SnapYummy

Intelligent Cooking Assistant

Intelligent Reasoning Systems

Practice Module

Semester One 2021/2022

PREPARED BY:

Student Name	Student ID
Chen Ziqi	A0231425R
Li Peifeng	A0231447J
Mediana	A0231458E
Xu Xuanbo	A0231436M

Contents

1 Executive Summary

Due to Covid19 that occurred since December 2019, a significant proportion of the world including Singapore have been changing the way they lived. Working from home, minimizing the interaction with people by keeping a safe distance and home-dining has become the norm. With people spending more time cooking at home, there has been an increased demand on cooking recipe and instructions.

Our team proposed SnapYummy, an Intelligent Cooking Assistant that can detect the ingredients available in the refrigerator, and then present to users all the available recipes that are suitable for users to cook, and according to their preferences. It interacts with users via Telegram, a widely used Messaging Platform, and shows the recipe images, instructions, ingredients, and cooking time for users as references.

The use of YOLOv5 for object detection and Neo4j knowledge graph for searching data are few of the key highlights of our Intelligent Cooking Assistant. With the YOLOv5 technology and Neo4j Knowledge Graph incorporated, it can recognize multiple food items in the refrigerator and search for the recipes in a faster manner. Besides, Intelligent Cooking Assistant also include DialogFlow for users' intent detection. This allows our users to save time and focus on the cooking, instead of the preparation work like input the available ingredients or visiting recipe websites for searching cooking instructions.

This Intelligent Cooking Assistant is targeting on users who need to:

- Use up most of the food items or leftover ingredients in the refrigerator,
- Plan for meal according to different preferences like cuisine type or dietary preferences,
- Learn or improve cooking skills,
- Spending less time in meal preparation and save time in the kitchen,
- Come out with new recipe ideas, new inspiration of cooking and eating.

To date, our Intelligent Cooking Assistant are based on the recipes shared by community. In the future, we are looking to collaborate with famous cooking chef to come out with secret recipe. Users who have subscribed to premium service shall be able to subscribe to secret recipe from famous cooking chef.

2 Market Research

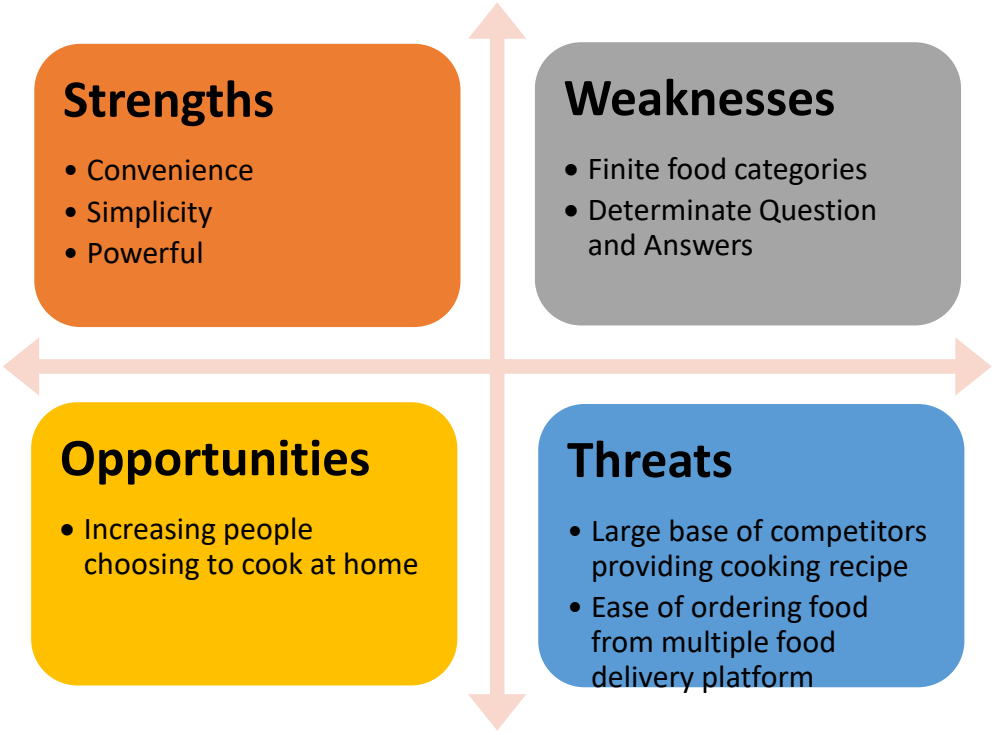

Figure 1 SWOT Analysis

In the context of the Covid-19 pandemic, home cooking has become one of the new normal that people do. When the authorities imposed the number of dine-in people, people feel unsafe to eat-out and opt to cook at home. According to Acosta Report[1], above half of customers (55%) said that they are more willing to eat at home instead of dining out during the pandemic. Another earlier research conducted by Bloomberg News and Morning Consult[2] reported similar result. Nearly one third of respondents said their frequency of home cooking will be increased once the staying home policy is lifted, while only 7% of them will cook less after economy reopens.

Cooking makes people feel more relax and even happier. It helps people free from heavy workload by temporally diverting their attention on a small but creative thing. According to a psychology study[3], creative activity like cooking will activate human's positive affect, which provides a sense of accomplishment and emotional grounding of a task. Therefore, kitchen is usually regarded as a place to destress, not just to cook and dine. A survey conducted based on 2,000 UK homeowners reports that about 60% of people believe cooking in the kitchen is a leisure time for them.

[1] https://www.acosta.com/news/new-acosta-report-details-how-covid-19-is-reinventing-how-america-eats
[2] https://www.bloomberg.com/news/articles/2020-07-07/newly-minted-home-chefs-mark-another-blow-to-u-s-restaurants
[3] https://www.tandfonline.com/doi/abs/10.1080/17439760.2016.1257049

Hunter's consumer survey[4] found that 54% of consumers report cooking more and 46% report baking more in America. There are even 51% consumers who thought they would continue to cook at home after the world returns to a new normal. Affected by this trend, The recipe application market, which relies on application traffic for commercial realization, will witness a new round of growth.

In this context, we launched a lightweight intelligent cooking assistant that can perceive the food in the user's refrigerator with one picture and intelligently arrange recipes for the user through dialogue. The SWOT analysis is as follows.

Strength:

- Convenience: It can automatically detect the food in the refrigerator with a picture of the refrigerator.
- Powerful: It can recommend recipes that best fit the food in the refrigerator and retrieve recipes for foods with similar flavours.
- Simplicity: It interacts with users through dialogue. And users can easily obtain services through telegram.

Weakness:

- The food categories that can be identified are limited: The categories of food that can be identified are currently limited but can be further increased.
- The questions that can be answered are limited: The question templates that can be answered are limited but can be further increased.

Opportunity:

- The number of people choosing to cook at home is increasing: Due to the Covid epidemic, more and more people choose to cook at home.
- People are becoming "lazy": With the acceleration of the pace of life and the popularization of smart applications, people are getting used to simple and easy-to-use things.

Threat:

- With large base of competitors who also offer cooking recipe and instructions, it is hard to get into market.
- Ease of ordering food from food delivery platform and promotion could hold back users from cooking at home and opt for food delivery.

[4] https://www.hunterpr.com/foodstudy_coronavirus/

3 System Architecture and Design

The architecture of the system is shown below.

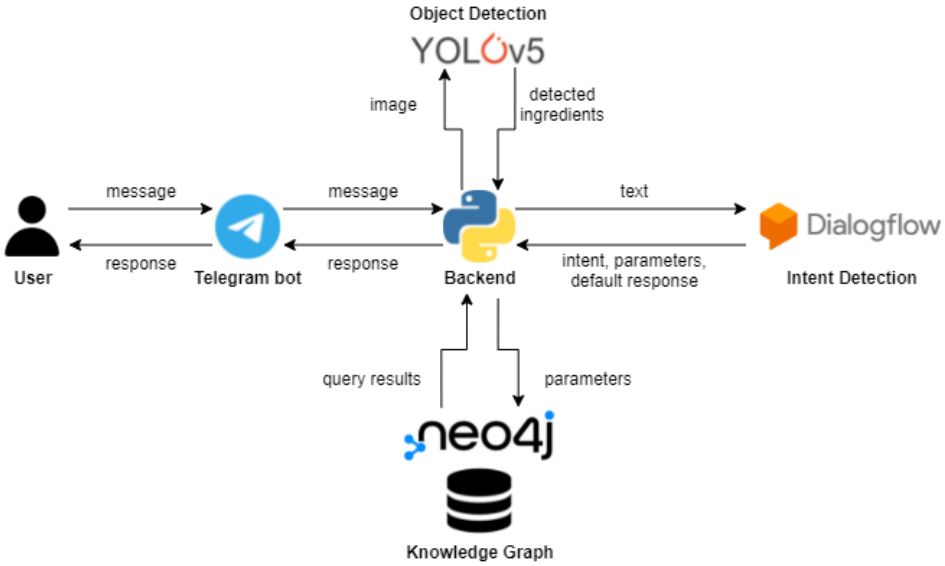

Figure 2 The architecture of the system

Followings are general functions of components:

Telegram Bot

- Send user messages to the Backend and return received responses to the user.
- Provide UI for interaction.

Backend (Python)

- Classify the format of user messages (text or image).
- Send image messages to the Object Detection part to recognize cooking ingredients.
- Send text messages / detected ingredients (as texts) to the Intent Detection part.
- Classify the intent in returns from the Intent Detection part.
- For query intents, use parameters to retrieve recipes from the Knowledge Graph.
- Return default responses / query results to the Telegram Bot.

Object Detection (YOLOv5)

- Apply object detection on user images and return detected ingredients to the Backend as texts.

Intent Detection (Google Dialogflow)

- Detect the intent of received texts.
- Extract parameters (e.g., cooking ingredients, preferred cuisine and dietary) from slots in received texts.

- Control the conversation flow based on the detected intent and extracted parameters.
- Return the detected intent, extracted parameters with certain default response to the Backend.

Knowledge Graph (Neo4j)

- Store entities of recipes, cooking ingredients, cuisines, dietaries, as well as relations between entities by graph.
- Support query and return retrieval results to the Backend.

4 Pre-Deployment

4.1 Data Gathering and Pre-processing

4.1.1 Recipe Dataset

Two datasets related to recipe are used in this project. Both datasets are sourced from Kaggle. The first dataset - Food.com[5] is used to build the recipe knowledge graph. It contains 522,517 recipes and 12,485 ingredients. Table 1 shows the main attributes of the Food.com dataset.

Food.com Dataset to build Recipe Knowledge Graph	
Attribute	**Description**
RecipeId	Unique ID for every recipe
Name	Recipe name
TotalTime	Total time needed to spend on this recipe
Description	Main Description of recipe
Images	Recipe image
Keywords	A list of keywords could represent this recipe most. e.g. [Indian, vegan, breakfast....]
RecipeIngredientParts	A list of ingredient needed in this dishes. e.g. [apple, banana, pork....]
RecipeIngredientQuantities	The corresponding quantities of the ingredient
RecipeInstructions	The instruction of the recipe
AggregatedRating	Average rating of this recipe

Table 1 Food.com Dataset

The second dataset - What's Cooking[6] is used as the training data for predicting cuisine type. It contains 39774 training samples. Table 2 shows the main attributes of What's Cooking dataset. The first dataset from Food.com are used as new data and input to trained model for predicting the cuisine type.

What's Cooking Dataset for Predicting Cuisine Type	
Attribute	**Description**
RecipeID	Unique ID for every recipe
Ingredients	A list of ingredients needed in this dishes. e.g. [apple, banana, pork....]

[5] https://www.kaggle.com/irkaal/foodcom-recipes-and-reviews
[6] https://www.kaggle.com/c/whats-cooking

Cuisine type	The cuisine type of this recipe. E.g. Chinese

Table 2 What's Cooking Dataset

4.1.2 Images Dataset for Object Detection

The object detection part aims to recognize different kinds of cooking ingredients from a photo of a fridge, so 43 kinds of ingredients which can be commonly seen in the fridge were selected as our classification categories. Because it's hard to find a satisfying multi-object dataset containing all the 43 categories, a self-built dataset was used for model training.

For 28 of the 43 categories, data from Open Images Dataset were used. For the remaining 15 categories, images were collected from Google and manually labelled by the group. All the images are 24-bit RGB color image in JPG or PNG format, and in various sizes. All the labels are TXT files in which information about bounding boxes in corresponding images is stored in 'encoded category label, center_x, center_y, width, height' format.

After data collection, several self-developed scripts were used to re-encode all category labels, for category labels of data from two sources were previously encoded in different ways. The merged dataset totally contains 1,961 images with 5,379 instances. The category contains most instances (351) is tomato and the one contains least (23) is peas.

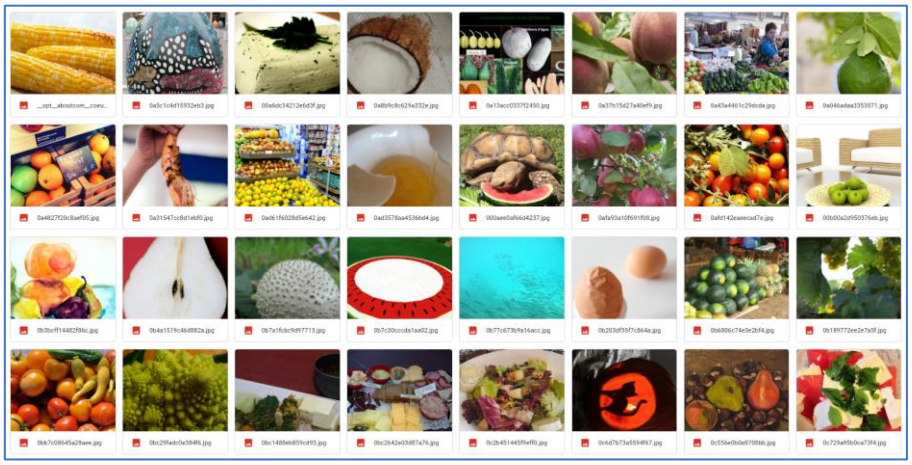

Figure 3 some samples from the self-built dataset

Considering the actual use scene, a dataset contains images with different background might be not enough for model's good performance. For this reason, 20 images of fridge were further collected and manually labelled to create a smaller dataset for fine-tuning. This second dataset totally contains 217 instances of all the 43 categories. Images and labels are in the same settings as the previous large dataset.

5 System Implementation

5.1 Object Detection (Image Vision)

5.1.1 Why YOLO

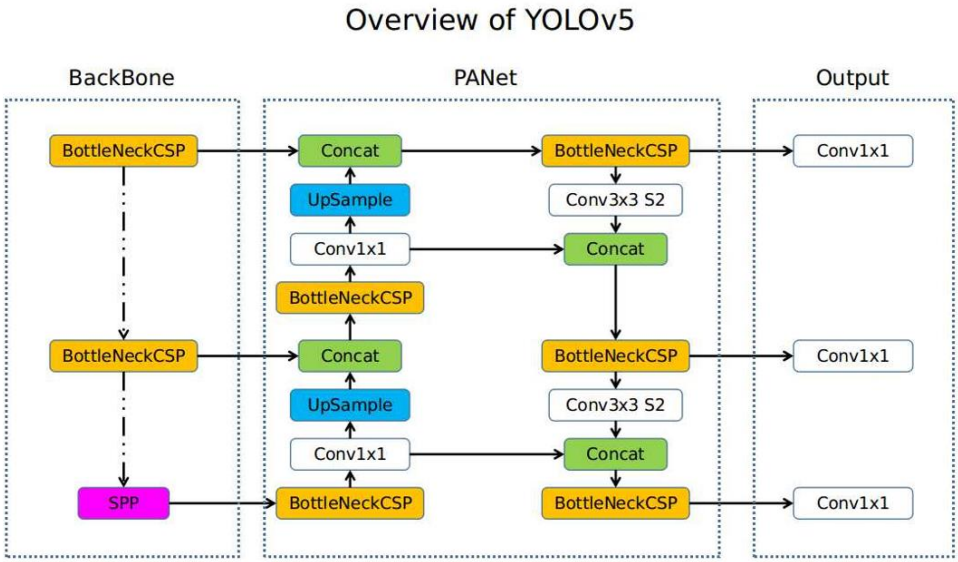

Figure 4 Yolo Overview[7]

Locating and identifying food items in the refrigerator are one of the main tasks in this project. With the object detection capability, users do not need to input the ingredients to Telegram. Instead, users may take a picture and upload to Telegram for detection with ease. Yolov5 is chosen as the object detection model with following reasons:

5.1.1.1 Speed

YOLOv5 is fast. The speed of inference will directly influence user's experience on our product. YOLO is designed as a one-stage flow and predict both class labels and bounding boxes simultaneously, which is 10 times faster than mainstream two-stages object detection models, e.g Fast R-CNN and Faster R-CNN.

5.1.1.2 Capable of Capturing Global Information

YOLO is naturally capable to extract features from the whole image. Unlike two-stages model, when it can only obtain the features (see) a region of the image. In the case of refrigerator, people always put the same kind of food together, for example eggs are placed together. This clustering effect can improve the recall of YOLO model.

[7] https://github.com/ultralytics/yolov5/issues/280

5.1.1.3 Generalization Capabilities

YOLO performs better in generalization. As a universal object detection model, it has been approved to perform well in images from other domains like artworks. Considering that our limited computing resources, we choose to continue training on a pre-train YOLOv5 model and further fine-tune on our refrigerator dataset.

5.1.2 YOLO Training

Two-phases training are conducted for YOLOv5. We used the official pre-trained model and continued training it on a large dataset; later, we further fine-tuned the model in a small refrigerator dataset. The logic behind this design is to improve the robustness of our model. The large dataset helps YOLO learn various angles and shapes of the food in different background settings, while the small refrigerator dataset further guides the YOLO model to learn some spatial information in refrigerator (e.g items are arranged in rows and the same kind of food cluster together).

We trained the YOLOv5 on large dataset for 80 epochs and fine-tuned on small dataset for 300 epochs. The model was implemented by PyTorch framework and trained on Google Colab GPU Tesla K80. The training results are shown in following figures.

As shown in Figure 5, the model starts to converge in 60th epoch, and reaches around 0.5 on Mean Average Precision(mAP@0.5) score.

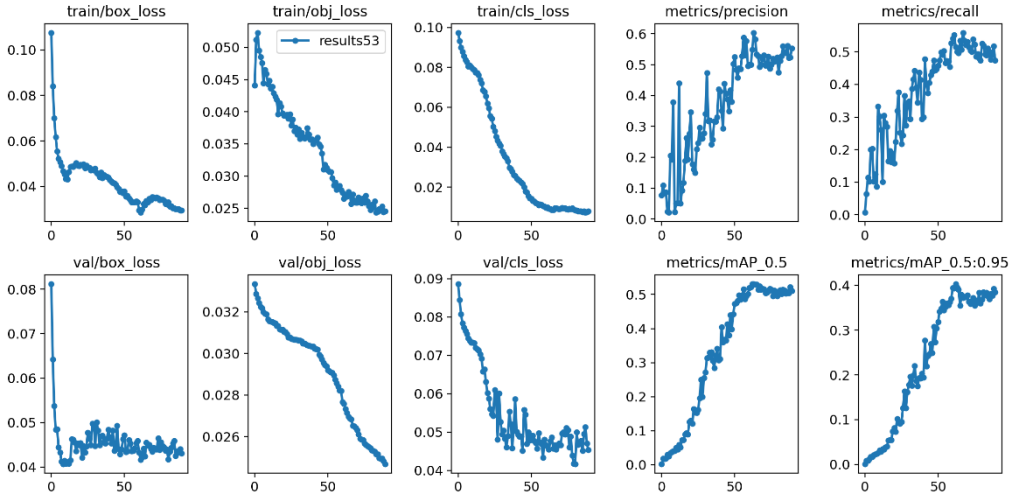

Figure 5 Training details for big dataset

In order to fine tune the YOLOv5 model in the refrigerator scenario, we manually built a small dataset consisting of food in the refrigerator. We use makesense.ai[8] tool to label the object and their bounding boxes. Figure 6 show samples of the dataset.

Figure 6 Samples of refrigerator dataset

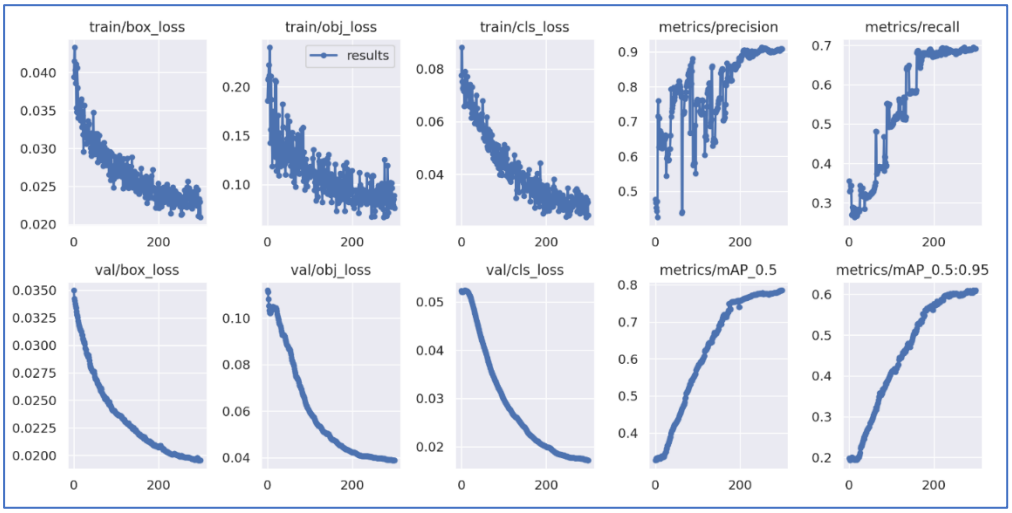

Figure 7 Training details for refrigerator dataset

[8] https://www.makesense.ai/

Figure 7 illustrates the curves in fine tune training process. Model starts to converge on 250th epoch and reaches 0.78 on mean average precision (mAP@0.5) score.

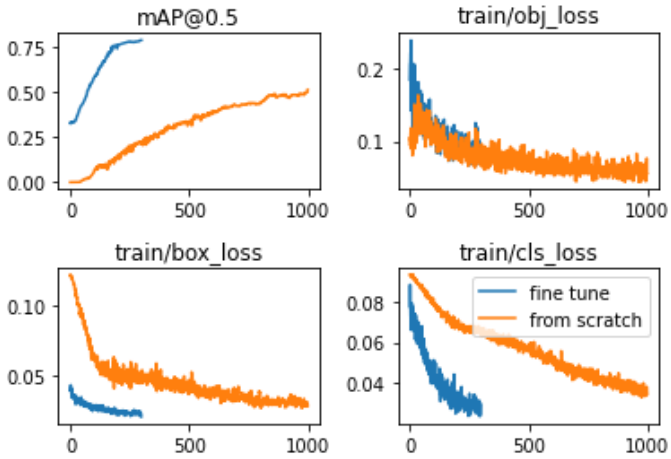

Figure 8 Comparison between two-phases training strategy and training on official pretrain model

To show the necessity of our two-phases training strategy, a comparison experiment has been done. In Figure 8, the orange line is a model trained on refrigerator dataset using official pretrain model, while the blue line shows the fine tune model fine-tuned by 300 epochs. The figure shows that it is useful to train YOLOv5 on a large dataset, it boosts the speed of convergence in fine tuning phase.

5.2 Knowledge Engineering

Our application is expected to accurately answer users' complex questions, which requires it to have domain knowledge in related fields. Therefore, knowledge engineering is applied in this part. Firstly, knowledge acquisition methods are applied to extract the relevant knowledge of recipes and ingredients in the data set. Then a knowledge graph is conducted for knowledge representation. Finally, some simple reasonings are done to complete knowledge discovery.

5.2.1 Entity Modelling Design

To enable the knowledge base to cover most of the questions raised by users, the following knowledge entities and attributes is needed based on our investigations. Figure 9 shows the entity frames system. The Recipe entities would contain most of the information that users would query like instruction and cooking time while ingredient entities are required to find the recipes that could better make use of foods in users' refrigerator. And with those label entities, we could do some customized recommendation to users.

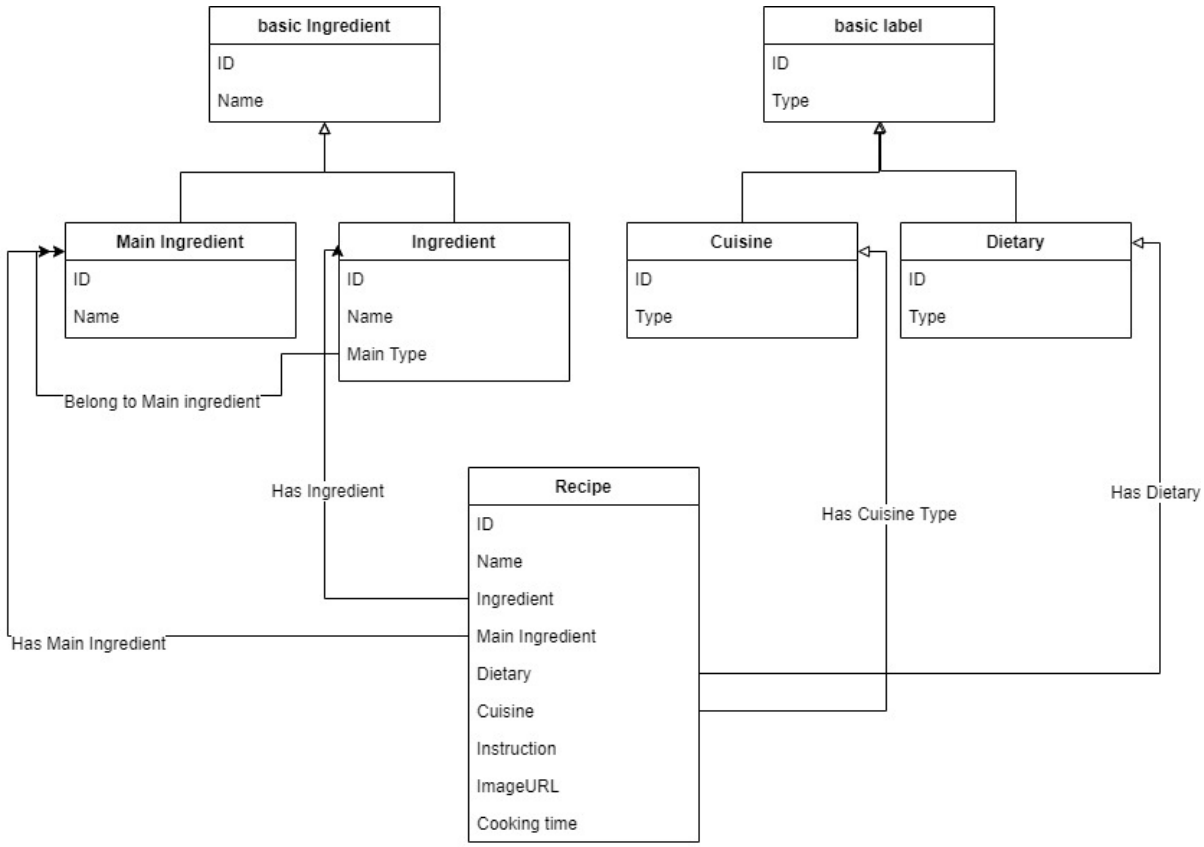

Figure 9 knowledge entities and attributes needed by the application

5.2.2 Knowledge Acquisition

In this part, the knowledge required by the above entities is extracted from the dataset. In addition to the knowledge that can be directly extracted from the data set, there is also some knowledge could not be directly extracted which is unstructured or implicit. Therefore, some learning based methods are applied to acquire this knowledge.

5.2.2.1 Recipe Label Acquisition

Cuisine and Dietary are two important attributes that are not provided by the dataset directly, which are expected to be acquired in the following two steps.

5.2.2.1.1 Word2Vec + K-Means Based method to cluster keywords

Although Cuisine and Dietary are not provided by the dataset directly, the dataset includes “keywords” columns that contains relevant keywords like “Chinese” or “Breakfast”. To extract these related keywords from many redundant keywords which aren’t needed, a Word2Vec + K-Means model is applied to cluster keywords. Word2Vec is a model that converts words into vector form. Through conversion, the processing of text can be simplified to vector operations in the vector space. The K-means clustering

algorithm is an iterative clustering analysis algorithm. The cluster centroid and the samples assigned to them together represent a cluster.

The Word2Vec + K-Means model automatically divides all keywords into nine different categories, namely: Cuisine, fruits or vegetables, meat, dietary, type, required time, snacks or sauce, cook machine, and main food. In this way, keywords related to dietary or cuisine are extracted and they are used as the values of these two attributes. The steps of Word2Vec + K-Means are as follows: 1) Load and finetune language model, 2) Get phrase embedding or word embedding, and 3) Use K-Means for guided clustering. The visualization results of keyword clustering are as follows (T-SNE result):

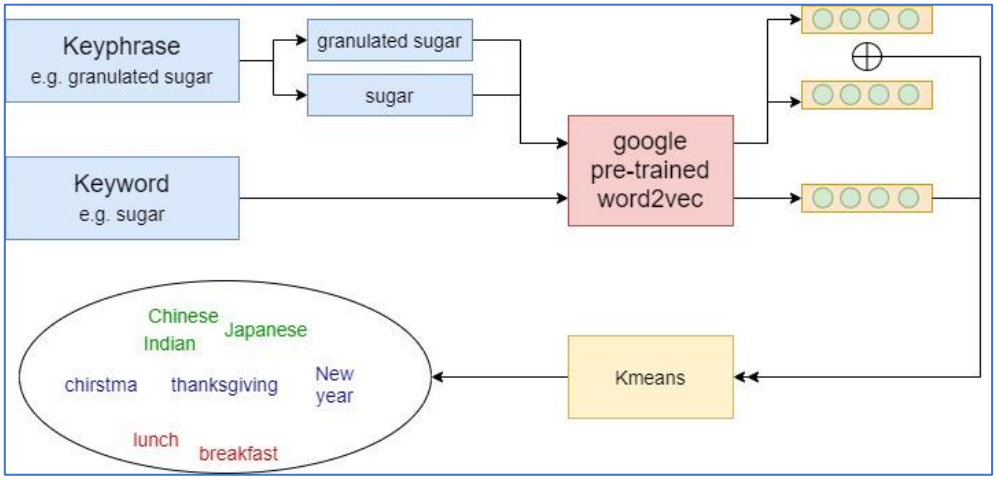

Figure 10 Word2Vec + K-Means

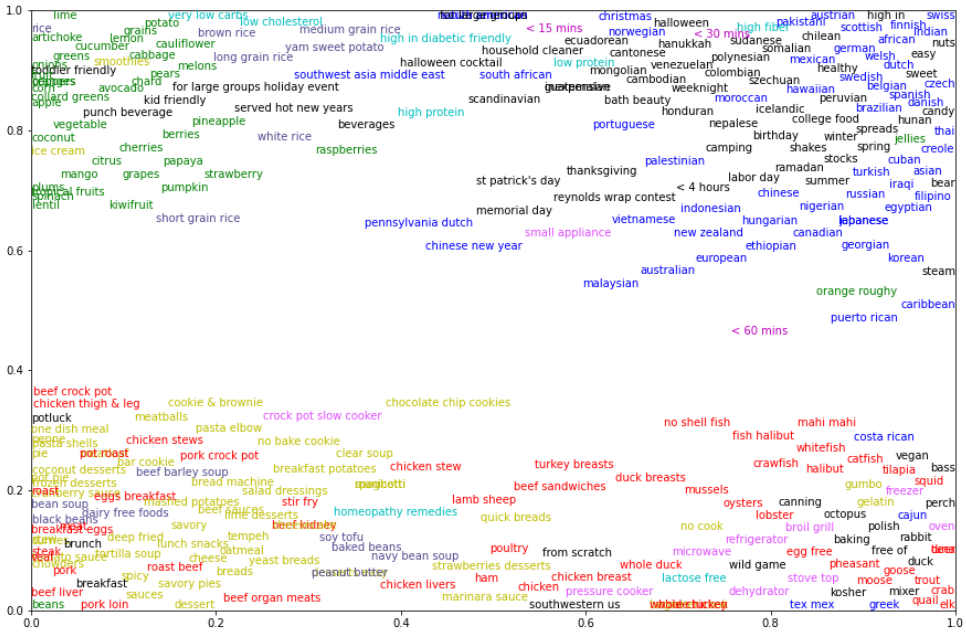

Figure 11 keyword T-SNE clustering result

5.2.2.1.2 Embedding-based CNN– Cuisine Type Prediction

Through the word2vec+Kmeans model in section 6.2.2.1.1, 40% of the recipes obtained the cuisine keywords that carry the cuisine type information while there are still 60% of the recipes that lack the cuisine type. An Embedding-based CNN is applied to predict the cuisine type of those recipes missing the cuisine related keyword.

To solve the problem of large and sparse feature space, we introduced a deep learning model and used the ingredient embedding layer as a feature encoder, which is learned from the NLP field. Specifically, we first trained an Ingre2Vec model and obtained the dense vectors of the ingredients of a certain recipe through this Ingre2Vec model. These vectors are then stacked as the input of the CNN model.

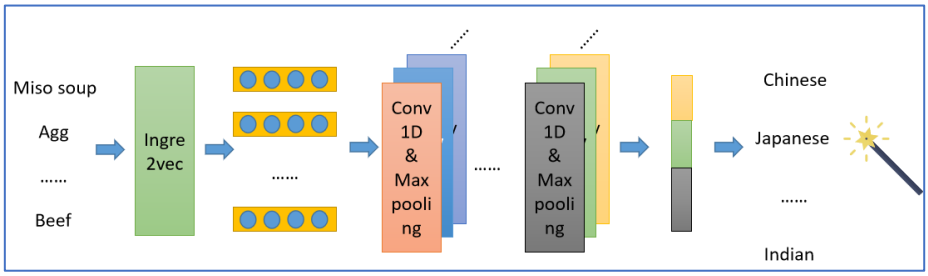

Figure 12: Embedding-based CNN

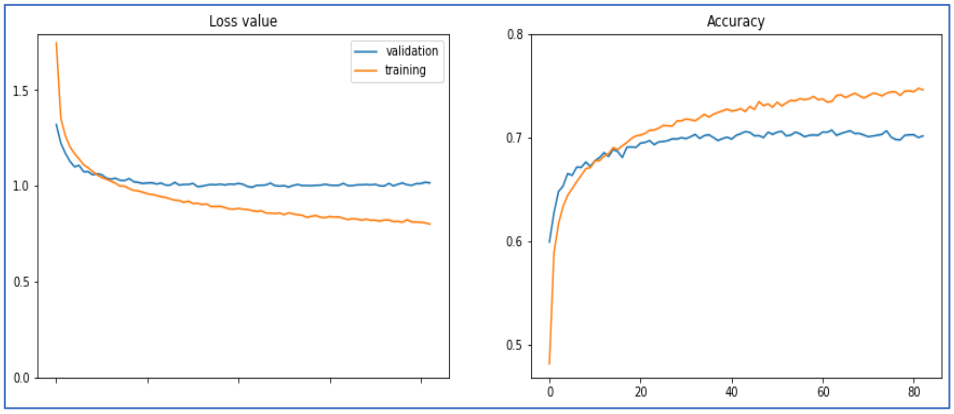

Figure 13 : Loss and accuracy of the Embedding-based CNN model

5.2.2.2 Main ingredient Entity acquisition

Although millions of recipes are included in the Knowledge Base, it is still not easy to find a recipe that can accurately match all the ingredients given by the user because of the variety of ingredients. Therefore, ingredients with similar flavors can be classified into a main ingredient category for further recommendation. Inspired by the word2vec language model, an ingre2vec model is trained by the open-source word2vec tool called genism to find ingredients with the similar taste. The entire model is a shallow DNN

network. The actual task of this model is to predict the missing ingredients given several ingredients in a recipe.

After the Ingre2vec model built, 57 main ingredients are manually selected and then the similarity between each ingredient and these 57 main ingredients are calculated to divided them into the corresponding main ingredient categories. Please refer to main ingredient Appendix for details. To better achieve the main ingredient alignment, four different similarity calculation methods are design. The experiments show that the fourth method has the best result:

Similarity calculation	Main insight	Silhouette Coefficient
$\cos\left(\sum u, \sum v\right)$	Use the sum of single word embedding to represent phrase embedding and calculate cosine similarity	0.419901161
$mean\left(\sum_{i=0}^{n}\sum_{j=0}^{m}\cos(u_i, v_j)\right)$	Calculate the cosine similarity of all words of two phrases and find the average value	0.388933544
$max\left(\sum_{i=0}^{n}\sum_{j=0}^{m}\cos(u_i, v_j)\right)$	Calculate the cosine similarity of all words of two phrases and find the maximum value	0.49546796
$\cos\left(\sum_{i=0}^{n} w_i u_i, \sum_{j=0}^{m} w_j v_j\right)$	Use the weighted sum of the single word embedding to represent the phrase embedding and calculate the cosine similarity. The closer to the end of the phrase, the greater the weight of the word.	0.544087151

Table 3: Similarities experiment

Where u represents the single word embedding of the ingredient (an ingredient usually consists of multiple single words, which is a phrase), v represents the single word embedding of the main ingredient, and w is a coefficient that linearly increases as i increases

5.2.3 Knowledge Representation using Knowledge Graph

The knowledge graph is a semantic representation of the real world, where each node represents the edge of the entity connecting the nodes and corresponds to the relationship between the entities. The knowledge graph is very suitable for integrating unstructured data to discover knowledge from scattered data. Therefore, knowledge graph is chosen as the knowledge representation of this project.

Neo4j is a graph database that uses the attribute graph model to model data. It can traverse nodes and edges at the same speed, in which the traversal speed is not limited by the amount

of data of graph. Due to Neo4j's high speed performance, reliability, scalability, and ecological integrity, it was chosen as the graph database to store the knowledge graph for this project.

The established knowledge graph is as follows:

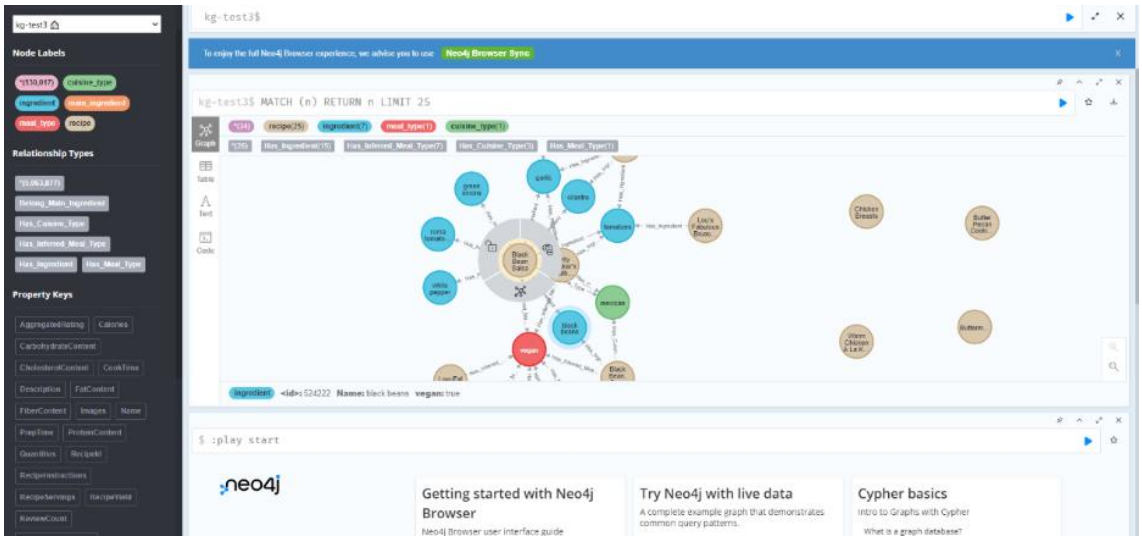

Figure 14 Constructed Knowledge Graph Shown in Neo4J

The types and number of nodes and relationships in our knowledge graph are shown in the following table:

Name	Type	Amount
Recipe	Node	522517
Ingredient	Node	7368
Main ingredient	Node	57
Cuisine type	Node	65
Meal type	Node	10
Has_Ingredient	Relationship	4102929
Has_Cuisine_Type	Relationship	522517
Belong_Main_Ingredient	Relationship	2650
Has_Meal_Type	Relationship	278226
Has_Inferred_Meal_Type	Relationship	157555

Table 4: Knowledge Graph description

5.2.4 Knowledge discovery using rule-based reasoning

After the knowledge graph is established, some knowledge can be discovered through rule-based reasoning.

5.2.4.1 Main ingredient and recipes connection

After main ingredients clustering, we created an "Belong_main_ingredient" relationship between ingredients and their corresponding main ingredients. Based on the rule, we can create new relationship "is_main_ingredient" linking the recipes and main ingredient together.

Rule 1: If a recipe *R* has ingredient *A* and *A* is belong to main ingredient *AA*, then *AA* is a main ingredient of *R*.

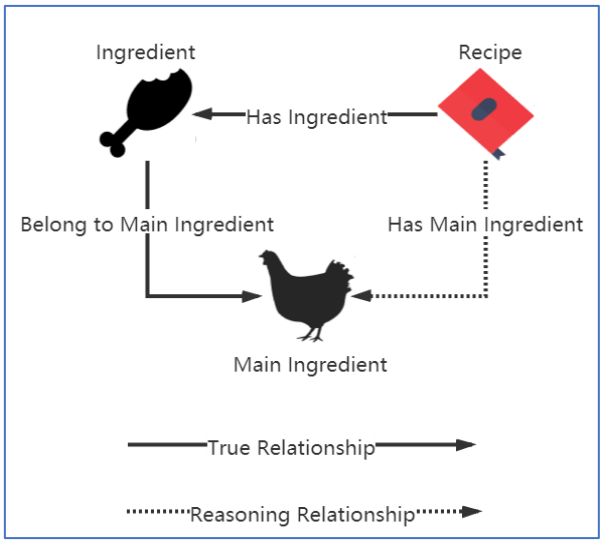

Figure 15 main ingredient and recipes connection rule

5.2.4.2 Vegan label

There are many missing vegan tags in the data set, but we can fill in these missing with the existing vegan tags.

Rule 2: If the label of a recipe is vegan, then all the ingredients in the recipe are vegetarian.

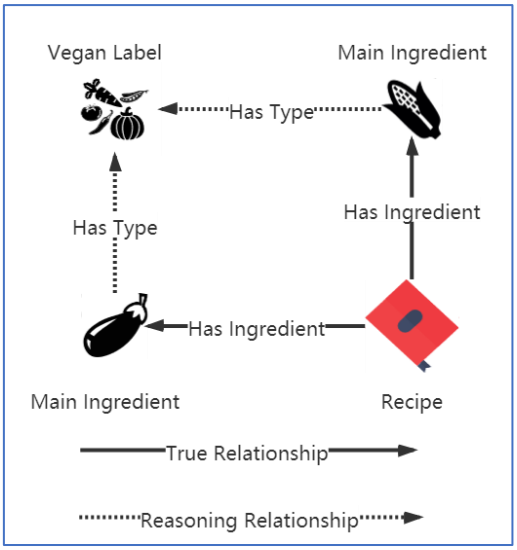

Figure 16 Vegan label rule 1

Rule 3: If all the ingredients in a recipe are vegetarian, then the label of the recipe is vegan.

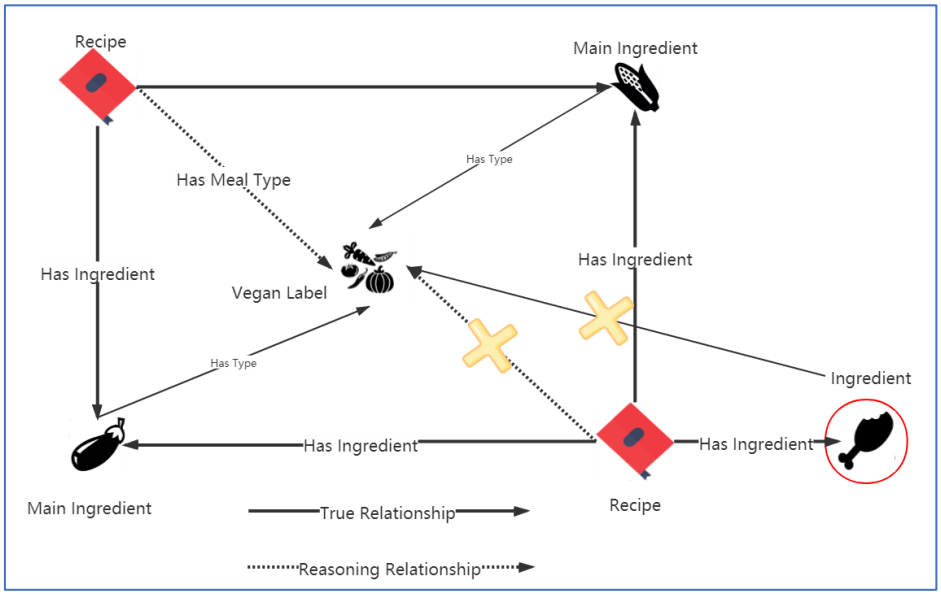

Figure 17 Vegan label rule 2

5.2.4.3 Halal label

There is no Halal tag in the data set. But we often need to meet the user's query requirements for Halal recipes in the usage scenario, so we fill in the Halal tag.

Rule 4: Recipes whose Main Ingredient category is Pork are labeled as non-Halal.

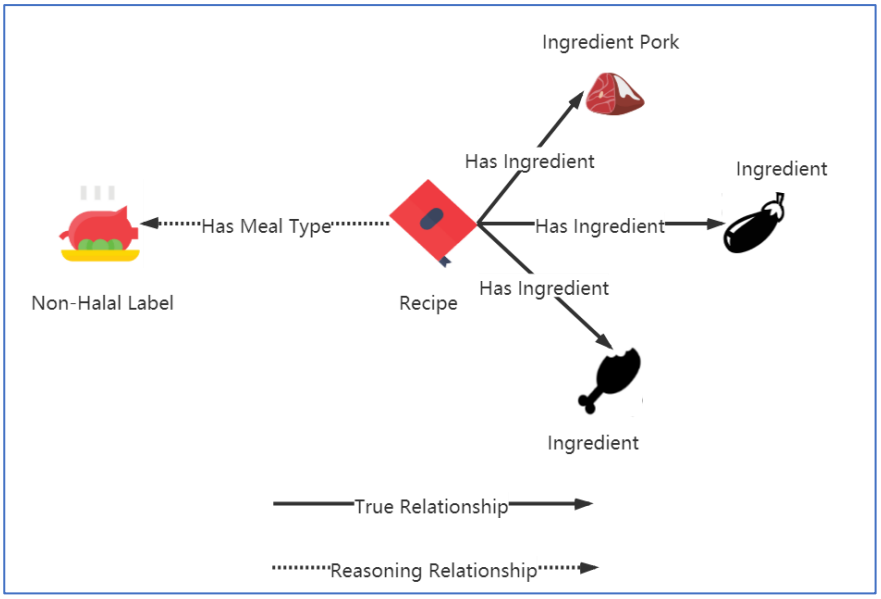

Figure 18 Halal label rule

5.3 Intent Detection

5.3.1 Google Dialogflow Agent

Dialogflow is a Google's bot platform that work on natural language processing. In this project, a Dialogflow agent "RecipeAgent" is created to handle the conversations with end-users. Multiple phrases that are expected from users are input into Dialogflow Training Phrases to pre-train the agent. Upon training the agent, the Dialogflow can then detect and match user's intent. Depending on the intent detected, it will extract entities such as cuisine, dietary, ingredients and recipe name accordingly, and map them to the appropriate agent action like text response.

5.3.2 Intents, Contexts, and Parameters

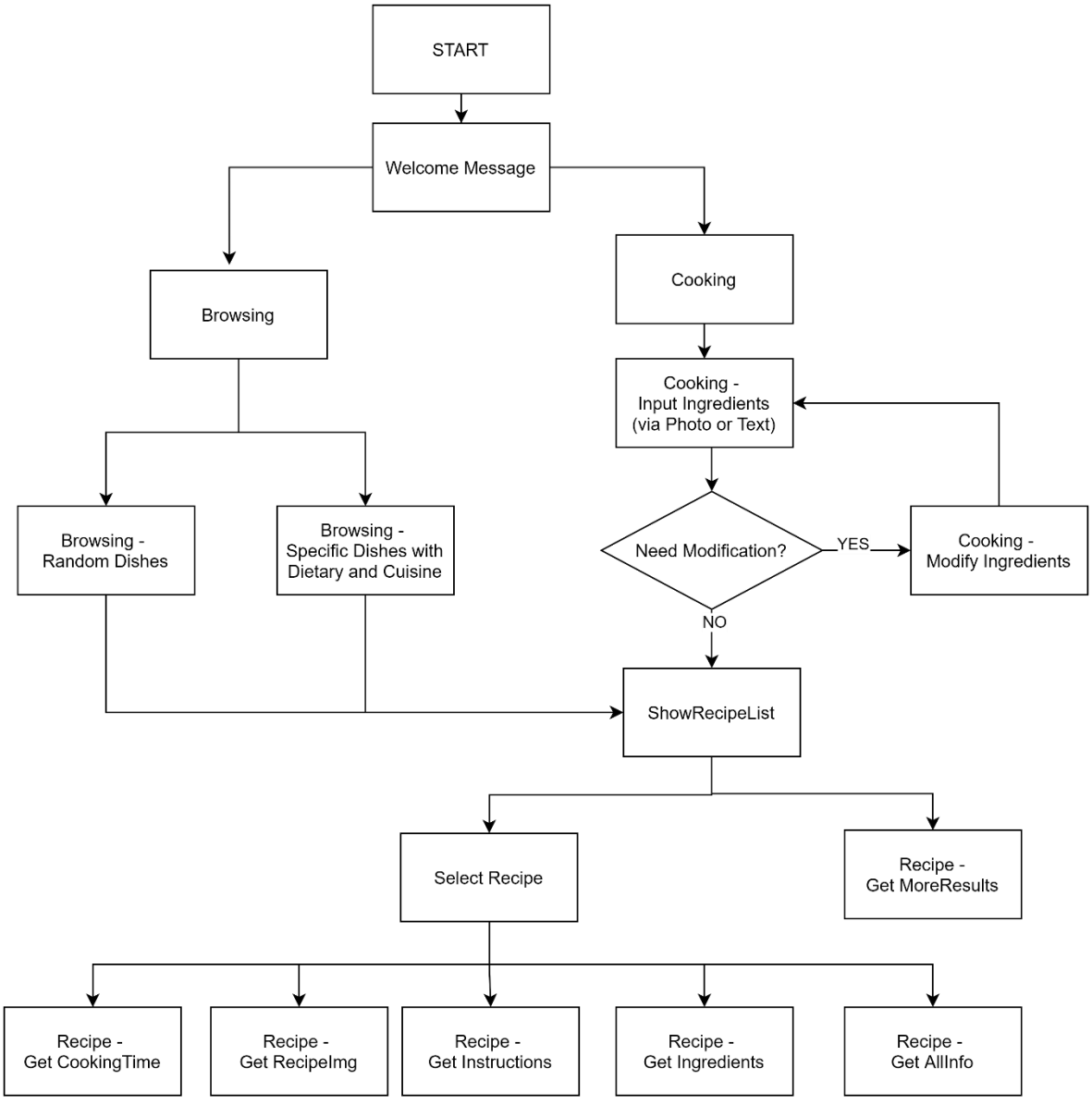

Figure 19 Google DialogFlow Intent

Following are the intent created in Dialogflow:

5.3.2.1 Intent: Browsing

When users type “recipe” or “browsing”, Dialogflow matches it to the “Browsing” intent, and response with question whether users would like to have random recipe or specific recipe. “Browsing-followup” contexts is also activated to carry the required information to other intents and link the different intents together.

Intent: Browsing	Remark
Output Contexts	• Browsing-followup
Training Phrases	• Recipe • Browsing
Text Responses	• We can offer random dishes that will surprise you or specific dishes based on your dietary constraints or preferred cuisine. Do you like random dishes or specific dishes? • Would you like to show random dishes or specific dishes?

5.3.2.1.1 Intent: Browsing – Random Recipe

“Browsing – Random Recipe” intent triggered when users type “anything”, “surprise me”, “give me random dishes” or “random”. In this intent, “browsing-followup” context inherited from the browsing intent is deactivated, while “browsing-randomdishes-followup” and “awaiting_recipename” are activated.

Intent: Browsing – Random Recipe	Remark
Output Contexts	• Browsing-randomdishes-followup, awaiting_recipename, • Browsing-followup* *(remove)*
Training Phrases	• Anything • Surprise me • Give me random dishes • random
Parameters	• Ingredients (type: list) • Cuisine

	• Dietary (type: list)

5.3.2.1.2 Intent: Browsing – Specific Dishes

"Browsing – specific dishes" intent triggered when users would like to browse for specific dishes according to their cuisines and dietary preferences. Users can trigger this intent by typing "specific", "I have preference" or any other phrases as showed in table below. This intent will activate "Browsing-specificdishes-followup" output contexts which will be passed over to "Browsing -Specific Dishes with Dietary_Cuisine" intent.

Intent: Browsing – Specific Dishes	Remark
Output Contexts	• Browsing-specificdishes-followup
Training Phrases	• Specific dishes • Set constraints and preference • I have preference • I have constraints • Set • Specified • Specific
Text Response	• I would show you specific dishes based on your preference. Please input your preferred cuisines and dietary constraints. • Leisure browsing for specific dishes. What are your preferred cuisines or any dietary constraints?

5.3.2.1.3 Intent: Browsing – Specific Dishes with Dietary_Cuisine

"Browsing-Specific Dishes with Dietary_Cuisine" intent is a follow-up intent from the "Browsing-Specific Dishes" intent. In this intent, agent will prompt users for their cuisine preference and dietary preference. Both parameters are required, and intent cannot be completed without these two parameters.

Intent: Browsing – Specific Dishes with Dietary_Cuisine	Remark
Intent Contexts	• Browsing-specificdishes-followup

Output Contexts	• Awaiting_recipename • Browsing-specificdishes-followup
Training Phrases	• No dietary • No cuisine • I like @cuisine food • I want to try @cuisine @dietary food • I like @cuisine @dietary • I prefer @cuisine @dietary • I am a @dietary lover
Parameters	• Cuisine (required) • Dietary (type: list, required) • Ingredients (type: list)

5.3.2.2 Intent: Cooking

When users type "cook" or "cooking", "Cooking" intent will be triggered and requesting users to input ingredients. Users can either upload their photo containing the ingredients they have or type in the ingredients in Telegram. This intent will activate "cooking_ingredients" output context.

Intent: Cooking	Remark
Output Contexts	• Cooking_ingredients
Training Phrases	• Cooking • Cook
Text Responses	• Please input your ingredients or upload an image

5.3.2.2.1 Intent: Cooking Ingredients Text

"Cooking ingredients text" intent is used to handle users' input of ingredients via text. It is a follow-up intent from "Cooking" intent, in which "cooking_ingredient" context is activated and awaiting users to input their ingredients. Upon input the ingredients, users will be prompted whether a modification is needed. In this case, "cookingingredientstext-followup" will be activated in output context and "Cooking_ingredient" context is deactivated and removed after exit from this intent.

Intent: Cooking Ingredients Text	Remark
Input Contexts	• Cooking_ingredients

Output Contexts	• Cookingingredientstext-followup • Cooking_ingredients* *(remove)*
Training Phrases	• @ingredients • Ingredients: @ingredients • I have @ingredients • These are my ingredients @ingredients
Parameters	• Ingredients (type: list)
Text Responses	• Your ingredients list: $ingredient. Need any modification? • Detected ingredients are $ingredients, would you like to make any modification?

5.3.2.2.2 Intent: Cooking Ingredients Text Modification (YES/NO)

"Cooking Ingredients Text Modification" is an intent to handle users request on modifying the ingredients that they have input. It is a follow-up intent from the previous intent in 6.3.2.2.1 "Cooking Ingredients Text" and contain two sub-intent "YES" and "NO". If user type "Yes" or using any list of "Yes" expression, then users can re-input the ingredients again. Otherwise, the agent will acknowledge that no modification is required and then response with the list of ingredients and ask users' cuisine and dietary preferences.

Intent: Cooking Ingredients Text Modification (YES)	**Remark**
Input Contexts	• Cookingingredientstext-followup
Output Contexts	• Cooking_ingredients
Training Phrases	• List of "Yes" expression
Text Responses	• Please input your ingredients again

Intent: Cooking Ingredients Text Modification (NO)	**Remark**
Input Contexts	• Cookingingredientstext-followup

Output Contexts	• cookingingredientstextmodify-no-followup
Training Phrases	• List of “No” expression
Parameters	• Ingredients (type: list) #cookingingredientstext-followup.ingredients
Text Responses	• These are your ingredients: $ingredients. What is your preferred cuisine and any dietary constraints?

5.3.2.3 Intent: Recipe

The recipe intents are used for handling all the recipe related request. Below is the request that the Intelligent Cooking Assistant is able to handle.

5.3.2.3.1 Intent: Recipe – ShowRecipeList

Showing all recipe list based on the ingredients that the users have input or image uploaded

Intent: Recipe – ShowRecipeList	Remark
Output Contexts	• Cooking_ingredients
Training Phrases	• Cooking • Cook
Text Responses	• Please input your ingredients or upload an image

5.3.2.3.2 Intent: Recipe – MoreResults

After showing the first ten recipes that match user’s requirement, user may request to showing the next ten recipes that matched if available or more results.

Intent: Recipe – MoreResults	Remark
Input Contexts	• Awaiting_recipename
Output Contexts	• Awaiting_recipename
Training Phrases	• More results • Tell me more • More info

Parameters	• Ingredients (type: list) Value: #awaiting_recipename.ingredients • Dietary (type: list) Value: #awaiting_recipename.dietary • Cuisine Value: #awaiting_recipename.cuisine

5.3.2.3.3 Intent: Recipe – GetInstructions

Getting the instructions of a recipe.

Intent: Recipe – Get Instructions	Remark
Input Contexts	• Recipe-followup
Output Contexts	• Awaiting_recipename
Training Phrases	• Instructions • How to cook • Get me the instructions • Show me the instructions
Parameters	• Recipename Value: #Recipe-followup.recipename

5.3.2.3.4 Intent: Recipe – GetCookingTime

Getting the cooking time of a recipe.

Intent: Recipe – GetCookingTime	Remark
Input Contexts	• Recipe-followup
Output Contexts	• Awaiting_recipename • Awaiting_cooktime
Training Phrases	• How much time needed • Preparation time • What is the cooking time for this dish • Cooking time
Parameters	• Recipename Value: #Recipe-followup.recipename

5.3.2.3.5 Intent: Recipe – GetAllInfo

Getting all information related to a recipe, including the instructions, images, ingredients and cooking time

Intent: Recipe – MoreResults	Remark
Input Contexts	• Recipe-followup
Output Contexts	• Awaiting_recipename
Training Phrases	• Show all • Show all info • All info • Show all information

5.3.2.3.6 Intent: Recipe – GetRecipeImg

Getting the image of a recipe.

Intent: Recipe – GetCookingTime	Remark
Input Contexts	• Recipe-followup
Output Contexts	• Awaiting_recipename • Awaiting_recipeimg
Training Phrases	• Recipe picture • Show me the recipe image
Parameters	• Recipename Value: #Recipe-followup.recipename

5.3.2.3.7 Intent: Recipe – GetIngredients

Getting all the related ingredients of a recipe.

Intent: Recipe – GetIngredients	Remark
Input Contexts	• Recipe-followup
Output Contexts	• Awaiting_recipename
Training Phrases	• Get ingredients • Show ingredients • Show me the ingredients • What are the ingredients

Parameters	• Recipename Value: #Recipe-followup.recipename

5.3.3 Entities

Three custom entities are created in Dialogflow for matching custom data related to this project. Following are the three custom entities type created:

5.3.3.1 Cuisine

Sixty over entries are created for cuisine entity. Refer to Cuisine Appendix for details.

5.3.3.2 Dietary

Five entries are created for dietary entity including vegetarian, halal, eggetarian, fruitarian and non-vegetarians. Each entry is defined with synonyms that will map to reference values. Refer to Dietary Appendix for details.

5.3.3.3 Ingredients

Forty over entries are created for ingredients entity. Most of the entries are the main ingredients in Knowledge Graph. Refer to Ingredients Appendix for details.

5.4 Backend Deployment

5.4.1 Overview

The structure of backend is shown in the figure below.

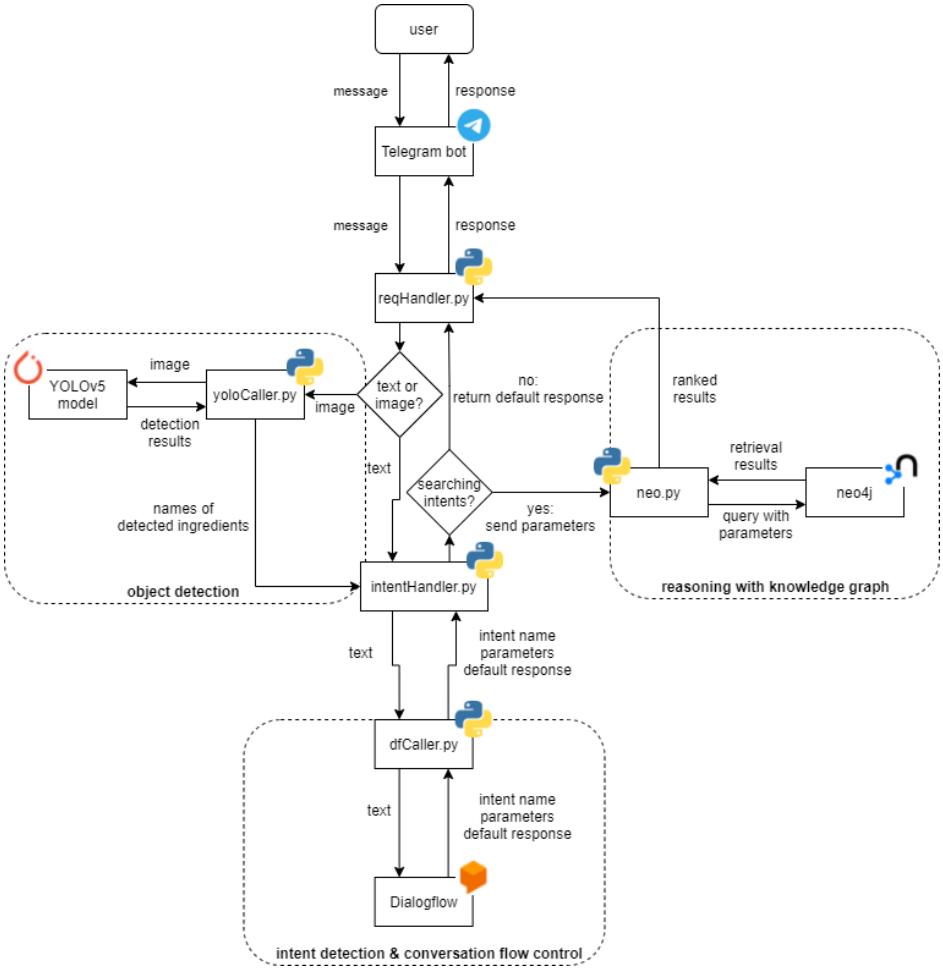

Figure 20 The structure of backend

Followings are the workflow of backend:

1. The user sends a message to the Telegram bot.
2. The message is passed to reqHandler.py and classified as a text or an image.
   a) For a text, it will be directly passed to intentHandler.py.
   b) For an image, it will be passed to yoloCaller.py and applied object detection on by the YOLOv5 model. Names of detected ingredients will be passed to intentHandler.py then re-formatted into a text message.
3. IntentHandler.py and dfCaller.py passed the text to the Dialogflow agent, and receive the detected intent, parameters (ingredients & preferences) and default response from it. A judgement on the intent will be done then.

a) For a searching intent, call neo.py to query recipes which meet the constraints of parameters in the Neo4j knowledge graph. Results returned will be further ranked and re-formatted, then passed to reqHandler.py as a response.

b) For other types of intent, directly passed the default response to reqHandler.py.

4. ReqHandler.py returns the response to the Telegram bot, and the bot send it to the user via the chat.

Additionally, it's obvious that our backend work between the Telegram bot and the Dialogflow agent. The reason why a Telegram + Dialogflow ensemble (a Dialogflow service, in which the Dialogflow agent directly links to the Telegram bot for intent detection) wasn't used is that it cannot handle image message.

Next are details about scripts integrating with different components (services).

5.4.2 Integrate with Telegram

The script reqHandler.py works for this part. It receives messages from a certain Telegram bot and judges whether messages are texts or images, after which texts will be sent to intentHandler.py, while images will be processed by the object detection module. It also receives processed feedback from intentHandler.py or reasoning module, re-formats them and returns them back to Telegram.

A Python library called telepot was used to interact with Telegram's APIs in the script.

5.4.3 Integrate with YOLO

The script yoloCaller.py works for this part. It receives images from reqHandler.py and put them into the YOLOv5 model for object detection. Names of detected ingredients will be sent to intentHandler.py as a list of strings for further procession.

Pytorch and official codes of YOLOv5 were used to create the model and apply detection.

5.4.4 Integrate with Neo4j

The recipes recommendation is based on knowledge graph stored in graph database Neo4j. The direct way to query on Neo4j database is using Cypher, an official database query language. In order to connect and query the database through python, the "pyneo" package was used. Based on our scenario demands, we designed 4 APIs to query on the Neo4j database.

5.4.4.1 Graph Structure Analysis

Before the introduction of our query APIs, we will take a look at our Neo4j graph database structure. Since the Neo4j provide user-friendly visual interface for the graph, we can easily and clearly see the structure of our KG.

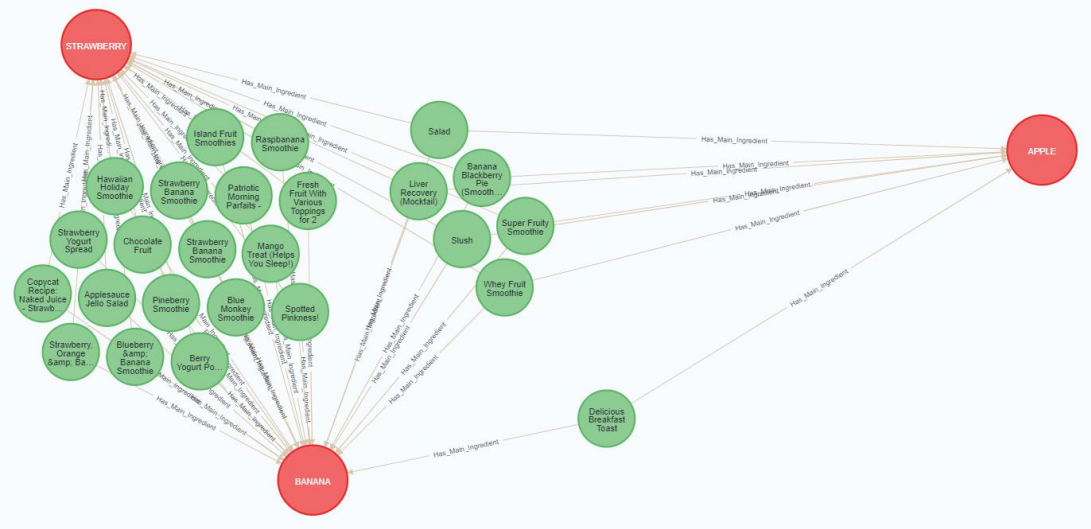

Figure 21 Relationship between recipes and main ingredients

Figure 21 shows the relationship between recipes (green nodes) and main ingredients (red nodes). Different recipes may share the same main ingredients.

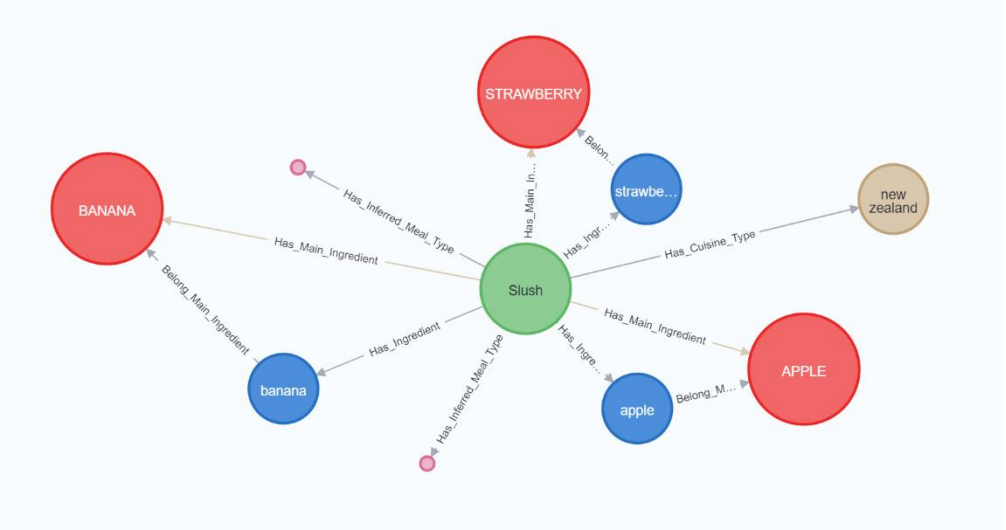

Figure 22 Overview of the relationship of recipes

Figure 22 illustrates all relationship among the recipes. Totally we build 4 kinds of entity connected to any recipe, including main ingredient, ingredient, meal type and cuisine type. As stated in previous section, the output labels of the YOLOv5 model form a subset of main ingredient. However, user can modify the output labels to their preferred one,

which will form a query including a mixture of main ingredient and ingredient. For example, ingredients recognized from YOLO part is ['apple', 'banana', 'strawberry'], user want to modify this result and he changes it to ['red apple', 'banana', 'strawberry']. In this case, 'red apple' is not a main ingredient but an ingredient and the query will contain both main ingredients and ingredients.

5.4.4.2 Get Recipes

This interface takes all of the ingredients that collected from user's input and return top K recipes which are most suitable for user's information. This interface supports two modes: with constraints / without constraints.

The whole algorithm procedure can be divided into 4 parts.

5.4.4.2.1 Ingredient separation

To handle the query including both main ingredients and ingredient, we iterate the query list and separate into two sets, since the query sentences for them are different.

5.4.4.2.2 Node ranking by community importance

Given a list of ingredients, we have multiple orders to put them into Cypher sentences, but which one is the best? Our solution is to consider the community importance of the ingredient nodes. We measure the importance of a node by counting its in-degree, as illustrated in the following formula:

$$Importance(Node_i) = \sum_j (I(i,j))$$

$$\begin{cases} I(i,j) = 1, Node_j \rightarrow Node_i \\ I(i,j) = 0, otherwise \end{cases}$$

An ingredient node with higher in-degree means there are more recipes contains it, and its influence in the whole knowledge graph is higher.

After computing the importance score of each ingredient, we sort them by their importance in reverse order and do the query.

5.4.4.2.3 Query without constraints

```
OPTIONAL MATCH ((rep:recipe)-[r:Has_Main_Ingredient]->
(i:main_ingredient{Name: 'Ingredient'}))                    ①
...

WITH rep, r, i,
size((rep:recipe)-[:Has_Ingredient]->(:ingredient)) as degree,
...                                                         ②

size((rep:recipe)-[:Has_Main_Ingredient]->
(:main_ingredient{Name: 'Ingredient'})) as minus_degree     ③


RETURN rep                                                  ④

ORDER BY degree-minus_degree * 2,
(case when minus_degree >= 1 then 1 else 0 end) desc,
degree                                                      ⑤

SKIP  0
LIMIT 25;                                                   ⑥
```

After re-ranking the ingredients list, we will query the database.

We organize the query sentence from six parts.

①. Take an optional matching for each main ingredient/ingredient in the sorted list. the variable 'rep' represents a subset of recipes including any ingredients in the list.

②. Calculate the number of ingredients in each recipe, named as "degree"

③. Count the number of certain kind ingredients in each recipe, named as "minus degree"

④. Return all recipes

⑤. Sort all the recipes by the difference between "degree" and "minus degree", because we want to match the recipes contain as less extra food as possible (users do not need to prepare extra food). The second key in the sorting ensure we should make full use of the given ingredients, i.e we match the recipes that include as more ingredients in the list as possible. The final key sort the recipes from smallest to largest number of ingredients.

⑥. Used for "more result" option. When user press "more result" button, we should recommend another list of recipes, which are different from the previous lists of recipes.

5.4.4.2.4 Query with constraints

Besides query only based on ingredients, sometimes user may have some preferences such as they love Chinese food, they are vegetarians, they are halal etc. We also support search with constraints in “Get Recipe” interface. We classify the constraints as two main categories, cuisine and dietary. If the “cuisine” or “dietary” parameters are assigned values, this mode will be activated.

For Cypher sentence, the “WHERE” clause can satisfy our demands. We use pre-defined “Has_cuisine_type” relationship to screen out the corresponding recipes, while the dietary part is more complicated. The following chart illustrates 5 types of dietary and their query methods.

Type of dietary	Query sentence
halal	MATCH (rep) WHERE rep.halal is null
vegetarian	MATCH (rep)-[rs:Has_Meal_Type]->(:meal_type{{Name: 'vegan'}}) WHERE (rep)-[:Has_Meal_Type]->(:meal_type{{Name: 'vegan'}}
fruitarian	MATCH (rep)-[:Has_Inferred_Meal_Type]->(:meal_type{Name:'fruit'})
eggetarian	MATCH (rep)-[:Has_Main_Ingredient]->(:main_ingredient{Name:'EGG'})

5.4.4.3 Get Recipes By Name

This interface takes the name of a recipe and search it in the graph database. The input is a string of recipe name and the output is an object of recipe or none (if no recipe found).

The query sentence is quite straight forward:

```
MATCH (rep:recipe)
WHERE rep.Name='Recipe Name'
RETURN rep
```

5.4.4.4 Get Ingredients

This interface is designed to query the ingredients for given recipe. To exactly find the recipe, we use recipe ID instead of recipe name as input. Since we have created relationship between recipe and ingredient, we can easily find the ingredients by relationship type “Has_ingredient”.

```
MATCH (rep:recipe)-[:Has_Ingredient]->(a:ingredient)
WHERE rep.RecipeId="ID"
RETURN a
```

5.4.4.5 Browser

Without ingredients, user can also take a look at our recipes and find something fresh and surprise. The browser interface is responsible for this demand. Similar with “Get Recipe” interface, the browser also supports two modes query – with constraint and without constraint.

For Cypher sentence, we use a random seed to randomly recommend recipes for users.

```
MATCH (a:recipe)
WITH rand() as r, a
MATCH (a) WHERE a.halal is null
MATCH (a)-[:Has_Inferred_Meal_Type]→(:meal_type{Name:'fruit'})
MATCH (a)-[:Has_Cuisine_Type]→(:cuisine_type{Name: 'chinese'})
RETURN a ORDER BY r LIMIT 10;
```

5.4.5 Integrate with Google Dialogflow

The scripts intentHandler.py and dfCaller.py work together for this part. They receive text messages from reqHandler.py, or lists of detected ingredients from the object detection module, and sent them to a certain Dialogflow agent for intent detection. The detected intent, along with parameters in slots and agent’s default responses will be returned from Dialogflow, after which intentHandler.py will further return them to reqHandler.py, or call reasoning module if the returned intent is about querying recipes.

The dialogflow library from google.cloud package was used to interact with the Dialogflow agent.

6 References

1. Bloomberg. (07 July, 2020). *Newly Minted Home Chefs Mark Another Blow to US Restaurants*. Retrieved from Bloomberg: https://www.bloomberg.com/news/articles/2020-07-07/newly-minted-home-chefs-mark-another-blow-to-u-s-restaurants

2. HunterPR. (2020). Food Study Coronavirus. Retrieved from HunterPR: https://www.hunterpr.com/foodstudy_coronavirus/

3. Irkaal. (2020). *Foodcom Recipes and Reviews*. Retrieved from Kaggle: https://www.kaggle.com/irkaal/foodcom-recipes-and-reviews

4. Kaggle. (2015). *What's Cooking | Kaggle*. Retrieved from Kaggle: https://www.kaggle.com/c/whats-cooking

5. *Make Sense*. (-). Retrieved from Make Sense: https://www.makesense.ai/

6. *New Acosta Report Details How COVID-19 Is Reinventing How America Eats*. (10 September, 2020). Retrieved from Acosta: https://www.acosta.com/news/new-acosta-report-details-how-covid-19-is-reinventing-how-america-eats

7. SeekFire. (2020). *Overview of model structure about YOLOv5*. Retrieved from Github: https://github.com/ultralytics/yolov5/issues/280

8. Tamlin S. Conner, C. G. (2016). Everyday creative activity as a path to flourishing. *The Journal of Positive Psychology* , 181-189.

Appendix A: Project Proposal

GRADUATE CERTIFICATE: Intelligent Reasoning Systems (IRS)

PRACTICE MODULE: Project Proposal

Date of proposal:

25 August 2021

Project Title:

ISS Project – SnapYummy, Intelligent Cooking Assistant

Sponsor/Client: *(Name, Address, Telephone No. and Contact Name)*

Institute of Systems Science (ISS) at 25 Heng Mui Keng Terrace, Singapore
NATIONAL UNIVERSITY OF SINGAPORE (NUS)
Contact: Mr. GU ZHAN / Lecturer & Consultant
Telephone No.: 65-6516 8021
Email: zhan.gu@nus.edu.sg

Background/Aims/Objectives:

During this Covid-19 pandemic, home cooking has become the new normal that people do. When the authorities imposed on the number of dine-in people, people opt to either home cooking or food delivery due to the feeling of unsafe going to restaurants and so forth. These culinary habits will likely continue even after the pandemic.

Our team proposed SnapYummy, an Intelligent Cooking Assistant that can detect the ingredients available in the refrigerator, and then present to users all the available recipes that are suitable for users to cook, and according to their preferences. It interacts with users via Telegram, a widely used Messaging Platform, and shows the recipe images, instructions, ingredients, and cooking time for users as references.

This Intelligent Cooking Assistant is targeting on users who need to:

- *Use up most of the food items or leftover ingredients in the refrigerator,*
- *Plan for meal according to different preferences like cuisine type or dietary preferences,*
- *Learn or improve cooking skills,*
- *Spending less time in meal preparation and save time in the kitchen,*

- *Come out with new recipe ideas, new inspiration of cooking and eating*

Requirements Overview:

- Research ability
- Programming ability
- System integration ability

Resource Identified:

1. Dataset for model training
    - Recipe Dataset – Kaggle Food.com and Kaggle What's Cooking
    - Ingredients Image Dataset for Yolov5 Training
2. Hardware
    - GPU*, Backend Server
3. Software
    - Chat-bots: Telegram Bot
    - Reasoning Systems: Neo4j Knowledge Graph
    - Cognitive Systems: Google Dialogflow
    - YOLO
    - Pertained Machine Learning Models and Deep Learning Models: Vision, NLP

*Optional

Number of Learner Interns required: (Please specify their tasks if possible)

A team of four project members:

1. Chen Ziqi
2. Li Peifeng
3. Mediana
4. Xu Xuanbo

Team Formation & Registration

Team Name: Group 8
Project Title (repeated): SnapYummy, Intelligent Cooking Assistant
System Name (if decided): SnapYummy (Telegram Bot)
Team Member 1 Name: Chen Ziqi
Team Member 1 Matriculation Number: A0231425R
Team Member 1 Contact (Mobile/Email): chenziqi@u.nus.edu
Team Member 2 Name: Li Peifeng
Team Member 2 Matriculation Number: A0231447J
Team Member 2 Contact (Mobile/Email): lipeifeng98@u.nus.edu
Team Member 3 Name: Mediana

Team Member 3 Matriculation Number: A0231458E
Team Member 3 Contact (Mobile/Email): e0703490@u.nus.edu
Team Member 4 Name: Xu Xuanbo
Team Member 4 Matriculation Number: A0231436M
Team Member 4 Contact (Mobile/Email): e0703468@u.nus.edu

For ISS Use Only		
Programme Name:	**Project No:**	**Learner Batch:**
Accepted/Rejected/KIV:		
Learners Assigned:		
Advisor Assigned: Contact: Mr. GU ZHAN / Lecturer & Consultant Telephone No.: 65-6516 8021 Email: zhan.gu@nus.edu.sg		

Appendix B: Techniques and Skills (MR, RS, CGS)

Course Module	Techniques and Skills
Machine Reasoning	• Word2Vec • Cosine Similarity • Deep Learning with neural network A multi-object detection model has been applied to recognize the food in the refrigerator.
Reasoning Systems	• Knowledge Graph The team has built a recipe knowledge base via Neo4j. The knowledge graph stores nodes and relationship with different attributes. To explore more latent values, the rule-based reasoning has been applied in existing graph.
Cognitive Systems	• Google Dialogflow The team has utilized Google Dialogflow to interact with human naturally, learn user's question, then response according to user's request. It serves as a system to mimic human reasoning, detecting intents and extracting information from user.

Appendix C: Installation and User Guide

Reference Link:

https://github.com/SCNUJackyChen/IRS-PM-2021-11-07-IS03FT-GRP8-SnapYummy/blob/main/ProjectReport/SnapYummy%2C%20Intelligent%20Cooking%20Assistant%20-%20User%20Guide.pdf

Appendix D: Individual Reports

Name: Chen Ziqi	**Matriculation Number: A0231425R**

Personal contribution to group project:

I participated in the development of four modules in our system:

1. Food Detection
   The food detection module acts as "eyes" of our overall system, if the food cannot be detected in high accuracy, it will cost the users a lot of time to modify the ingredients, thus resulting in poor user experience. In order to train a robust object detection model with high recall, I collected and reorganized public dataset using the FiftyOne platform. I also participated in the building of customized refrigerator dataset and labelled for over 120+ images using makesense.ai tool. For model training, I wrote a Jupyter notebook script to enable the training and testing of YOLOv5 model, collected the training result data and visualize the training results by plotting them to charts.
2. Knowledge Graph Deployment and Querying
   I deployed the Neo4j knowledge graph on Docker so user don't need to install and configure the environment of it. I also took charge of the knowledge database querying part in the project, and designed the CQL query sentence in order to search expected content, like return the most suitable recipes based on input ingredients and preferences.
3. Backend Development
   In this module, I was mainly responsible for writing codes for the interaction logic, including linking the neo4j database and querying through python interface, intention handling for dialogflow.
4. Project Report and Video
   I wrote my work (mentioned as above) in the report and made system architecture introduction video with my teammate.

What learnt is most useful for you:

1. This project helps me have a deeper understanding and more experience on building a knowledge graph. The most useful thing that I learnt would be how can I apply the knowledge graph in a real-world scenario and how to explore and mine the latent value inside the KG.

2. Looking back at our training log, totally we have conducted 62 experiments to explore and find the most suitable setting (image size, dataset etc) for training. It helps me grip some tricks and experience in building a robust object detection model.

How you can apply the knowledge and skills in other situations or your workplaces:

I am interested in NLP field, knowledge graph works together with NLP can make a big difference in lots of scenarios like chatbot and recommendation system. If I was assigned to develop a chatbot or recommendation system for my customers, I would try to use knowledge graph skills to in the system.

Name: Li Peifeng	Matriculation Number: A0231447J

Personal contribution to group project:

In project's development, I was mainly responsible for implementing object detection with YOLOv5 and developing backend with Python.

Object detection: data collection & processing

To implement object detection, I worked closely with my teammate Chen Ziqi. We first selected 43 kinds of commonly seen cooking ingredients as our detection categories, then we built our own dataset by re-using some data from Google Open Images dataset, and manually collecting & labeling some by ourselves. Because data from different sources were labeled in different ways (numbers represent categories were duplicated or discontinuous), I also wrote a Python script to process category labels, in order to merge data into one usable dataset.

Object detection: YOLOv5 model training

After building dataset, Ziqi and I fine-tuned the official YOLOv5s model, which is pre-trained on ImageNet dataset, using our self-built dataset for around 250 epochs with Google Colab. In order to fit the actual use scene, another self-built dataset, which contains about 50 images of fridge with multiple ingredients stored in it, was used to fine-tune the model again. The mAP@0.5 of the final model reached 0.76 and the model performs relatively well in actual use.

Backend development

For the backend development, I worked with my teammate Chen Ziqi and Mediana. I implemented the backbone of our backend: connection and communication with the Telegram bot and the Dialogflow agent, with telepot and google-cloud-dialogflow libraries. Also, I used Pytorch to deploy our YOLOv5s model. My parts were later integrated with components developed by the other two, forming the final workable structure.

What learnt is most useful for you:

Through this project journey, I have gained:

- experience in data collection, labelling, processing, and ML/DL model training
- knowledge about object detection, including common models used (e.g. YOLO, SSD), their working principles, and evaluations to these models (e.g. IOU, mAP).
- large improvement in Python skills.

How you can apply the knowledge and skills in other situations or your workplaces:

I want to work in computer vision field after graduation, so the knowledge and skills I have gained through this practice will be very useful to me. If I am assigned with similar object detection tasks, I can use what I have learnt to propose and implement workable systems fast, which will be very helpful for finding an ideal solution.

Name: Mediana	Matriculation Number: A0231458E

Personal contribution to group project:

Following are my contributions in SnapYummy, IRS project:

1. Recipe Dataset Acquisition and Pre-Processing
   I researched on multiple recipe datasets that match our project's requirements and proposed the datasets to the team. Although, there are multiple sources available on the Internet, however many of it is either incomplete (with missing values), or do not contain the information that we needed. Finally, two datasets are chosen and then perform further pre-processing.
2. Recipe Knowledge Design and Activity Flow
   After confirmed on the dataset, directions, and task we would like to solve in this project, I drafted the initial design of the Recipe knowledge modelling and flow. This includes the recipe knowledge representation and reasoning using Frame System, and the activity flow of how users may interact with our systems, including browsing mode and cooking mode.
3. DialogFlow – Intent Detection
   I worked out on the intent that are needed in this project and implemented using DialogFlow. I have also used the Dialogflow Contexts function to control the flow the conversation and ensure the chatbot understand or know what it means or the intention when users input certain words.
4. FrontEnd Telegram Setup and Backend DialogFlow Script Integration
   For the FrontEnd, I setup a telegram account and create SnapYummy chatbot. While for backend, I worked on the integration of backend script with Telegram and DialogFlow, together with Peifeng and Ziqi.
5. Report Writing and Video Presentation
   I wrote up the user guide and project report for the segment mentioned above as well the executive summary, and made the second video presentation (business value and use case demo)

Throughout the course of the project, I also took up the responsibility to manage the project. This includes planning for the next task and timeline to complete, communicating with the team members regularly either via chat or meeting to update the progress of the project, list out potential challenges and provide few options for resolving it.

What learnt is most useful for you

Through this project, I appreciated the important of creating a good chatbot, and challenges in creating a good one. A chatbot that do not understand users' intention or capture users' intention wrongly, or even not user friendly may deter users from using the chatbot and reflects badly on the developer (or company). At the same time, as the databases grows or chatbot becomes more complex, the likelihood of it giving the wrong response also increases. We need to find a good balance between usability, accuracy, and complexity. Secondly, data acquisition and pre-processing are crucial stage in AI project. It is important to choose suitable and well-processed the data, otherwise it will impact the subsequent task. For example, no matter how sophisticated our model training algorithm, it might get the wrong prediction results if our data are not process well. Lastly, I also learnt how to manage an AI project, improve my communications skills and management skills.

How you can apply the knowledge and skills in other situations or your workplaces:

In business place, new joiners always need a period to understand the company. On top of having buddy (colleagues) to assist and familiar with the company, we can also create an AI system with chatbot to learn the details of the company, including culture, organization structure, HR matters and so forth. The new joiners can then interact with the chatbot to learn about the company. New joiners may also upload a photo of specific location in the office to chatbot (scene recognition), the chatbot may then interesting things of this location or 'good to know'.

Name: Xu Xuanbo	Matriculation Number: A0231436M

Personal contribution to group project:

Knowledge acquisition:

1. I proposed a word2vec+kmeans model to mine the keyword information related to cuisine and dietary from the dataset.
2. To fill the rest of the missing cuisine labels, I used the word2vec model to encode the ingredient to a vector. And further those vectors are applied it to the deep CNN model for the recipe cuisine type Prediction. The final model achieved 70% prediction accuracy on all 26 categories.
3. To improve the effective matching rate of recipe retrieval, I also used four different similarity methods to cluster the ingredients based on the Ingre2vec model, so that the ingredients were divided into the corresponding main ingredients.

Knowledge representation:

1. I use neo4j to represent the constructed knowledge as a knowledge graph. To load millions of data, I conducted deployment experiments on two operating systems (windows and linux) to compare the four different official data loading methods of neo4j. Finally, the knowledge obtained is converted into a specific csv format through a script and imported through the batch script neo4j-admin of neo4j. This method can stably import millions of data within 5 minutes. The estimated time is nearly 100 times faster than the original method.

Knowledge discovery:

1. I designed some rule-based reasonings and implemented it with CQL language.

Backend:

1. I also assisted teammate Ziqi to write some complex knowledge graph CQL query logic.

What learnt is most useful for you:

I have benefited a lot from practicing all aspects of knowledge engineering. In this project, I experienced how to build a knowledge base almost from scratch. Firstly, I learned how to deal with the challenge that knowledge cannot be extracted explicitly or directly. And I also learned how to construct a knowledge graph. Finally, I learned how to better make use of a knowledge graph.

How you can apply the knowledge and skills in other situations or your workplaces:

I may enter traditional manufacturing in the future and become a data scientist in the field of manufacturing. I think the solution pipeline of this entire project can also be applied to the manufacturing industry. Our project has the whole process from perceive (camera), learn (yolo train and knowledge graph building), reason (yolo, dialogflow, knowledge graph) to act (answer or recommend). In the manufacturing industry, taking anomaly detection as an example, solution pipelines such as image-based anomaly detection, anomaly cause finding, and automatic anomaly handling could also be applied to solve the problem. What I learned from this project includes image recognition, knowledge extraction based on NLP technology, and knowledge graph construction can lay a good foundation for my future work.

Appendix E: Cuisine Lists and Dietary Lists

Cuisine			
Nigerian	Swedish	Australian	Filipino
Lebanese	Pennsylvania Dutch	South African	Turkish
Portuguese	Ethiopian	Greek	Tex Mex
Belgian	Cajun	Southern_Us	Swiss
Czech	Cajun_Creole	Caribbean	Austrian
Finnish	European	Italian	Dutch
South American	Hawaiian	French	Egyptian
Costa Rican	Canadian	German	Danish
Moroccan	Welsh	Vietnamese	Japanese
New Zealand	Georgian	Indonesian	British
Thai	Russian	Hungarian	Chinese
Asian	African	Indian	Pakistani
Irish	Malaysian	Norwegian	Iraqi
Puerto Rican	Creole	Korean	Jamaican
Native American	Mexican	Brazilian	Cuban
Palestinian	Spanish	Scottish	Christmas

Dietary	Synonyms
vegetarian	vegetarian, veggies, veg, vegan, vegetables only, veggies only, vegg, herbivorous, gluten-free, plant-eating, veggie-lovers
halal	halal, no pork, no lard
eggetarian	Eggetarian, vegetarian
fruitarian	fruitarian, fruits, fruit
non-vegetarians	non-vegetarians, non-veg, non-vegetarian, meat-lovers