing a dataset of over 21 million recipes, the RNN model achieves an impressive accuracy of 0.96, enabling the generation of personalized recipes based on the detected ingredients. The processed data combines essential features—ingredients, recipe, direction, and Named Entity Recognition (NER)—into a unified string, facilitating efficient training. The implementation of a Gated Recurrent Unit (GRU) layer within the model architecture enhances its performance, particularly given the large dataset. Furthermore, the integration of text-to-speech functionality allows users to listen to recipe instructions, promoting a hands-free cooking experience. Overall, the Culinary Companion represents a significant advancement in kitchen technology, providing an accessible, efficient, and personalized solution for everyday cooking challenges.

TABLE OF CONTENTS

1. INTRODUCTION

The food industry has seen significant growth in technology-based solutions to improve the cooking experience. However, most of these tools still rely on manual entry of ingredients or provide recipes that may not match what a user has on hand. This project, titled "Culinary Companion," aims to address this gap by creating an intelligent recipe generation system that dynamically responds to the ingredients available in a user's kitchen.

By leveraging cutting-edge techniques like Optical Character Recognition (OCR) for ingredient detection and a Recurrent Neural Network (RNN) model for recipe generation, Culinary Companion provides a seamless experience where users can receive personalized recipe suggestions without manually entering data. The system also converts these recipes into audio, providing a hands-free, user-friendly solution for cooking.

1.1. PROPOSED SYSTEM

The Culinary Companion system is designed to provide a seamless, intelligent solution to the challenge of generating recipes based on the ingredients a user has available. At the core of this system is a camera that automatically captures and identifies ingredients by reading the text on labels attached to jars or containers. By employing Optical Character Recognition (OCR) technology, the system extracts the text from these labels, allowing it to accurately detect the ingredients without requiring manual input from the user. Once the ingredients are identified, they are passed to a pre-trained Recurrent Neural Network (RNN) model, which processes the input and generates appropriate recipes. The RNN model is specifically trained on a vast dataset of recipes, enabling it to understand the relationships between ingredients and produce contextually relevant recipe suggestions.

After the recipe is generated, the system converts the text output into an audio format using text-to-speech technology. This feature allows users to listen to the recipe instructions, providing a hands-free cooking experience. The user interface is designed to be simple and intuitive, allowing users to view, save, or listen to recipes easily, ensuring that the system is accessible to users of all technical backgrounds. Overall, Culinary Companion integrates modern technologies to offer an efficient, user-friendly way to enhance the cooking process based on available ingredients.

2. SYSTEM SPECIFICATION

2.1. SOFTWARE SPECIFICATION

- Visual Studio Code
- Jupyter Notebook
- Ubuntu 22
- PYTHON 3.12

2.2. HARDWARE SPECIFICATION

- RTX - 3060
- Camera

2.3 SOFTWARE AND HARDWARE DESCRIPTION

2.3.1. Software Description

- **Visual Studio Code (VS Code):** VS Code is a widely-used source code editor that supports multiple programming languages and frameworks. In this project, it is used for writing, editing, and debugging the Python code that runs on any operating system. Its extensive extensions and tools make it a powerful Integrated Development Environment (IDE) for coding, testing, and deploying the smart security system. Developers use VS Code to write and manage the scripts that control various aspects of the IoT-based smart security system, such as motion detection, image capture, text generation.

- **Jupyter Notebook**: Jupyter Notebook is an open-source web application that allows developers to create and share documents containing live code, equations, visualizations, and narrative text. In this project, Jupyter Notebook is used for developing, testing, and documenting the Python code that powers the recipe generation system. Its interactive environment supports data exploration, model training, and code iteration, making it an ideal tool for building and refining the RNN model used in the Culinary Companion project. With its ability to run code in real-time, Jupyter Notebook provides a flexible platform for experimenting with machine learning algorithms, visualizing the data flows, and debugging issues throughout the development process.

- **Ubuntu 22**: Ubuntu 22 is a popular and stable Linux distribution widely used for software development, particularly in machine learning and artificial intelligence projects. In this project, Ubuntu 22 serves as the operating system for developing and deploying the recipe generation model. It provides a robust environment for running Python scripts, managing dependencies, and configuring necessary libraries for machine learning tasks. Notably, Ubuntu 22 supports CUDA, allowing the project to leverage GPU acceleration for

training the RNN model efficiently. With CUDA-enabled GPU support, the system can handle computationally intensive tasks, such as deep learning model training, significantly faster than CPU-based alternatives. This makes Ubuntu 22 an excellent choice for handling the resource-intensive processes involved in the Culinary Companion project.

- **Python 3.11**: Python 3.11 is the latest stable release of the Python programming language, known for its simplicity, readability, and extensive ecosystem of libraries and frameworks. In this project, Python 3.11 is used as the primary programming language for developing the machine learning model, handling data input/output, and integrating various components of the Culinary Companion system. Python 3.11 offers enhanced performance and improved error messages, which are particularly beneficial during the training and fine-tuning of the Recurrent Neural Network (RNN) model. With support for libraries such as TensorFlow, PyTorch, and OpenCV, Python 3.11 enables efficient model training, data preprocessing, and text recognition tasks. Its compatibility with both CPU and GPU environments ensures flexibility in deploying the project across different hardware configurations.

2.3.2. Hardware description

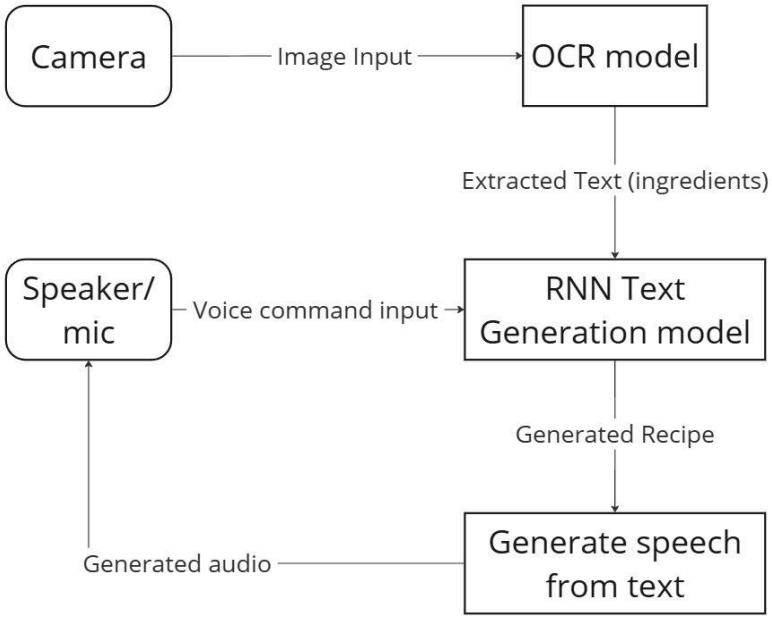
- **NVIDIA RTX 3060:** RTX 3060 is a highly capable graphics card, well-suited for machine learning tasks, particularly in training large Recurrent Neural Network (RNN) models. Featuring 12GB of GDDR6 memory and built on the Ampere architecture, the RTX 3060 provides substantial computational power and supports NVIDIA's CUDA (Compute Unified Device Architecture), allowing developers to harness GPU acceleration. In this project, CUDA support enables faster training of the large-scale RNN model by utilizing the GPU's parallel processing capabilities, significantly reducing the training time compared to traditional CPU-based approaches. The RTX 3060's balance of performance and cost makes it an excellent choice for handling the resource-intensive deep learning tasks required for the Culinary Companion system.

- **Camera**: The camera plays a critical role in the Culinary Companion system, serving as the primary tool for capturing the ingredients used in recipe generation. It is responsible for scanning and detecting the labels on jars and containers, which typically hold various ingredients. The camera's high-resolution image capture ensures that the text on the labels is clear, enabling the system's Optical Character Recognition (OCR) technology to accurately extract the ingredient names. By automating the ingredient detection process, the camera eliminates the need for manual input, enhancing the user experience and streamlining the process of generating recipes. The real-time capture capability of the camera allows the system to swiftly pass the

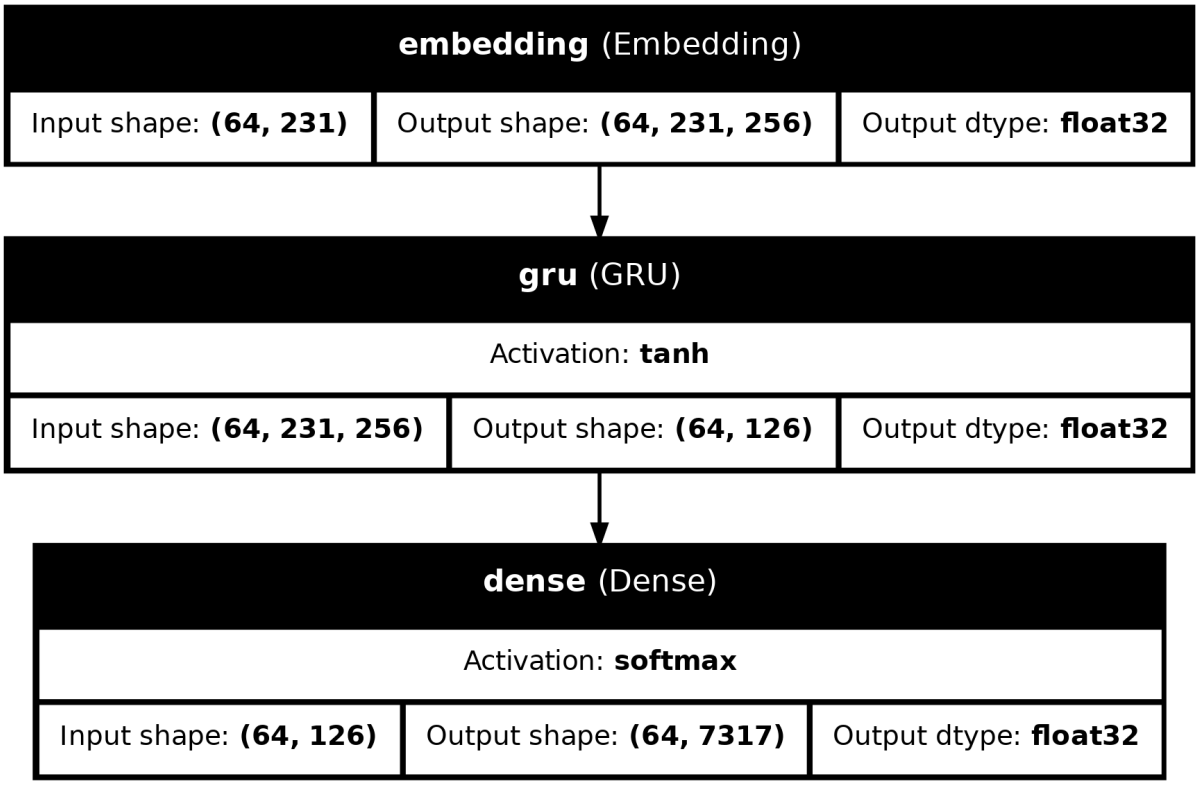
extracted text to the RNN model, which uses this data to generate relevant recipes.

3. DESIGN APPROACHES

3.1 IOT Architecture



3.2 RNN model Architecture



4. MODEL TRAINING

The training process for the RNN model in the Culinary Companion project involved collecting a substantial dataset comprising over 21 million recipes. This extensive dataset serves as the foundation for enabling the model to generate relevant recipes based on available ingredients.

Data processing began with extracting and combining essential features, specifically "ingredients," "recipe," "direction," and "Named Entity Recognition (NER)," into a single string. This approach simplifies the training process, as training an RNN model on a unified string structure is more efficient. Following this, the combined string was tokenized to convert the text data into numerical representations, which are crucial for machine learning models. Additionally, an n-gram sequence was generated from the tokens, allowing the model to capture the context and relationships between words effectively.

The dataset was then split into features and labels, with the features containing the input strings and the labels representing the corresponding recipes. To build the RNN model, a sequential architecture was defined featuring an embedding layer, a Gated Recurrent Unit (GRU) layer, and a dense layer with a softmax activation function. The choice of the GRU layer was particularly important due to the large size of the dataset, as GRUs are known for their efficiency in training on extensive datasets while maintaining performance.

Once the model architecture was defined, the dataset was fitted into the model, and training commenced. After training, rigorous testing was conducted, and accuracy metrics were verified to ensure reliable performance. The trained model, along with the tokens, was then saved for integration with the IoT components of the Culinary Companion system.

5. SOURCE CODE

recipe recommendation mode.ipynb

```
import numpy as np

import pandas as pd

import pickle

import matplotlib.pyplot as plt


import tensorflow as tf

import tensorflow.keras as keras

from tensorflow.keras.preprocessing.text import Tokenizer


tf.config.run_functions_eagerly(True)


gpu_devices = tf.config.list_physical_devices('GPU')

details = tf.config.experimental.get_device_details(gpu_devices[0])

print("Available GPU: ", details.get('device_name'))


rawdataset = pd.read_csv("./dataset/full_dataset.csv")

print("Full Dataset shape:", rawdataset.shape)

dataset = rawdataset[:1900]

dataset = dataset.drop(["link", "source"], axis=1)

print("Data shape:", dataset.shape)

dataset.head()


def clean_text(txt):
```

```
    txt = "".join(v for v in txt if v not in '!"#$%&*+:;<=>?@[\]^_`{}~').lower()
    return txt


title = [clean_text(x) for x in dataset.title]
NER = [clean_text(x) for x in dataset.NER]
ingredients = [clean_text(x) for x in dataset.ingredients]
directions = [clean_text(x) for x in dataset.directions]


finaldataset = []
for i in range(len(NER)):
    finaldataset.append(NER[i]+" | "+title[i]+" | "+ingredients[i]+" | "+directions[i]+ " ~ ")
max_length = 0
for i in finaldataset:
    max_length = max(max_length, len(i))
print("max_length:", max_length)
print()
print("Recipe at index 0:\n", finaldataset[0])


tokenizer = Tokenizer(
    char_level=False,
    filters='',
    lower=False,
    split=' '
)
tokenizer.fit_on_texts(finaldataset)
```

```
#tokenizer.fit_on_texts(["~"])


vocab_size = len(tokenizer.word_index) + 1

print("Total words:", vocab_size)


text = ["pineapple condensed milk lemons pecans graham cracker crusts"]

token_list = tokenizer.texts_to_sequences(text)

print(text)

print(token_list)


input_sequences = []

for line in finaldataset:

    token_list = tokenizer.texts_to_sequences([line])[0]

    for i in range(1, len(token_list)):

        n_gram_sequence = token_list[:i+1]

        input_sequences.append(n_gram_sequence)


input_sequences = np.array(keras.utils.pad_sequences(input_sequences, maxlen=max_seq_len, padding = 'pre'))


features, labels = input_sequences[:, :-1], input_sequences[:, -1]

labels = keras.utils.to_categorical(labels, num_classes=vocab_size)


n = 0.1

slice_size = int(len(features)*n)
```

```
print("Slice size:", slice_size)
features = features[:slice_size, :]
labels = labels[:slice_size]
print("Features Shape:", features.shape)
print("Labels Shape:", labels.shape)
print("dataset size in disk")
print("Features:", features.nbytes/1e+9)
print("Labels:", labels.nbytes/1e+9)
def generator_model():
    model = keras.Sequential([
    keras.layers.Embedding(vocab_size, 256, input_length=max_seq_len),
    keras.layers.GRU(126),
    keras.layers.Dense(vocab_size, activation='softmax')
    ])
    model.compile(loss = 'categorical_crossentropy',
            optimizer = keras.optimizers.Adam(learning_rate = 0.002),
            metrics = ['accuracy'])
    return model
model = generator_model()
history = model.fit(features, labels, epochs = 25, batch_size=64)
print("Model Accuracy:", max(history.history['accuracy']))
print("Loss:", min(history.history['loss']))
```

6. ADVANTAGES AND DISADVANTAGES

6.1 Advantages of Culinary Companion

- Hands-Free Interaction: The system automates the process of ingredient detection using a camera and generates recipes without requiring manual input, making it highly convenient for users.
- Personalized Recipes: By generating recipes based on the available ingredients, Culinary Companion tailors the cooking experience to what the user already has, minimizing food waste and unnecessary shopping.
- Enhanced Accessibility: The text-to-speech functionality provides audio instructions, making the system accessible to users who prefer or need a hands-free cooking experience, such as those with visual impairments.
- Time-Saving: Automated ingredient detection and recipe generation save users the time they would otherwise spend looking up recipes or entering ingredients manually.
- Scalability: The system can easily adapt to accommodate more advanced models, larger datasets, or additional features, like integrating nutritional information or dietary preferences.
- Cutting-Edge Technology: The project leverages modern technologies such as OCR, machine learning (RNN) and Natural language processing (NLP), making it an advanced solution in the kitchen tech space.

6.2 Disadvantages of Culinary Companion:

- **Dependence on High-Quality Image Capture**: The system relies heavily on the camera's ability to accurately capture clear images of ingredient labels. Poor lighting or unclear text on the label could lead to incorrect or incomplete ingredient detection.
- **Complex Hardware Setup**: For optimal performance, the project may require specific hardware, such as high-quality cameras, which could increase the cost for users.
- **Model Limitations**: The RNN model may not always generate perfect or accurate recipes, particularly if the available ingredients are uncommon or if the model hasn't been trained on a sufficiently diverse dataset.
- **Initial Setup and Maintenance**: Installing and maintaining the system might require technical expertise, especially for setting up the machine learning environment and ensuring compatibility with other software components.

- **Resource-Intensive**: The training of large RNN models and real-time image processing can be resource-intensive, requiring substantial computational power, which might not be available in all user setups.
- **Potential Errors in OCR**: Optical Character Recognition (OCR) technology is not perfect and may struggle with certain fonts, poor lighting conditions, or misaligned labels, leading to incorrect ingredient identification.

7. CONCLUSION

In conclusion, the **Culinary Companion** project successfully integrates advanced technologies to create an innovative solution for recipe generation based on available ingredients. By utilizing a camera for automatic ingredient detection, an RNN model for recipe generation, and text-to-speech functionality for audio instructions, the system offers a hands-free and user-friendly cooking experience. The RNN model demonstrates impressive performance, achieving an accuracy of 0.96, which highlights its capability to understand and generate relevant recipes based on the ingredients provided.

This project not only enhances convenience and accessibility in the kitchen but also promotes resource efficiency by minimizing food waste through personalized recipe suggestions. While there are some challenges to address, such as ensuring high-quality image capture and managing complex hardware requirements, the Culinary Companion represents a significant step forward in leveraging artificial intelligence and machine learning to improve everyday cooking tasks. Future work could involve refining the model, expanding its recipe database, and enhancing user interface features to further enrich the cooking experience. Overall, Culinary Companion has the potential to transform how individuals approach meal preparation, making it a valuable addition to modern culinary practices.

8. REFERENCE

- Moolya, Disha, et al. "Recipe generator using Deep Learning." International Journal for Research in Applied Science and Engineering Technology 10.5 (2022): 846-851.
- Wang, Hao, et al. "Learning structural representations for recipe generation and food retrieval." IEEE Transactions on Pattern Analysis and Machine Intelligence 45.3 (2022): 3363-3377.
- Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).
- Chaudhuri, Arindam, et al. Optical character recognition systems. Springer International Publishing, 2017.
- Dutoit, Thierry. "High-quality text-to-speech synthesis: An overview." Journal Of Electrical And Electronics Engineering Australia 17.1 (1997): 25-36.