NAMED ENTITY RECOGNITION Ontology Chatbot

Lim Zhi Yang

School of Computer Science and Engineering 2018

NANYANG TECHNOLOGICAL UNIVERSITY

SCE17-0524 NAMED ENTITY RECOGNITION ONTOLOGY CHATBOT

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Computer Science of the Nanyang Technological University

by

LIM ZHI YANG

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING AY 2017/2018

Abstract

The proliferation of conversation agents is becoming prevalent due to the myriad of benefits it brings about. This has led to a revolutionised way of consumers interacting with machines. Thus, in this paper, the development of chatbot using Dialogflow as intents and queries extractor will be evaluated. The focus of this chatbot is on the parsing of complex queries using named entity recognition and returning the total cost and summary of the food and drinks order back to the users in NTU North Spine Canteen A. In addition, information of the system's architecture and modules used to implement the system for both the Front-end and Back-end will be discussed.

Acknowledgements

This Final Year Project (FYP) was made possible with the help and support from the following people, who have helped me gain much more than what I have expected. I am truly grateful and thankful to them.

I would like to express my deepest gratitude to my FYP supervisor, Assoc Prof Chng Eng Siong, for his valuable time devoted in guiding me throughout the project. His insights have always challenged me to think more critically on how to improve the quality of work for my FYP.

Next, I would like to thank Vu Thi Ly, a research associate in NTU and Kyaw Zin Tun, a project officer from Rolls-Royce@NTU Corporate Lab, for their constant feedback and assistance throughout the development of the software. Without them, my FYP would not have been able to progress so smoothly.

Contents

List of Figures

1. Introduction

1.1 Background

With the rapid advancement of technology, automation technology has now restructured several business landscapes, supercharging performance for their operations [4]. This has brought about disruptive implications to businesses, transforming the way decisions are made and how processes can be optimised. Inevitably, businesses are pressured to make modifications to their business models or adopt entirely new models in view of the industrial revolution.

By leveraging on automation technologies, administrative and or repetitive matters for several processes over a wide range of domains can be automated [3]. This frees up financial and human resources that can be better allocated in other areas of operations which proposes revenue and profit gains. Recent findings have shown the integration of automation technology into business models, have led to increased competitiveness and more customized control for customers. This is evident in New York where check-out times are seen faster, and unexpected operational improvement are observed since 2008 with the presence of self-checkout kiosks in fast food restaurants [5]. Out of the multiple applications of AI technologies, chatbots are ranked the most popular in demand.

Chatbots are defined as computer programs or a service, powered by rules and sometimes artificial intelligence, that users interact via a chat interface. Chatbots provide human-like technology that mimic the way a human would converse with the user. Thus, Chatbots are deployed by businesses to answer user's enquiries for frequently asked questions as they are available 24 hours a day without requiring much overseeing by humans.

With the increased adoption of mobile Internet and messaging platforms in the society, Chatbots are usually deployed on mobile messaging platforms [6]. Leveraging on the mobile messaging platforms, the extent of reach of the chatbots would be enormous. According to Business Insider (2016), approximately 3 billion people worldwide use mobile messaging applications such as Facebook Messenger, WeChat, Skype, Telegram, Slack, Viber, and

Kik [7]. For many users of these services, natural language is expected in online interactions, making automated processes using natural language a business opportunity with huge potential. Recognising this opportunity, global technology companies such as Facebook and Microsoft have offered tremendous support in providing resources for chatbot developments. As of 2016, Chatbots is seen growing rapidly with 11,000 Facebook Messenger chatbots being developed [1] and 80% of businesses are envisaged to have some sort of chatbot automation implemented by 2020 [2] with the initiatives by the technology giants.

The introduction of automation technology into the industries in particularly, Food and Beverages Industry, has revamped the way food and beverages are ordered. Many restaurants chains are seen jumping onto the bandwagon by utilising chatbots as one of their food ordering platform. This can be seen in a few restaurants such as Wingstop [8], Pizza Hut, Burger King, Taco Bell and Whole **Foods** Market [9]. After the implementation of the Chatbot, boost in profit and revenue can be observed in the restaurants [8]. As such, implementation of food ordering chatbots can be done in schools, to speed up the rate at which food and beverages are ordered. This will provide students with more time to eat their meals as lesser time can be spent on queueing for their food given the short breaks they have in between their lessons.

1.2 Aims and Objectives

Humans usually speak informally when they are ordering food. This makes their queries informal and possess different characteristics as compared to formal English language. Thus, the objective of this project is to develop a chatbot that can process both formal and informal queries pertaining to the ordering of food and drinks in Canteen A in North Spine NTU.

1.3 Scope

This scope of this project covers the comparison of the performance between the various chatbot development platforms, developing a full-stack prototype chatbot, how entities can be recognised and extracted based on user's queries, multiple platforms for the user interfaces (HTML, Facebook Messenger). Lastly, this project evaluates the speed of how fast replies can be processed and returned to the user.

1.4 Report Organisation

This report consists of 6 Chapters. The first chapter provides the background information of this Final Year Project. The second chapter discusses about various existing techniques and system designs for Chatbot Development. The third chapter explains the proposed solution and methodology. The fourth chapter discusses the implementation details of the system, followed by the elaboration on the system demonstration in the fifth chapter. Finally, the six chapter closes the report with conclusion and future works of this project.

2. Literature Review

This section covers the overview of how chatbot functions. In addition, the design and implementation options to it will be covered.

2.1 Overview of Chatbot

The main function of a Chatbot is to take in inputs from users and return an appropriate response back to them. Chatbots based its conversational ability on Natural Language Processing (NLP) and Machine Learning (ML) concepts.

2.2 Types of Development

A chatbot can be built in two ways, mainly the rule-based approach and machine learning.

2.2.1 Rule-Based Approach Development

Rule-based approach requires the chatbots to adhere to certain specific rules and provide a dedicated response to the specific input. This approach is reflected in Figure 2.2.1, below.
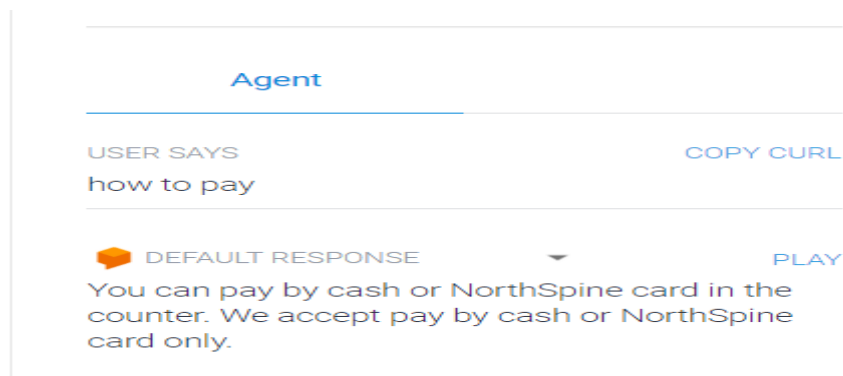
Figure 2.2.1 Response to Rule-Based Approach

In Figure 2.2.1, it is observed that the user asked, "How to pay". After receiving this input from the user, the chatbot automatically answer "You can pay by cash or NorthSpine card in the counter. We accept pay by cash or NorthSpine card only.". This is a typical approach used in creating question-answer bots as seen
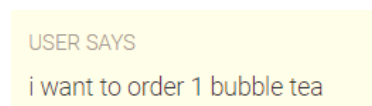
in the above example since the chatbot is hardcoded to provide this specific answer to the specific input that the user queried. Using back the same example mentioned above, should the input changed to "How to pay now? "The chatbot would mismatch incorrect intents to the input as it deems the input as a different data from what it was trained with. This arose due to a surplus of word input into the chatbot. Thus, inference can be drawn from it whereby should there be a lack or surplus of words input into the chatbot, the chatbot will not able to accurately allocate the correct intent to the input.

Rule-based chatbots requires minimal financial resources to implement and can be developed easily in a short period of time with minimal difficulty should the requirements be relatively simple. However, this type of approach renders the chatbot to not possess human intelligence since they cannot understand the intent or context of the queries by the users. This leads to the developer having to programme a myriad of answers to accommodate to the various ways of which questions can be asked, which is not feasible due to the infinite possibilities of how the queries can be phrased.

2.2.2 Machine Learning Approach Development

Machine learning approach allows chatbots to learn as they go rather than following specific hardcoded examples as discussed in section 2.2.1 of this report.

With the advancements in machine learning and natural language processing domains, algorithms pertaining to neural networks now allow the chatbots to draw valid assumptions based on the patterns of the large amount of data that the chatbot was trained with. After which, the chatbot will map it to a concept such as a semantics, or the intent of the input. Take for example, the user queried:

USER SAYS
i want to order 1 bubble tea

Figure 2.2.1 Example of a User Query

After being exposed to multiple queries of the same pattern as seen in Figure 2.2.2,

I want to order <quantity> <drink>

Figure 2.2.2 Sentence Pattern of User Query

Upon future queries of similar pattern seen in Figure 2.2.3,

USER SAYS
i want to order 4 Coffee

Figure 2.2.3 Example of a Different User Query

When the user queries the input in Figure 2.2.4, the chatbot will be able to deduce the user is trying to order 4 cups of the drink, Coffee, even if it previously do not know Coffee is a form of drink and 4 is a cardinal number.

Intuitively to humans, one can detect the pattern after exposing to multiple similar sentence structures. However, this is not the same for machines. Training a chatbot to recognise such patterns are time-consuming and labour intensive. Without exposing sufficient amount of sample sentences that comes along with labels within the sentences, there is this possibility of the machine misclassifying the structure of the sentence. Hence, there lies a need for the developer to come up with not ten or hundred samples but rather, ten-thousands, hundred-thousands or millions number of sentences that are varied in the way they are phrased. One can imagine the amount of resources spent on labelling the sentences and for training the model of the chatbot just to ensure the accuracy of the pattern recognition. This type of approach is classified as a "Black Box" approach since the developers do not possess any knowledge of the internal workings of the chatbot and only can observe the inputs and outputs of it.

2.3 Comparison Between Various Chatbot Development Platforms

A chatbot entails a publishing and development platform. A chatbot publishing platform is the medium through which the user can access the user interface. It is usually on a webpage or through other messaging platforms such as

Facebook Messenger, Slack, Telegram, Discord or Kik. Whereas a chatbot development platform is the tool that aids in the infusion of functions into the chatbot. Some of the examples of the chatbot development platform includes Microsoft bot Frameworks, Wit.ai, Dialogflow, IBM's Watsons. These development platforms assist in creating intelligence in a chatbot by leveraging on ML and NLP concepts. A comparison of the various development platforms can be observed in the table below.

Bot Name	Features	Programming languages / Apps / Integration	Technical details	Project Link
IBM Watson Conversati on Service	Built on a neural network (one billion Wikipedia words). Has three main components: Intents, Entities, Dialog	Node SDK Java SDK Python SDK iOS SDK Unity SDK	-	https://www.ibm.com/watson/ai-assistant/
wit.ai	Allows to use: Entities Intents Context Actions Natural Language Process (NLP)	Node.js client Python client Ruby client On other platforms: HTTP API	Is available for developers to use with iOS, Android, Windows Phone, Raspberry Pi, Python, C and Rust. JavaScript plugin.	https://wit.ai/
rasa NLU	intent classification entity extraction	HTTP api Python	You can use rasa as a drop-in replacement for wit, LUIS, or Dialogflow	https://rasa.com/
Dialogflow	DIALOGFLOW matches the query to the most suitable intent based on information contained in the intent (examples,	SDKs: Android iOS Cordova HTML JavaScript Node.js .NET Unity	-	https://dialogflow.com/

	entities used for annotations, contexts, parameters, events) and the agent's machine learning model. DIALOGFLOW transforms the query text into actionable data and returns output data as a JSON response object. Leverage predefined knowledge packages collected over several years.	Xamarin C++ Python Ruby PHP (community supported) Epson Moverio Botkit Java		
Microsoft Bot Framework	Understands the user's intent. To give your bot more human-like senses, you can incorporate LUIS for natural language understanding, Cortana for voice, and the Bing APIs for search.	Bot Builder SDK (.NET SDK and Node.js SDK.) Bot Connector Developer Portal Bot Directory	The framework provides the Direct Line REST API, which you can use to host your bot in an app or website.	https://docs.microsoft.com/en-us/azure/bot-service/?view=azure-bot-service-3.0
Microsoft Language Understand	Uses intents and entities.	C# SDK Python SDK	LUIS offers a set of programmatic REST APIs that	https://www.luis.ai

| ing Intelligent Service (LUIS) | All LUIS applications are centered around a domain-specific topic or content related.

Active learning.

You can use pre-existing, world-class, pre-built models from Bing and Cortana.

Deploy models to an HTTP endpoint with one click. LUIS returns easy-to-use JSON. | Node JS SDK
Android SDK | can be used by developers to automate the application creation process. | |

As there are multiple chatbot development platforms available in the market, this report will only be focusing on Dialogflow, as that is the platform of which the proposed solution of this report was developed in. For more information on the comparison of the Natural Language Understanding services, one can refer to the project links stated in the table above or, refer to an article on the evaluation of services for conversational question answering system [10].

2.3.1 Dialogflow

Dialogflow is previously known as Api.ai and is a platform catered for intent-based chatbots. Dialogflow supports more languages than Microsoft Language Understanding Intelligent Service (LUIS), RASA NLU and IBM Watsons Conversation Service. Dialogflow would match the user's query based on the intents and the machine learning model it was trained with. To accurately match to the correct intent, Dialogflow would consider the entities labelled annotation, context, parameters and events as seen in Figure 2.3.1 before

deciding which intent it would classify the input under. Dialogflow takes in the input as string format and transform this data into JSON response object output as reflected in Figure 2.3.2.
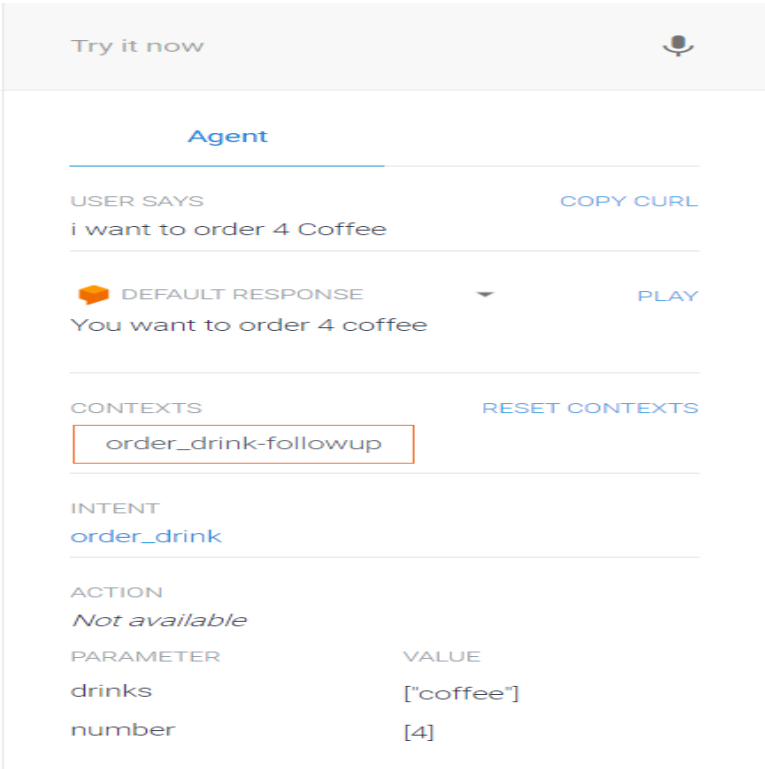
Figure 2.3.1 Values Reflected in Dialogflow Console

Diagnostic info


```
21            }
22         }
23       ],
24 ▼     "outputContexts": [
25 ▼       {
26           "name": "projects/foodbotntu/agent/sessions/3eddef91-7c52-b086-e7d3-505f6790474b/contexts
                /order_drink-followup",
27           "lifespanCount": 2,
28 ▼         "parameters": {
29 ▼           "drinks": [
30               "coffee"
31             ],
32             "drinks.original": "Coffee",
33 ▼           "number": [
34               4
35             ],
36             "number.original": "4"
37           }
38         }
39       ],
40 ▼     "intent": {
41         "name": "projects/foodbotntu/agent/intents/580ecbfd-10e5-4603-95dd-70a2da925546",
42         "displayName": "order_drink"
43       },
44       "intentDetectionConfidence": 0.59,
45 ▼     "diagnosticInfo": {
46         "webhook_latency_ms": 83
47       },
48       "languageCode": "en"
49     },
50 ▼   "webhookStatus": {
51       "code": 5,
52       "message": "Webhook call failed. Error: 404 Not Found"
53     }
54 }
```

Figure 2.3.2 Diagnostic Info of Dialogflow

It is worthy to take note of the way Dialogflow presents the data to the user in the user interface. The user experience will diminish given the current version of the Dialogflow since Dialogflow is currently not able to format its output with line breaks when integrated on webpage with its web demo settings. This can be observed in Figure 2.3.3 where the responses are not aligned.

Figure 2.3.3 Misalignment of Spacing

2.4. Design and Implementation Considerations

With reference to section 2.2, conclusion about the limitations of the different development techniques can be made. Particularly, a lot of improvements can

20

still be made for the areas of NLP and ML as many matters remains ambiguous. Both approaches require intensive manual labour to come up with tons of data to facilitate in training the model of the chatbot. The situation is further exacerbated by the fact of having to allocate resources to label the entities in the data for the ML approach. On top of that, both processes will need a long duration of time to train the model since the training of the model is based on a neural network approach. To heighten the accuracy of detection of entities, a few processing can be done to the input data.

2.4.1 Pre-Processing of Data

Data will come in "dirty" upon input from the users. Thus, to achieve the highest efficiency and accuracy for information extraction, data cleaning can be done. Data cleaning done aptly will help to significantly reduce the data dimensionality of term space. This translates to better information extraction efficiency as there are lesser data for processing for intent classification.

2.4.1.1 Case of Data

To facilitate in the data retrieval from databases like SQL server, there is a need to pay attention to the case of the data in whether it is in upper or lower case. This is important since databases like SQL server, are case-sensitivity. Hence, to avoid error from arising upon data retrieval, all the data can be changed to lower case before passing on to the databases.

2.4.1.2 Special Characters, Punctuations, Symbols and Multiple Spaces

Since users may accidentally type or insert special characters, symbols, punctuations and multiple spaces into the query, such regular expressions are removed. This will ensure the query only contains information that are vital to the recognition of the entity types.

2.4.1.3 Word forms of numbers

Numbers exist in integer and word forms. This can be confusing to the machine as the machine may perceive the word forms of numbers to be a normal English word instead of as a quantity that the user wishes to use for his or her order. Thus, a dictionary of numbers was created that maps the word forms of the numbers to the integer value of the numbers. This will help to transform all the word forms of numbers to integers values which can be readily used later for reflecting the quantity of the food.

2.4.1.4 Singularity of the Input Item

To ensure the correct named entities is recognised by the chatbot's model, the singularity of the input should be consistent where all plural nouns should be changed to singular nouns. This hence eliminate any ambiguity of any potential wrong entities connotated with the correct noun.

2.4.1.5 Stopwords

Stopwords are frequently used words by humans. It comprises of examples such as, "I", "and", "you", "he". Such words prove not to be useful in the analysis of data since they do not add on to the uniqueness of the pattern of the data. Thus, these words could be removed to avoid redundant computations for intent classification. To suit the topic of food ordering, a customised list of stopwords was utilised. This list of stopwords contains the common stopwords like, "I", "and", "you", "he", "she", "on" as mentioned above and, common terms used in food ordering like "plates", "cup","wish","want" and 'order'.

2.4.1.6 Stemming and Lemmatisation

Stemming refers to the crude heuristic process that removes the ends of words to reduce the word to its stem or base form. The rules on how to remove the ends of words includes following rules that arise from derivational affixes. Stemming might reduce the inflected word to illogical forms that do not exist in the dictionary.

Lemmatization is the process of removing inflectional endings to return the base or dictionary form of a word, which is known as the lemma. If confronted with the token "ponies", stemming will reduce the token to "poni" whereas lemmatisation will reduce the word to "pony", which is the singular form of ponies.

 A conclusion of lemmatisation taking into consideration the context of the sentence can be drawn. Similarly, a conclusion of stemming not considering the context of the sentence can be concluded as the word form after the process confirmed the hypothesis.

2.4.1.7 Spelling Correction

Lastly, any misspelled words should be replaced with a correct word. A customised corpus suiting to the names of the food dishes was applied here to increase the replacement success rate.

3. Proposed Methodology and System Specifications

3.1. Chatbot Development Platform

In this project, Dialogflow is chosen as the chatbot development platform. Dialogflow is being chosen as the development platform since it is intelligent enough to understand what the user is referring to with its internal mechanism that permutates the query to correctly map the intents.

In addition, it supports an array of messaging platforms for deployment. As of now, it supports 16 platforms which allows an extensive amount of people to be reached easily. This feature is further commendable with its one-click integration, which eases the job of the chatbot developer.

Lastly, it has built-in analytics tool that allow the chatbot owner to track the performance of the chatbot. Its integrated analytics tool can showcase usage patterns, latency issues, high and low-performing intents. This provides chatbots owners to be able to debug and improve the performance of their chatbot with minimal effort.

3.2 Chatbot Publishing Platform

With the large number of consumers utilising the Facebook Messenger application [7], Facebook Messenger is chosen as the publishing platform for the chatbot. This helps users to access the chatbot easily and enhances the interaction with Facebook. On top of that, it will allow smooth integration with Dialogflow since Facebook Messenger is one of the listed messaging platforms that was supported.

3.3 System Architecture

The system comprises of 5 component layers. Figure 3.3.1 shows the system architecture diagram which is, a conceptual model that depicts the structure,

behaviour and relationship between the components of the system. The first layer is the User Interface layer, which is the front end where users will be able to interact with the chatbot by clicking and inserting inputs. Outputs back to the users will be reflected in this layer as well.

The next layer is the abstraction layer, Dialogflow, which is the interface that contains the data required for the next layer, the business layer. At the abstraction layer, the queries of the users and intents will be extracted at this layer and passed into the next layer which is the Python server.

The Python server is the business logic layer that contains the algorithms responsible for the parsing of the data from the abstraction layer. The data is passed from the abstract layer to the business logic layer by webhooking into the server with intents identified in Dialogflow.

Depending on the intents being identified, the next layer, the dependency layer, may or may not be called. The dependency layer contains Spacy libraries and API that does processing for queries that are classified as complex queries, which will be mentioned in the section 3.4 of this report. The dependency layer will only be called upon when the intent identified in Dialogflow is "Complexquery". Should the dependency layer be called, a JSON object will be passed back to the server (business logic layer). Following which, the server will send a query request to the last layer, the persistence layer.

Upon receiving the query from the business logic layer, data as requested in the query will be retrieved from the MySql database. This retrieved data would be passed back to the server, subsequently, Dialogflow and utimately, Facebook Messenger.
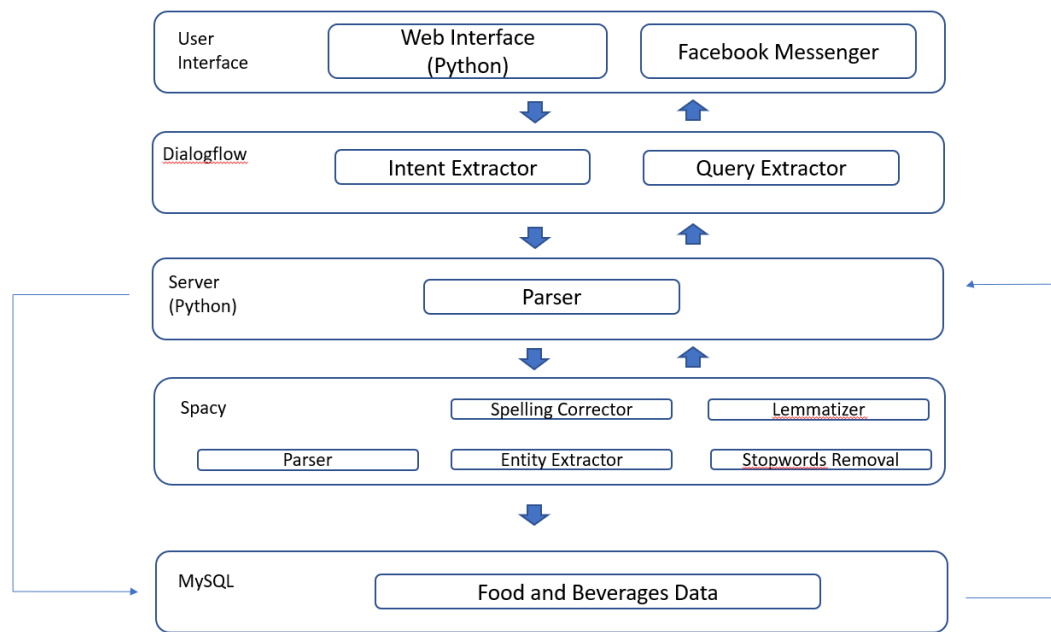
Figure3.3.1 *System Architecture Diagram*

3.4 Complex Queries

The focus of this project is on handling complex queries that is queried by the users. Complex queries are not simple queries that makes sense to the machine. Instead, complex queries are queries that can possibly contain special symbols or characters, Upper and lower-case text within a sentence, multiple white spaces, punctuations and multiple customised order of food. An example of a simple query can be seen in Figure 3.4.1.

Simple Query:

"I want 2 plates of large Chicken Rice"
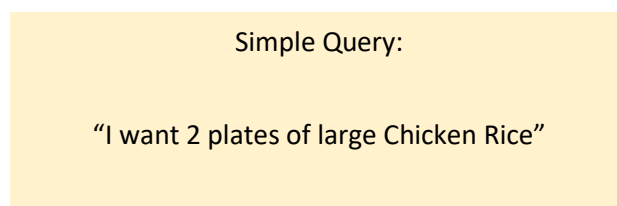
Figure3.4.1 Example of Simple Query by User

On the other hand, complex queries can be of any of the following below in Figure 3.4.2.

Complex Queries:

e.g. "I want 2 small chicken rice, 3 medium fishball noodles and 4 large spaghetti"

e.g. "I want 2 plates of small chicken rice, 3 bowls of medium fishball noodles and 4 large plates of spaghetti"

e.g. "I want 2   small chicken   rice, 3 medium   fishball noodles and 4   large   spaghetti"

e.g. "I want 2 !@@## small chicken rice, 3 !@#$% medium fishball noodles and 4 l%^arge spag!@hetti!"

e.g. "2 small chicken rice, 3 medium fishball noodles, 4 large spaghetti"

e.g. "2 small chicken rice 3 medium fishball noodles 4 large spaghetti"

Figure 3.4.2 Example of Complex Queries by Users

It is highly possible for humans to send a complex query to the chatbot since they may misspell some of the words or even accidentally pressed other alphabets, symbols, characters or spacing. It is not possible to know exactly what sentences the users would send to the chatbot due to the inherent incomprehensible nature of humans. Taking into consideration this point, a solution is formulated to tackle this issue.

4. Implementation

4.1 Server-side

The server side is implemented using Python. Python is a powerful high-level, object-oriented programming language used since 1991. Python utilises human readable language which makes it easy to develop and debug the program. Python developers do not have to worry about the conversion to machine language since python has in-built functionalities to automatically convert the high level interpreted language to machine language for them. This thus saves on memory resources for the python scripts. In addition, its object-oriented feature allows us to solve complex problems easily by creating objects to divide the complex problems into smaller manageable sets. With such appealing features, python proves to be a great language for developing servers.

4.1.1 Environment Set-up

To enable webhook feature in Dialogflow, the server need to be ran with a secured connection, HTTPS and the URL must be publicly accessible. To fulfil both requirements, domain name and SSL certificates are necessary for setting up a public server. Hence, to resolve this, Ngrok can be used. Ngrok exposes local servers behind firewalls to the public internet over secured tunnels. This creates a public HTTPS URL for a website instantly by running the program on our local developing machine. However, for the webhook to be online continuously, the user cannot terminate the Ngrok process nor power off the computer. Thus, to address this problem, a free cloud service provider, Heroku, can be used. Detailed information of Heroku will be discussed in section 4.1.4.2 of this report.

4.1.2 Server Setup and Routing Mechanisms

Creating a server in Python is relatively simple as Python has a framework called Flask. Flask is a relatively new framework that was released in recent years, 2010. Flask can be easily installed with Python Package Index (PPI). Flask application requires an app python script, that is an instance of Flask and the application's root. A http server is created based on the object defined in app.py when app.py is being ran. The setting up of the routing is done in app.py where @app.route decorator specifies the points of interactions the server has with external applications. The interactions consist of GET and POST requests. The GET requests is being used for Facebook authentication

while the POST requests contain the information of the incoming data of both Facebook and Dialogflow. This information for the incoming data will be stored in a request object. The object will then be examined and extracted for its information so that a response can be returned to the client.

4.1.3 Program Control Flow

Figure 4.1.3.1 depicts the control flow diagram of the server-side implementation.
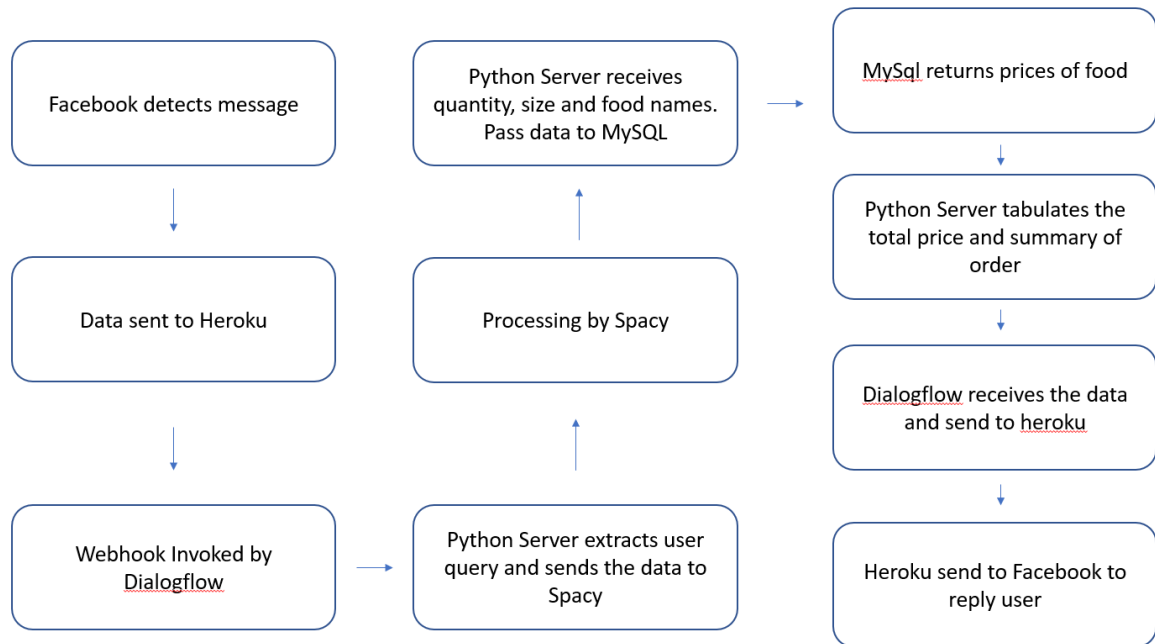
Figure 4.1.3.1 Control Flow Diagram for Server

4.1.3 Named Entity Recognition (NER)

Named Entity Recognition is a process that annotates entities for a sequence of words. One of the state of the art NER model includes Spacy. Spacy is an exceptionally efficient system for NER in Python as it can recognise a wide range of named or numerical entities, which includes food names, quantity and size. In addition to this, it provides the ability to add on new arbitrary classes to the existing trained model.

Spacy outperforms other state of the art models in many domains. For NER, Spacy has a higher F-Score than Stanford NER model. This translates to better accuracy when recognising entities from sentences [11]. At the same time, Spacy significantly outperforms NLTK in word tokenization and Part-of-Speech (POS) tagging as evident in Figure 4.3.1.2 due to the different algorithms used for both the models. [12].
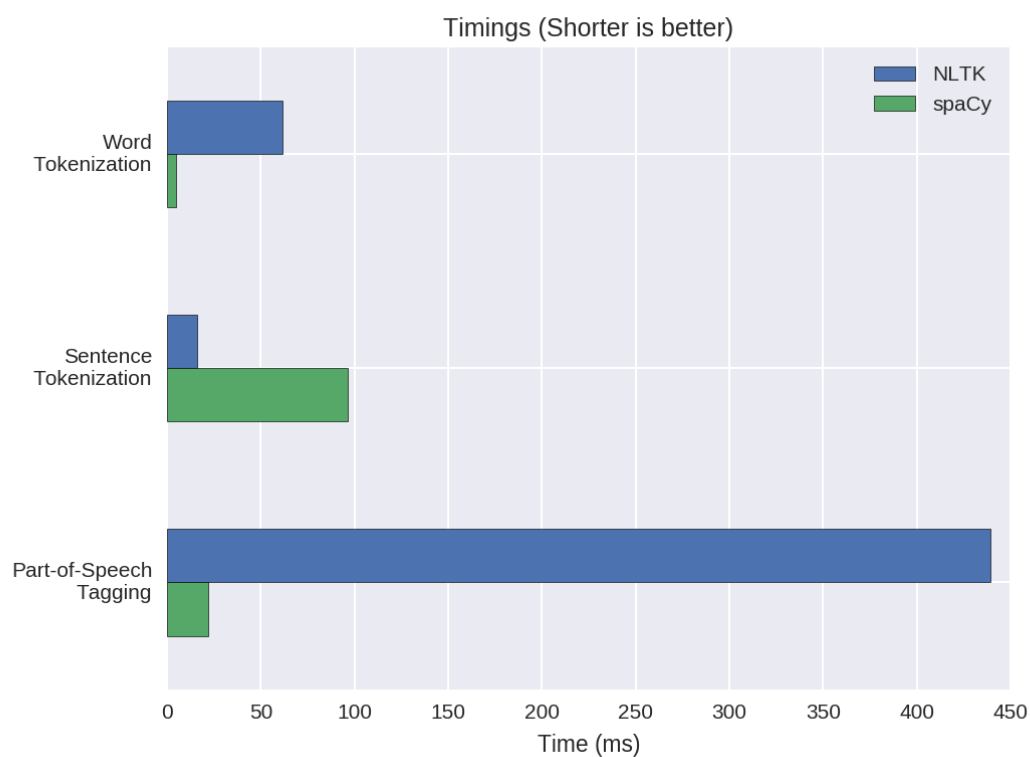
Figure 4.1.3.2 Comparison of Performance Between NTLK and Spacy

4.1.3.1 Training Data for NER

To begin the process of using Spacy to train the model for NER, the training data for the model must be manually created and labelled. An example of the training data for the model can be seen in Figure 4.1.3.1.1.

```python
def fooddatatraining():
    for i in food:
        x = i
        TRAIN_DATA = (

            (x + " is a cuisine", {
                'entities': [(0, len(x), 'FOOD')]
            }),

            ("2 " + x, {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 2 + len(x), 'FOOD')]
            }),

            ("4 large " + x, {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 7, 'SIZE'),
                             (8, 8 + len(x), 'FOOD')]
            }),

            ("3 " + x + " 3 bak kut teh", {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 2 + len(x), 'FOOD'),
                             (len(x) + 3, len(x) + 4, 'QUANTITY'),
                             (len(x) + 5, len(x) + 16, 'FOOD')]
            }),

            ("3 medium " + x + " 5 small chicken rice", {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 8, 'SIZE'),
                             (9, 9 + len(x), 'FOOD'),
                             (len(x) + 10, len(x) + 11, 'QUANTITY'),
                             (len(x) + 12, len(x) + 17, 'SIZE'),
                             (len(x) + 18, len(x) + 30, 'FOOD')]
            }),

            ("2 small " + x + " 4 large biryani 5 large chicken rice", {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 7, 'SIZE'),
                             (8, 8 + len(x), 'FOOD'),
                             (9 + len(x), 10 + len(x), 'QUANTITY'),
                             (11 + len(x), 16 + len(x), 'SIZE'),
                             (17 + len(x), 24 + len(x), 'FOOD'),
                             (25 + len(x), 26 + len(x), 'QUANTITY'),
```

Figure 4.1.3.1.1 Labelling Used for Training the Model

Due to the fact that many Singapore delicacies are not named after dictionary English words such as Singapore famous delicacies, "Char Kway Teow", "Laksa" and "Bak Kut Teh", there lies a need to create a NER model from scratch. To achieve a high accuracy of the NER, millions of data must be constructed and used to train this model. However, due to resources constraints such as insufficient memory size, lack of resources and long training time, the current model was trained using 1047 number of sentences which leads to a lower but, acceptable accuracy of the Spacy model. The number 1047, is the total number of sentences used for training the model over 26 epochs. This number is derived from looping the items contained in each category as seen in Figure 4.1.3.1.2 into the different sentence patterns seen in Figure 4.1.3.1.3.

```
food = ("char kway teow", "biryani", "oyster omelette", "chicken rice", "pasta", "satay", "hokkien prawn mee", "fried carrot cake",
        "hokkien char mee", "curry fish head", "noodle", "spaghetti with sausage", "beef", "rojak", "aglio olio chicken cutlet",
        "roti prata", "spaghetti with chicken chop", "aglio olio chicken chop", "bee hoon", "spaghetti with chicken cutlet",
        "bun bo hue", "spaghetti with ham and mushroom", "pho", "mee siam", "dim sum", "wanton mee", "pho bo", "laksa", "chicken burger",
        "bak chor mee", "crabs", "nasi lemak", "fish burger", "bak kut teh", "fried hokkien mee", "fish n chips")

drinks = ("milk tea", "lao hor", "soy bean", "coffee", "kopi", "beer gao", "milo", "tea", "teh", "teh bing", "milo bing",
        "iced milo", "iced coffee", "iced milk tea")

sides = ("fries", "mash potato", "toast cube", "salad", "soup", "spaghetti")
```

Figure 4.1.3.1.2 Categories of Items

1) <ITEM>

2) <QUANTITY><ITEM>

3) <QUANTITY><SIZE><ITEM>

4) <QUANTITY><ITEM><QUANTITY><ITEM>

5) <QUANTITY><SIZE><ITEM><QUANTITY><SIZE><ITEM>

6) <QUANTITY><SIZE><ITEM><QUANTITY><SIZE><ITEM><QUANTITY><SIZE><ITEM>

Figure 4.1 3.4.3 Sentence Patterns for Training the Model

The spacy model is trained based on a combination of different sentence patterns as seen in Figure 4.1.3.1.3. For items that are annotated with "ITEM" as part of the pattern, it means all the items are trained regardless of whether it is a food, drink or side dish with the possible permutations. Elaborating on that, the third pattern in Figure 4.1.3.1.3 can be "1 small milk tea" which has sentence pattern of same entity type <QUANTITY><SIZE><DRINKS>. However, using a single sentence pattern alone may prove to be unrealistic since it is not a good representation of what actual examples may look like. Thus, patterns that incorporate other entities types were included in the training process. Some of the patterns that consist of the different entities types can also be observed in Figure 4.1.3.1.3.

One of the sentence pattern that varies in terms of characteristics from the third sentence pattern would be the sixth pattern. The sixth sentence pattern consists of a hybrid of different entity types with the example of "2 small chicken rice 4 large coffee 5 medium fries" which follows <QUANTITY><SIZE><FOOD><QUANTITY><SIZE><DRINKS><QUANTITY><SIZE>

<SIDES>. This pattern is evident in Figure 4.1.3.1.4, which shows a clear demonstration of what was being described.

```
("1 " + x + " 3 soup 4 biryani 5 kopi", {
        'entities': [(0, 1, 'QUANTITY'),
                (2, 2+len(x), 'FOOD'),
                (len(x) + 3,  len(x) + 4, 'QUANTITY'),
                (len(x) + 5,  len(x) + 9, 'SIDES'),
                (len(x) + 10,  len(x) + 11, 'QUANTITY'),
                (len(x) + 12,  len(x) + 19, 'FOOD'),
                (len(x) + 20,  len(x) + 21, 'QUANTITY'),
                (len(x) + 22,  len(x) + 26, 'DRINKS')]
    }),

    ("2 small " + x + " 4 large coffee 5 medium fries", {
        'entities': [(0, 1, 'QUANTITY'),
                (2, 7, 'SIZE'),
                (8, 8+len(x), 'FOOD'),
                (9+len(x), 10+len(x), 'QUANTITY'),
                (11+len(x), 16+len(x), 'SIZE'),
                (17+len(x), 23+len(x), 'DRINKS'),
                (24+len(x), 25+len(x), 'QUANTITY'),
                (26+len(x), 32+len(x), 'SIZE'),
                (33+len(x), 38+len(x), 'SIDES')]
    }),
```

Figure 4.1.3.1.4 Annotation Patterns with Different Entity Types

With the inclusion of the other entities types, the model would be more accurate as it is able to handle more unpredictable inputs from the users. This will hence prepare the model to be readier for the different possible ways the users may phrase their food orders.

To further elaborate on the training process, reference to Figure 4.1.3.1.1 can be made. Figure 4.1.3.1.1 shows the usage of a string with a variable "x", for training. In this case, the variable "x" denotes the name of the item. There are three main categories of item being ordered when consumers buy their food. These three main categories are Food, Drinks and Sides. Thus, to achieve acceptable volume for the training, "x" was substituted with different item names for the same sentence pattern. Upon finishing the substitution for all the food names for that sentence pattern, this whole substitution process will be done the same for the other remaining types of sentence patterns as observed in Figure 4.1.3.1.3.

Similarly, the substituting process will apply to the other 2 categories, drinks and sides as well. Figure 4.1.3.1.5 is a demonstration of how some of the drinks data were being trained and Figure 4.1.3.1.6 of how some of the sides data were being trained.

```python
def drinksdatatraining():
    for i in drinks:
        x = i
        TRAIN_DATA = (

            (x + " taste good", {
                'entities': [(0, len(x), 'DRINKS')]
            }),

            ("1 " + x, {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 2 + len(x), 'DRINKS')]
            }),

            ("2 medium " + x + " please", {
                'entities': [(0, 1, 'QUANTITY'),
                             (2,  8, 'SIZE'),
                             (9,  9 + len(x), 'DRINKS')]
            }),

            ("1 small " + x + " 6 large spaghetti chicken chop", {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 7, 'SIZE'),
                             (8, 8 + len(x), 'DRINKS'),
                             (9 + len(x), 10 + len(x), 'QUANTITY'),
                             (11 + len(x), 16 + len(x), 'SIZE'),
                             (17 + len(x), 39 + len(x), 'FOOD')]
            }),

            ("1 small " + x + " 3 large coffee 5 medium soy bean", {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 7, 'SIZE'),
                             (8,  8 + len(x), 'DRINKS'),
                             (9 + len(x), 10 + len(x), 'QUANTITY'),
                             (11 + len(x),  16 + len(x), 'SIZE'),
                             (17 + len(x),  23 + len(x), 'DRINKS'),
                             (24 + len(x),  25 + len(x), 'QUANTITY'),
                             (26 + len(x),  32 + len(x), 'SIZE'),
                             (33 + len(x),  41 + len(x), 'DRINKS')]
            }),

            ("3 medium " + x + " 5 small milo", {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 8, 'SIZE'),
                             (9, 9 + len(x), 'DRINKS'),
                             (len(x) + 10, len(x) + 11, 'QUANTITY'),
                             (len(x) + 12, len(x) + 17, 'SIZE'),
```

Figure 4.1.3.1.5 Training of Drinks Data for the Model

```
def sidesdatatraining():
    for i in sides:
        x = i
        TRAIN_DATA = (

            (x + " taste good", {
                'entities': [(0, len(x), 'SIDES')]
            }),

            ("3 " + x, {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 2 + len(x), 'SIDES')]
            }),

            ("1 small " + x + " please", {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 7, 'SIZE'),
                             (8, 8+len(x), 'SIDES')]
            }),

            ("two large spaghetti 4 small fries 5 large mash potato", {
                'entities': [(0, 3, 'QUANTITY'),
                             ( 4,  9, 'SIZE'),
                             (10, 19, 'SIDES'),
                             ( 20,  21, 'QUANTITY'),
                             (22, 27, 'SIZE'),
                             (28, 33, 'SIDES'),
                             ( 34,  35, 'QUANTITY'),
                             (35, 41, 'SIZE'),
                             (41, 53, 'SIDES')]
            }),

            ("2 large " + x, {
                'entities': [(0, 1, 'QUANTITY'),
                             (2,  7, 'SIZE'),
                             (8,  8 + len(x), 'SIDES')]
            }),

            ("8 " + x + "  9 salad", {
                'entities': [(0, 1, 'QUANTITY'),
                             (2, 2 + len(x), 'SIDES'),
                             (len(x) + 3, len(x) + 4, 'QUANTITY'),
                             (len(x) + 5, len(x) + 10, 'SIDES')]
            }),

            ("3 small " + x + " 5 medium toast cube 9 large lao hor", {
                'entities': [(0, 1, 'QUANTITY'),
```

Figure 4.1.3.1.6 Training of Sides Data for the Model

4.1.4 Hosting a Server for Spacy Model

4.1.4.1 Locally Hosted Server

Upon testing the trained Spacy model, the result returned for Named Entity Recognition is great, format is correct but there is a huge flaw when loading this model. This model requires a long duration of 18 seconds to load, process the query and return the data. Naturally, this will not be acceptable by any users as no user would want to wait for such a long duration for a reply. This flaw dramatically decreases the user experience which will equate to low user satisfaction. Thus, to tackle this bottleneck problem, Spacy Server API can be used. The Spacy Server API loads the Spacy model in a locally hosted server in a separate terminal process. This API keeps the model constantly online listening with a server and client protocol that is based on the Remote Procedure Call-based Services (RPC) concept. Therefore, the model is continuously receptible to inputs and do not have to be loaded with each run.

Caching is done on top of the call-based services using the Least Recently Used (LRU) scheme. This exponentially decreases the loading speed as seen in Figure 4.1.4.1.1 with loading speed dramatically decreasing from 18 seconds to approximately 0.0479 seconds. This implies the Spacy's performance has improved by 429 times.

Improvement in speed

Third input	0.041978121
	18
Second input	0.053968191
	18
First input	0.256837845
	18

0   5   10   15   20
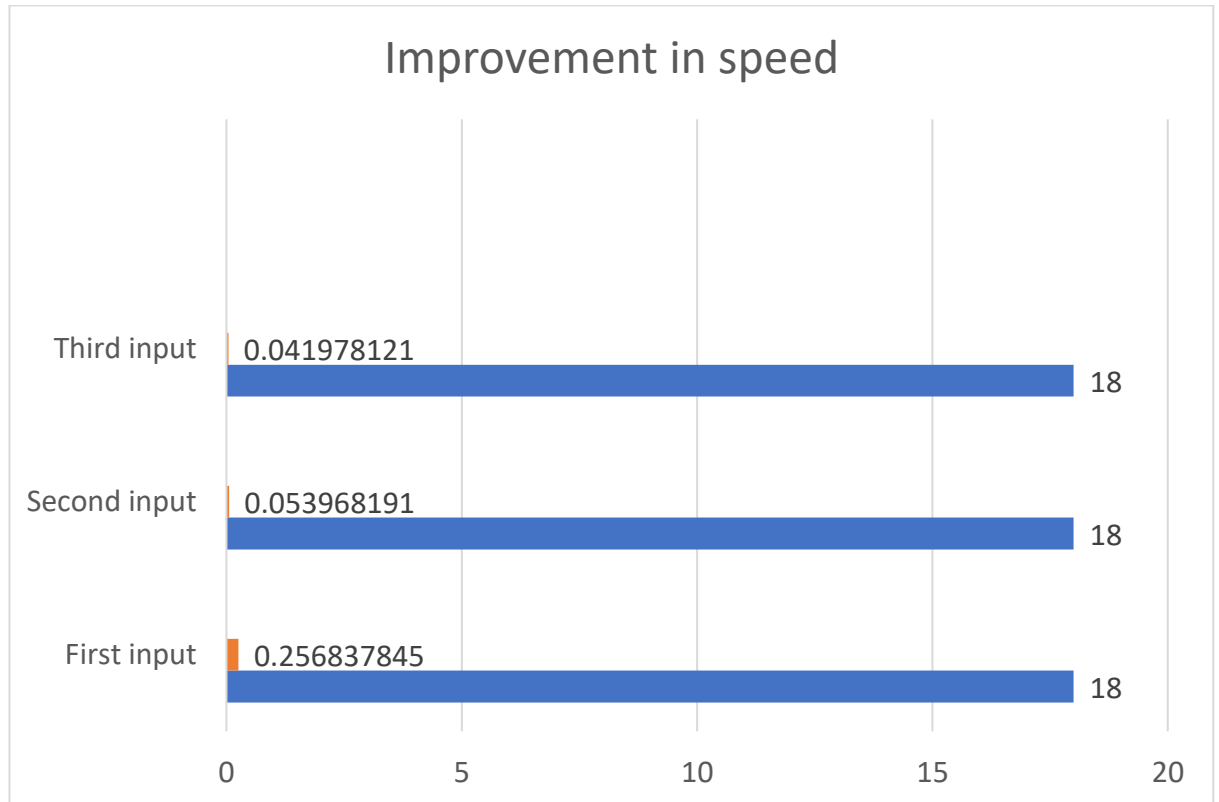
Figure 4.1.4.1.1 *Comparison of Old Run Time with New Run Time After Using Spacy API*

4.1.4.2 Publicly Hosted Server

However, it is resources exhaustive to run Ngrok continuously throughout the day, hence there is a need to switch to a cloud service provider platform that can host the scripts and model without the user running a local server. Heroku, a cloud-based development platform as a service (PaaS) provider is being chosen for this project as a public server for our scripts and model. It supports several development languages such as Node.js, Ruby on Rails, Java and most importantly, Python [13], which is the language used for development in this project. Heroku was chosen since it was built based on Git, which allow effective management over the various versions of the source code.

This project was run using the free-tier plan which provides 1 free web dynos running per process type. The dynos in this tier will go to sleep after 30 minutes of inactivity

[14]. Hence, this limited the server by making it go to idle mode and requiring reconnection to the database and cloud server before any queries can be parsed. This may potentially result to the user experience being lowered. Fortunately, the reconnection will only need a short duration of 10 to 20 seconds to re-establish, before it can return a reply to the user.

By using this server, the need to use spacy server API will be omitted. This action can be justified with the spacy model being loaded from Heroku at a very fast rate since there are multiple background workers servicing the web requests traffic. These background workers may thus help to ensure the site not having a bottleneck while processing the requests of the user. The Heroku server will hence be able to produce a response back to the user at a rapid rate due to concurrency [15]. The tasks the workers will perform will be taken from a queue that has requests in it delivered by the web dynos as seen in the diagram below, Figure 4.1.4.2.1.
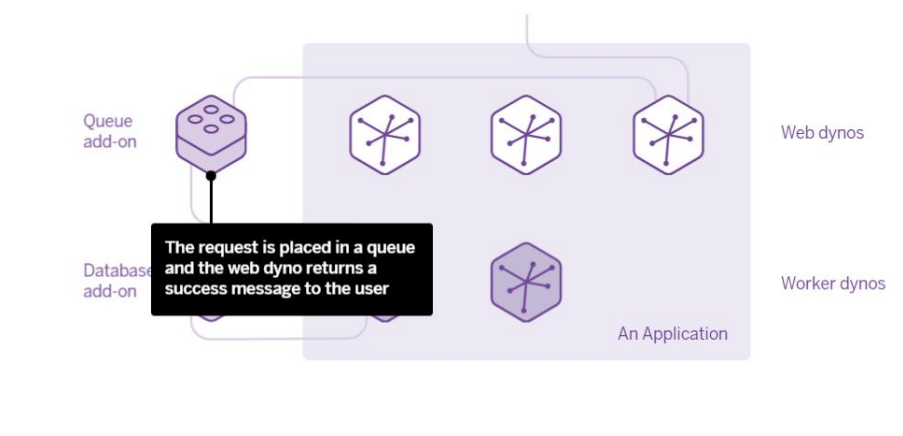
Figure 4.1.4.2.1 How Worker and Web Dynos Work Together in Heroku

4.1.5 Spell Checker Module

A spelling checker or auto correct module is implemented on the server side to pre-process the user's queries. SymSpell library is used for this implementation instead of the well known PyEnchant since PyEnchant is not being updated anymore. SymSpell is a spelling correction module that uses Symmetric Delete spelling correction algorithm. This algorithm, opposite to other algorithms, only requires deletes and not transposes, replaces and inserts. Transposes, replaces and inserts of the input term are substituted with deletes of the dictionary term. Replaces and inserts are computationally expensive and language dependent since they follow a set of dictionary rules for mapping the terms. Take for example, Chinese has 70,000 Unicode Han characters. Thus, the exceptional fast speed of this library is attributed to

36

its inexpensive delete-only edit candidate generation and pre-calculation [16]. Based on Figure 4.1.5.1   , SymSpell is 1,870 times faster than BK-tree with the following conditions (dictionary size=500,000, maximum edit distance=3, query terms with random edit-distance from 0 to maximum edit distance, verbose=0).

It is 1 million times faster than Norvig's algorithm when ran with the following conditions (dictionary size=29,157, maximum edit distance=3, query terms with fixed edit distance = maximum edit distance, verbose=0).
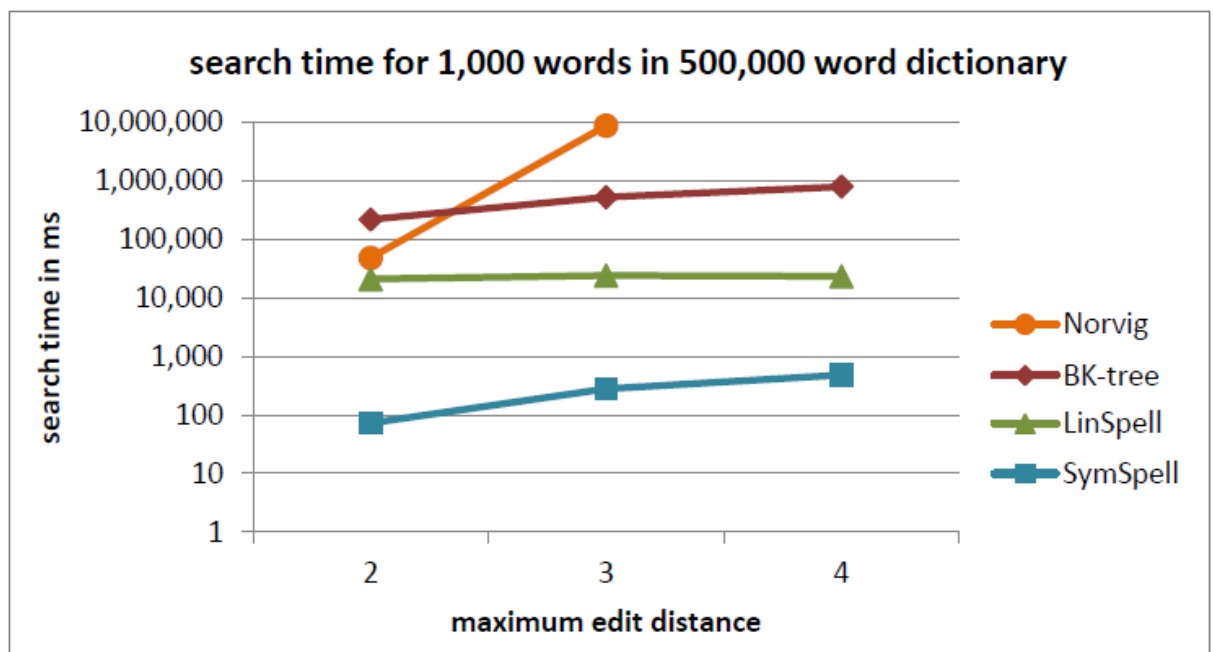
Figure 4.1.5.1 Comparison of the Search Time for Different Spelling Checkers [16]

To further affirm the testing of the SymSpell library, it is tested on a 2012 Macbook that contains a single core [17]. The result was 0.033 milliseconds/word when edit distance is 2. and 0.180 milliseconds/word when edit distance is 3. To aid in increasing the accuracy of rectifying the misspelled words, a customised corpus was created. This corpus mainly contains the unigrams of all the names of the items, be it food, drinks or sides. Since Symspell takes into consideration the frequency of the words when it is doing spelling correction, each word has its own frequency connotated with it. Naturally, words that are in the list of stopwords, will have a higher frequency since there is a higher chance these words will be used. With this customised

corpus, words are not found in an English dictionary such as "Char Kway Teow", can be correctly rectified with ease.

4.1.6 Parser for Food Information

The settings in Dialogflow makes it only receptible to JSON formatted object as input for webhook. Thus, the preparation of this formatting will be done at the server (Business Logic Layer) to accommodate to the Dialogflow settings.

Before any response is being sent back to the user, the message is transformed into a JSON object. This JSON object is annotated with content-type header, JSON, where the client tells the server that a JSON data will be sent to it in the POST request. Similarly, Facebook Messenger only accept JSON object data just like Dialogflow. Hence, the data are converted to a JSON object with the method "json.dumps' from the 'json' library of python.

4.1.7 Database

The server database chosen for this project is ClearDB. This database can be smoothly integrated into Heroku with the available ClearDB add on that helps to tackle compatibility problems with the cloud service provider and the database platform. ClearDB uses a combination of advanced replication techniques and clustering technology with layered web services to provide a robust way in storing data into the MYSQL database [19]. The database created for this project stores the information related to the food and beverages such as the prices, names, description and image URL of the items. Data retrieval can be performed easily from the database to the server by using the MySql connector designed for Python Flask. The connection to the database is done upon initialization of the server. This therefore saves down on the time required to connect to the database before any queries can be performed.

However, it should be noted that the connection to the database will timeout after 30 minutes of inactivity from the users. This happens because in the free tier Heroku accounts, there is limited system resources for usage. Thus, Heroku sleeps the app's container to free up the global system resources [14]. This will lead to users not getting a response immediately when they chat with the Chatbot as there is a need for a reconnection to the server and database to take place as mentioned previously in section 4.1.4.2. This reconnection will require 10 to 20 seconds to establish before any web request can be processed [14]. Although this may create dissatisfaction, the

benefits outweigh the cons since the price increases rapidly upon moving to a different tier.

The database used for this project contains 3 tables, Food, Drinks and Sides. Each of these tables contains 4 columns, name, price, description and image URL of the item.

4.2 Front-End

As discussed in section 3.1, there are a few publishing platforms to be used as user interface. In this section, we will discuss the implementation details for the Facebook Messenger platform as that is the publishing platforms used for this project.

4.2.1 Facebook Messenger

For the users talking to the Chatbot for the first time, there is a need to guide the users in using the Chatbot. Users will be greeted with a welcome screen that comes with a "Get Started" button at the bottom of the screen when they attempt to send a message to the page. After clicking the "Get Started" button, they will be greeted with a message that states how they can begin ordering their food with a quick reply button at the button of their screen. This quick reply button will allow them to request for the pictorial menu without having to type the text out. The sending of the greeting message is activated upon receiving a postback with the message "get started" as seen in Figure 4.2.1.1.

```python
elif message_text == "FACEBOOK_WELCOME":
    r = requests.post("https://graph.facebook.com/v2.6/me/messages",
                    params={"access_token": ACCESS_TOKEN},
                    data=json.dumps({
                        "recipient": {
                            "id": sender_id
                        },
                        "message": {
                            "text": "What do you wish to eat today? You can browse our menu with the button below."
                                    " Alternatively, you can tell me the food/beverage that you want, with the quantity behind it. "
                                    "For example, 2 char kway teow and 1 milk tea",
                            "quick_replies": [
                                {
                                    "content_type": "text",
                                    "title": "Menu",
                                    "payload": "Menu"
                                }
                            ]
                        }
                    }),
                    headers={'Content-type': 'application/json'})
    return 'OK'
    if r.status_code != requests.codes.ok:
        print(r.text)
```

Figure 4.2.1.1 Greeting Message with Quick Reply

4.2.1.1 Details of Food

The integration with Facebook Messenger was done with the one-click integration feature provided by Dialogflow. There are several features in Facebook Messenger

that could be used in structuring the reply messages such as, messaging templates, rich media like audio, video and images as responses instead of only just text response. Such features enrich the conversation experience by making the experience is more engaging and interactive with the visual aesthetics.

In this project, the generic messaging template, quick replies, carousel list and text response were being used. The generic template allows basic cards to be output to the user where each card comprises of components such as the name, image, price and description of the food item as reflected in Figure 4.2.1.2. At the same time, a button will be integrated below each card that allows the user to order the item they desire just by clicking the "Order now" button. For future developers that wish to replicate messages that pertains to quick replies, basic cards, carousel list and text response, they can refer to Appendix B for the template.

```
"message": {
    "attachment": {
        "type": "template",
        "payload": {
            "template_type": "generic",
            "elements": [
                {
                    "title": (info[0]+" : $"+str(info[1])),
                    "image_url": info[3],
                    "subtitle": (info[2] + " The price of the food is " + str(info[1])),
                    "default_action": {
                        "type": "web_url",
                        "url": info[3],
                        "webview_height_ratio": "tall",
                    },
                    "buttons": [
                        {
                            "type": "postback",
                            "title": "Order now",
                            "payload": info[0]
                        }
                    ]
                }
            ]
        }
    }
}
```

Figure 4.2.1.2 Basic Card Containing the Details of the Food Item

Upon asking for the details of the food item with queries as reflected in Figure 4.2.1.3, the message template as a JSON object in Figure 4.2.1.2 will be sent to the Facebook server. The details of the food item queried by the user does not pertain to main dishes only. It similarly works for the side dishes and drinks categories. To detect which category the user is querying, the entities needs to be annotated in DialogFlow by highlighting the food item as seen in Figure 4.2.1.3. This will hence aid in the identification of the correct intent upon the webhook call. The description of the food,

drinks and sides can be detected accordingly to the intents, ask_food_description, ask_drinks_description and ask_sides_description respectively.

Figure 4.2.1.3 Ways to Ask What the Food Item is

4.2.1.2 Menu

The menu of the foodchatbot can be retrieved by establishing a connection to the ClearDB database. There are two types of menu being implemented in this system where user can view the pictorial menu or the textual menu. By querying the phrases as seen in Figure 4.2.1.2.1, the consolidated menu in text will be replied to the user in the messenger since the intent "ask_menu_word" in Dialogflow was detected. This textual menu is implemented by consolidating all the items details that was cached upon loading of the server as seen in Figure 4.2.1.2.2. The different categories are separated by newline using "u\000A" rather than "\n" as "\n" is not the correct syntax to create a new line in Facebook messenger platform.

Figure 4.2.1.2.1 Ways to Query a Text Menu

```
elif intentofDF == "ask_menu_word":
    menu = "------------------------------"
    menu += "\u000A"
    menu += "Food:"
    menu += "\u000A"
    menu += "------------------------------"
    menu += "\u000A"
    menu += listoffood
    menu += "\u000A"
    menu += "------------------------------"
    menu += "\u000A"
    menu += "Drinks:"
    menu += "\u000A"
    menu += "------------------------------"
    menu += "\u000A"
    menu += listofdrinks
    menu += "\u000A"
    menu += "------------------------------"
    menu += "\u000A"
    menu += "Sides:"
    menu += "\u000A"
    menu += "------------------------------"
    menu += "\u000A"
    menu += listofsides
    reply(sender, menu)
```

Figure 4.2.1.2.2 Implementation of Consolidated Text Menu

Similarly, the list of food, drinks or sides can be shown by using commands such as "list of food", "list of drinks" and "list of sides" respectively. These commands will incur the intents list_food_by_cuisine, list_drink and list_side_dishes to be identified respectively. The output shown for the list of items is based on a feature in Facebook Messenger call Carousel List. Carousel list allows user to scroll left and right in both the mobile and desktop platform. Upon coming to the end of the carousel list that supports up to a maximum of 10 cards, there is a need to implement the function that allows users to navigate to the new page since there is more than 9 items in the menu. The linking between the pages with the "View More food" card has to be created yourself as the template documented in the Facebook Developer website [18] do not come with that field and function. Hence this function can be implemented by making the last card to be a card that sends back a postback of the food page number required to our backend server for processing, evident in Appendix B.

The purpose of creating a consolidated menu is to provide the users a more concise menu if they know exactly what kind of item they are seeking out for. This will hence

save them the hassle of scrolling through the long list of items and navigating through the pages to find out what they wish to order.

Since there are a myriad of items in the menu, the information pertaining to the items are being loaded upon initialization of the server. This thus allow the chatbot to be able to reply faster and more efficiently as the action to retrieve the data from the database has now been eliminated.

4.2.1.3 Summary of Order

Upon ordering and confirming the order of the food, the user will be shown a summary of the order that he or she made. This summary reflects the items ordered, and the calculated price for each item at every row. The summary will not only include individual pricing of each items, but also the total price of the whole order at the bottom of the summary. The user will be provided their order number to facilitate in the collection and payment of their food at the food counter. The total price and list of ordered food for individual users, should always be re-instantiated upon ending each order. This will prevent the items and its prices from being stored in the server, which will lead to them being used for calculating the next total price in the next order summary.

4.2.2.1 Dialogflow Small Talks

To make the chatbot more human-like, a feature of Dialogflow, Small Talk, is being implemented in the Chatbot. This would intensively enhance the way users can interact with the machine where the interaction will be more natural and intuitive.

There are a few downsides to implementing this feature as the Chatbot can only understand certain questions. Should the users communicated with open-ended questions, it will not be able to produce a response since it do not understand what the users want. This situation arises due to the underlying implementation approach, the rule-based approach where the chatbot reacts accordingly only to the phrases it was being trained with. Thus, to minimize such situations, the replies of the chatbot for the small talk feature needs to be streamlined to guide the user in producing the response that it wants.

5. System Demonstration

5.1   Overview

Process

1	Change query to lower case
2	Remove punctuations, special characters, multiple spaces
3	Change numbers in word forms to integers
4	Spelling Checking
5	Lemmatization
6	Remove Stopwords
7	Find entities of query

Input

Load model
(Server)

Output

Quantity
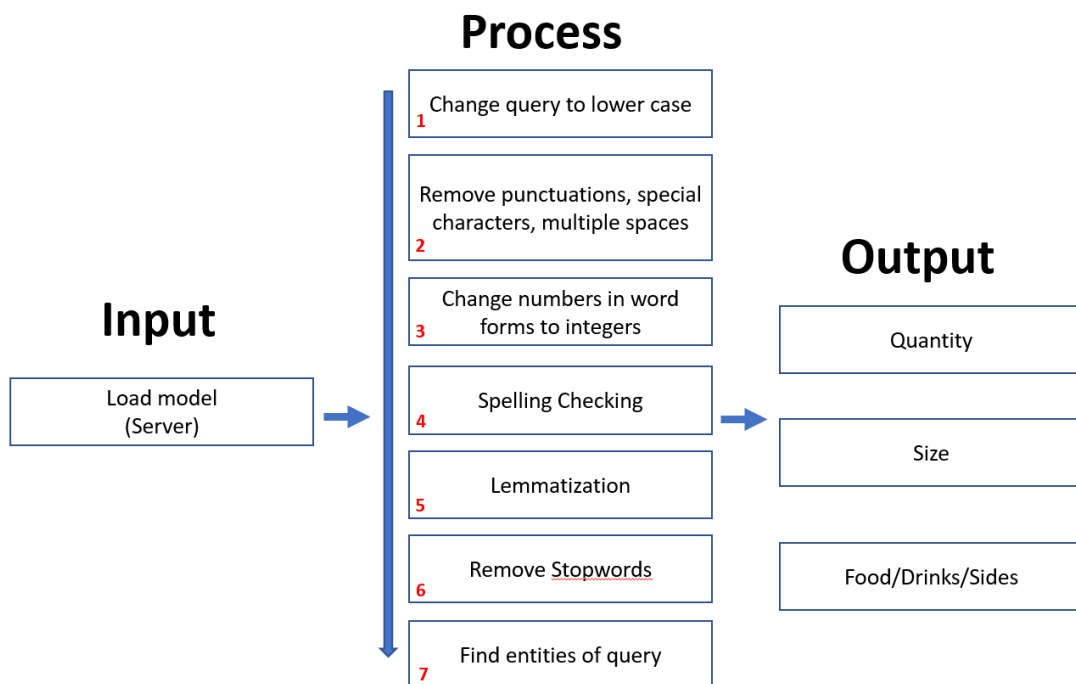
Size

Food/Drinks/Sides

Figure 5.5.1 Overview of Complex Query Processing

The overview of the system begins with the loading of the Spacy model to cloud service provider, Heroku. Following that, data received from the Facebook messenger will be passed to Dialogflow to extract the intents. Should the intent be "ComplexQuery", the data will be processed using the Spacy framework coupled with the Regex and SymSpell libraries as seen in Figure 5.1.1. The processing follows a sequential notion where the order of processing starts from 1, which changes the Facebook messenger's query to lower case, followed by 2, which removes punctuations, special characters and multiple spaces. This processing will continue with 3, changing the numbers in word forms to integers, 4, spelling checking, 5, lemmatization, 6, removing stopwords and lastly 7, the identification of entities based on the query. The results of the processing would then be output as a JSON object containing the quantity, size and item's name back to user interface on the Facebook messenger platform.

5.2　　　Server Process

Before deciding what functions to be used, a check will be done on the diagnostic info of Dialogflow to see which intents is being detected based on the user's queries. The detected intent will be used as a deciding factor in choosing the functions used for generating a response back to the user.

For the example of an user complex query, "9 stiks of large oyster omelette   three small frys four lage briyani 5 smal soup six meium char kay tew", the server will first change the query into all lower case. After that, all the special characters, symbols, punctuations and multiple trailing whitespaces will be removed. This will transform the query to, "9 stiks of large oyster omelette three small frys four lage briyani 5 smal soup six meium char kay tew". Following which, it will map the word forms of numbers into numerical form, changing the query to "9 stiks of large oyster omelette 3 small frys 4 lage briyani 5 smal soup 6 meium char kay tew". Spelling checker will next change the query into, "9 sticks of large oyster omelette 3 small fries 4 large briyani 5 small soup 6 medium char kway teow". Lemmatization and stopwords removal that includes changing the singularity of the query to singular terms, will be performed to transform the query to "9 oyster omelette 3 fries 4 biryani 5 soup 6 char kway teow". Based on this result, the entities will be extracted as seen in Figure 5.2.1, before returning a response that repeats the items and quantity of the items that the user wish to order. The size of the dish is removed as currently all the dishes has only one price associated with it in the database. Upon confirming the order with the user, the user will be shown a summary of their order shown in Figure 5.3.8.

```
2018-10-20T14:17:48.364073+00:00 app[web.1]: Entities [('9', 'QUANTITY'), ('large', 'SIZE'), ('oyster omelette', 'FOOD
'), ('3', 'QUANTITY'), ('small', 'SIZE'), ('fries', 'SIDES'), ('4', 'QUANTITY'), ('large', 'SIZE'), ('biryani', 'FOOD'
), ('5', 'QUANTITY'), ('small', 'SIZE'), ('soup', 'SIDES'), ('6', 'QUANTITY'), ('medium', 'SIZE'), ('char kway teow',
'FOOD')]
```

Figure 5.2.1 Entities Extracted from Query

5.3　　　Front-End

Upon sending message to the food ordering chatbot for the first time, the user will see the page name with a "Get Started" button below as seen in Figure 5.3.1. To begin using the chatbot, users can simple click on the "Get Started button".

45

Figure 5.3.1 Greeting Message with Quick Reply

The following message with a Quick Reply button at the bottom of the screen as shown in Figure 5.3.2, will then be displayed.
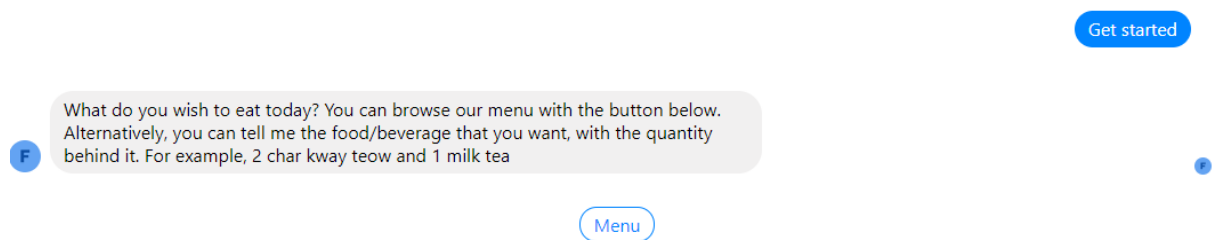

Figure 5.3.2 Greeting Message with Quick Reply

The user can proceed to click the quick reply "Menu" button to see the pictorial menu of the canteen. This pictorial menu is shown as a carousel list where user can navigate left and right for more items as seen in Figure 5.3.3 and Figure 5.3.4.
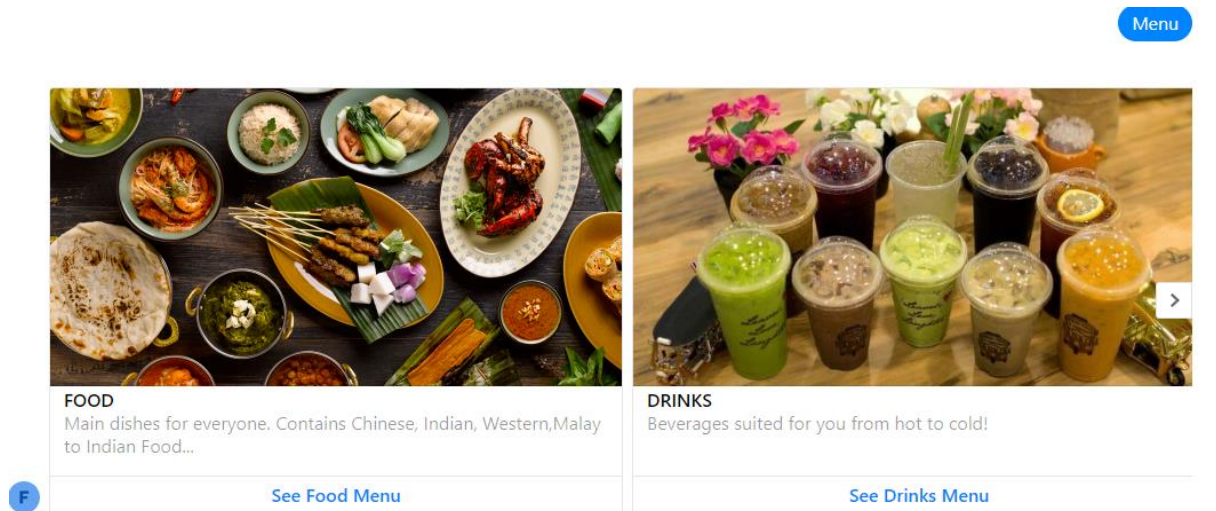
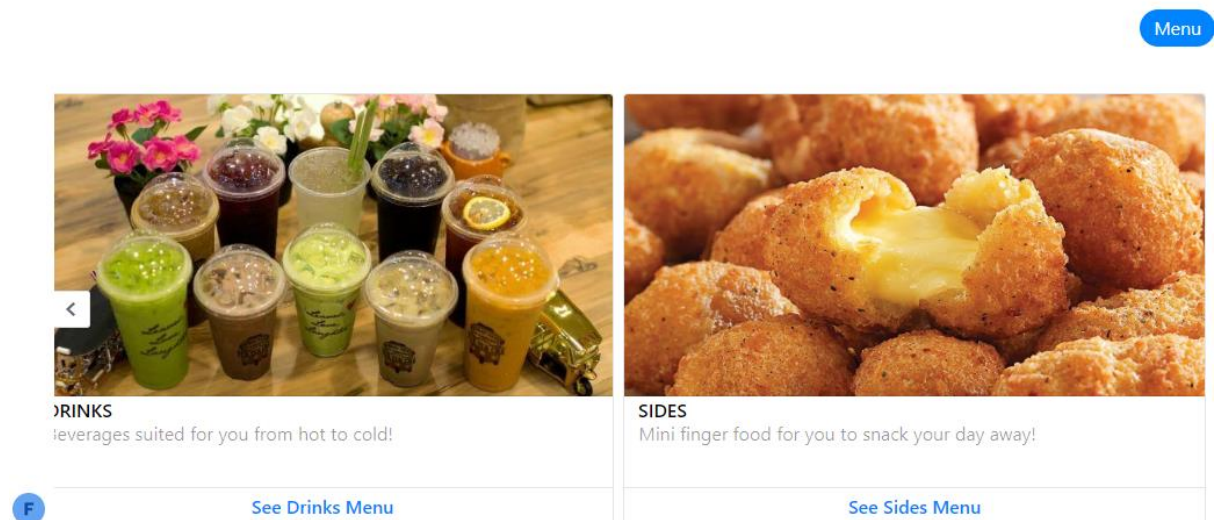*Figure 5.3.3 Carousel List Showing Food and Drinks Category*

Figure 5.3.4 Carousel List Showing Drinks and Sides Category

Should the user know what they wish to order but cannot remember the exact full name of the item, they can directly query "menu without pictures" to get the consolidated text menu in Figure 5.3.5.

```
----------------------------
Food:
----------------------------
aglio olio chicken chop
aglio olio chicken cutlet
bak chor mee
bak kut teh
bee hoon
beef
biryani
bun bo hue
char kway teow
chicken burger
chicken rice
crabs
curry fish head
dim sum
fish burger
fish n chips
fried carrot cake
fried hokkien mee
hokkien char mee
hokkien prawn mee
laksa
mee siam
nasi lemak
noodle
oyster omelette
pasta
pho
pho bo
rojak
roti prata
satay
spaghetti with chicken chop
spaghetti with chicken cutlet
spaghetti with ham and mushroom
spaghetti with sausage
wanton mee


----------------------------
Drinks:
----------------------------
beer gao
coffee
iced coffee
iced milk tea
iced milo
kopi
lao hor
milk tea
milo
milo bing
soy bean
tea
teh
teh bing


----------------------------
Sides:
----------------------------
fries
mash potato
salad
soup
spaghetti
toast cube
```

Figure 5.3.5 Consolidated Text Menu

By clicking the "See Food Menu" in Figure 5.3.3, a carousel list of food items in the food menu will be presented to the user. This list will contain multiple basic cards of the items in the food category. Each card contains a picture, description and a "Order now" button at the bottom of it similarly to the food item details card in Figure 5.3.6. The user can scroll to the last card of the list and click the "View More food" button, which will make the chatbot return the next page of food related to the item category that they chose, back to them. This viewing of next page function will only be present if there are more than 9 items in the chosen category. Similarly, if user click the "See Drinks Menu" or "See Sides Menu" button in Figure 5.3.4, the carousel list of drinks and sides respectively will be returned to them.

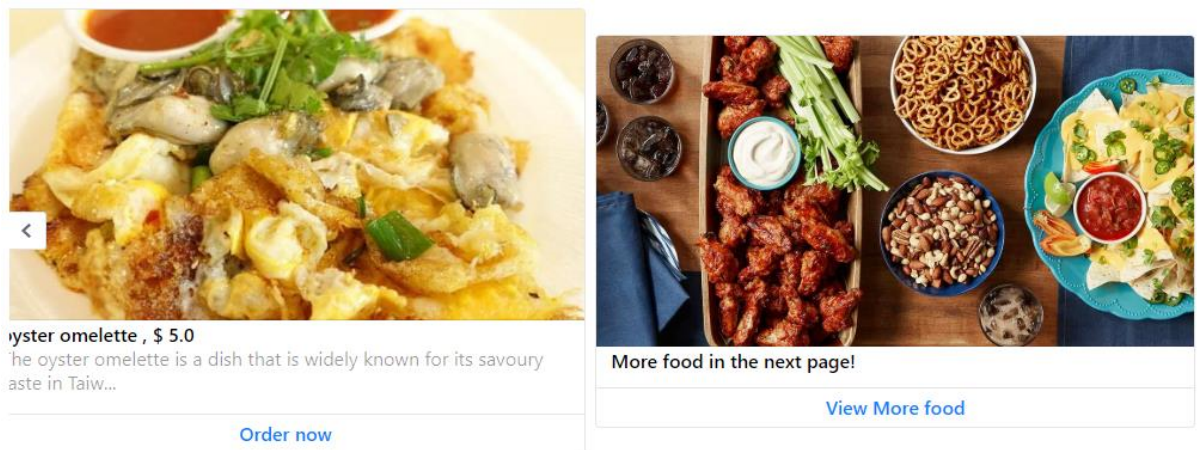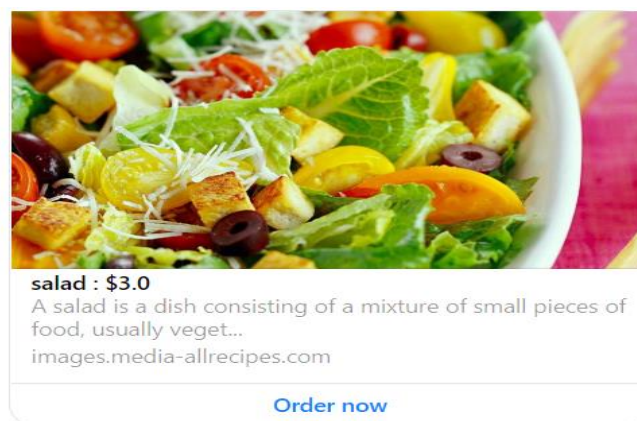*Figure 5.3.6 Food Menu with Pictures*

If the user wishes to find out more information about the details of the item, they can query the name of the item following the sentence pattern, "what is a <name of the item>?". For example, typing the query, "what is a salad?", will return the basic card in Figure 5.3.7, that shares the same properties as the cards in the carousel list of the

menu.

49

If the user clicks the "Order now" button of the basic card of the item regardless whether in the menu or upon enquiring the information of the item, they will be prompted on the quantity they wished to order for that item. Following up on the quantity input by the user, a summary of the order with order number will be generated for the user. An example of this summary can be seen in Figure 5.3.8 below.
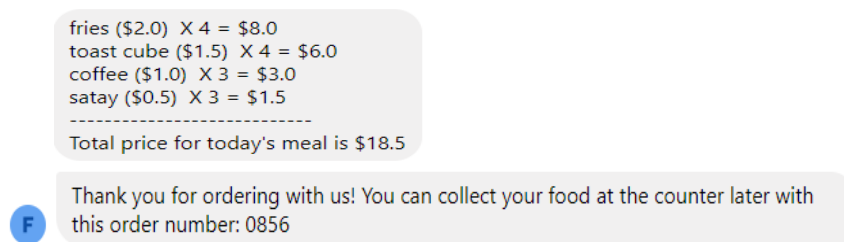
```
fries ($2.0)  X 4 = $8.0
toast cube ($1.5)  X 4 = $6.0
coffee ($1.0)  X 3 = $3.0
satay ($0.5)  X 3 = $1.5
---------------------------
Total price for today's meal is $18.5
```

Thank you for ordering with us! You can collect your food at the counter later with
F   this order number: 0856

Figure 5.3.8 Order Summary

To engage the user, the chatbot is integrated with small talks to make it more interactive. Since the chatbot is unable to handle open handle questions and queries not related to food ordering, the conversation will be streamlined towards the ordering of food. The users will be prompted by the Chatbot to converse their orders on what they wish to order instead of chatting about other topics as seen in Figure 5.3.9.

I am bored

So am i 🙁 How about you tell me what you want to eat for your meal later? I will
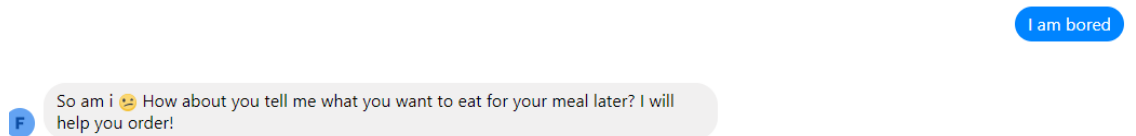F   help you order!

Figure 5.3.9 Small Talks Feature

6. Conclusions and Future Work

6.1. Conclusion

A food ordering Chatbot in the domain of Named Entity Recognition (NER), was developed in this project. This Chatbot is a proof-of-concept prototype for entity extraction with concept ontology.

This Chatbot is deployed on the Facebook Messenger platform integrated with the framework, Dialogflow. The technologies employed in this Chatbot aims to target queries that are deemed too complex for the system to process for the food in the Singapore context.

This Chatbot has its NER model built from scratch using the Spacy library with customised training data pertaining to Singapore local cuisines. It does its processing of queries based on its own customised list of spelling correction words, stopwords and tokenization list.

The Chatbot allows processing of complex queries, which are queries that contains multiple items, size and quantities. At the same time, it helps to address situations where user's queries contain special symbols, punctuations and multiple trailing blank spaces.

This Chatbot supports the ordering of the food both in English and Singlish, a Colloquial Singaporean English that contains words not found in the English dictionary. Based on the food items queried by the users in the Facebook Messenger, the Chatbot can extract out the entities of the items that identifies the name, size, quantity and category of food the item falls into.

The Chatbot has functions that displays the consolidated and categorical menu of the food in NTU North Spine Canteen A. To encourage the usage of this Chatbot, the Chatbot was made more interactive and human-like with the Small Talks feature integrated in it. This will intensively enhance the user experience in interacting with machines and at the same time, help to streamline user's messages to food ordering context.

This Chatbot was designed and implemented with the software development principle, re-usability in mind. It is highly modular and has low coupling with high cohesion for its system architecture which makes it extensible. This development of this Chatbot is well documented which allows future developers to replicate this project easily.

6.2. Recommendations for Future Work

Due to the complexity and large scale of the system, some areas may not be fully examined and there exist some limitations in the current system. Future work can be examined in the following directions:

1. Ontology development for Named Entity Recognition
2. Functionality improvement of the Chatbot

6.2.1 Ontology Development

A simplified NER ontology was manually crafted and deployed on Heroku for this project. Future researchers can work on the development of the ontology such as the development process, algorithms and methodologies associated with NER for building ontologies.

More research can be done on the free-tier cloud service providers that can accommodate to the large memory size trained model and corpus for named entity recognition. This is important in making the user experience more satisfying since the Heroku server goes into sleep mode for the free tier after 30 minutes of inactivity from the users.

The vision of the Named Entity Recognition is to allow relevant data to be extracted and categorized for further usage from unstructured textual content. Ontology is the crux for adding rich semantics to corresponding resources. The development of advanced ontologies would gain increasing importance with the proliferation and assimilation of data in everything we do in the upcoming Global Technology Revolution 2020.

6.2.2 Functionality Improvement

The Chatbot was developed by following the open/closed software principle, which allows future developers to add on new features to it without modifying its source code. Future developers can enrich the Chatbot by integrating new functionalities such as:

- Real-time autocomplete word prediction that helps to complete a user's queries without the need for them to type out the whole sentence.
- Slot-Filling
- Letting users have a choice to customize their food orders (More or less rice, chilli, food dishes)
- Food promotions
- Better visualization for showing the menu of items
- Payment methods for the orders
- Shopping Cart for storing the items
- Development of an IOS or Android application catered for the ChatBot

References

[1] D.Marcus. Messenger Platform gets an update!
https://www.facebook.com/notes/david-marcus/messenger-platform-gets-an-update/10155014173359148/, June 2016. [Online; accessed 20-Oct-2018].

[2] 80% of businesses want chatbots by 2020.
http://www.businessinsider.com/80-of-businesses-want-chatbots-by-2020-2016-12, Dec 2016. [Online; accessed 20-Oct-2018].

[3] From robotic process automation to intelligent automation, p2-3.

[Online; accessed 20-Oct-2018].

[4] M. Chui, S. Lund and S. Ramaswamy. What's now and next in analytics, AI, and automation. Mckinsey & Company, May 2017

[5] B. David. Applying Today's Automation, Convenience Store Decisions, Apr 2017, p58-61 [Online; accessed 20-Oct-2018].

[6] A. Følstad, P. Brandtzaeg. Chatbots – the new world of HCI, ACM Interactions, Aug 2017, p38

[7] A. Følstad, P. Brandtzaeg. Why people use chatbots, 4th International Conference on Internet Science, Nov 2017

[8] B. Jaekel. Wingstop soars with conversational commerce via chatbot initiative", Retail Daily, Jun 2016 [Online; accessed 20-Oct-2018].

[9] M. Hennessy. Chatbots Are Changing How Customers Order, QSR Magazine, Nov 2016 [Online; accessed 20-Oct-2018].

[10] Daniel. B, Adrian. H.M and Florian.M, "Evaluating Natural Language Understanding Services for Conversational Question Answering Systems", SIGDIAL 2017 conference, Aug 2017

[11] A Review of Named Entity Recognition (NER) Using Automatic Summarization of Resumes. https://towardsdatascience.com/a-review-of-named-entity-recognition-ner-using-automatic-summarization-of-resumes-5248a75de175, Jul 2018. [Online; accessed 20-Oct-2018].

[12] Robert. NLTK vs. spaCy: Natural Language Processing in Python, Apr 2016. [Online; accessed 20-Oct-2018].

[13] M.Rouse. Heroku. https://searchsalesforce.techtarget.com/definition/Heroku, Jun 2012. [Online; accessed 20-Oct-2018].

[14] The End of Heroku? https://hackernoon.com/the-end-of-heroku-cda380b3087f, Jun 2017. [Online; accessed 20-Oct-2018].

[15] Optimizing Dyno Usage. https://devcenter.heroku.com/articles/optimizing-dyno-usage, April 2018. [Online; accessed 20-Oct-2018].

[16] W.Garbe. SymSpell vs. BK-tree: 100x faster fuzzy string search & spell checking. https://towardsdatascience.com/symspell-vs-bk-tree-100x-faster-fuzzy-string-search-spell-checking-c4f10d80a078, July 2017. [Online; accessed 20-Oct-2018].

[17] W.Garbe. SymSpell. https://github.com/wolfgarbe/SymSpell, Apr 2018 [Online; accessed 20-Oct-2018].

[18] Generic Template. https://developers.facebook.com/docs/messenger-platform/send-messages/template/generic [Online; accessed 20-Oct-2018].

[19] Why ClearDB? http://w2.cleardb.net/why-cleardb/ [Online; accessed 20-Oct-2018].

Appendices

Appendix A

Environment Setup for Application Development

This appendix states the environment setup procedures used in the development of the application. The procedures stated below are based on the Windows 10 Home platform, 64-bit operating system with specifications of Intel Core i7 CPU@ 2.50 GHz, 12 GB RAM. For other operating systems or system specifications, users may follow similar procedures to set up the environment but with different commands and settings for installation.

A.1 Install Python 3

You may install the latest Python 3 version from Python's download page (https://www.python.org/downloads/). The Python package manager pip is included in Python 3.4 and above by default. The python version used for development in this project is version 3.6.4. The application will work perfectly fine with the latest version of python from the download page.

A.2 Creating a Git Repository

A.2.1 Installing Git

Visit Git's download page (https://git-scm.com/downloads) and install the latest version of Git.

A.2.2 Cloning Project from GitHub

Open command line and navigate to the directory where you wish to clone the repository. Next, run the command "git clone https://github.com/zhiyanggg/Chatbot.git ". The source code is now cloned into your local directory.

A.3 Cloud Service Provider Heroku

To create a new app named "example" in Heroku, install the Heroku CLI from the page, https://devcenter.heroku.com/articles/heroku-cli and run the following commands:

```
$ mkdir example
$ cd example
```

```
$ git init

$ heroku apps:create example
```

After running the above commands, the system will output:

```
Creating ⬡ example... done

https://example.herokuapp.com/ | https://git.heroku.com/example.git

Git remote heroku added
```

The command's output shows that the app will be available at http://example.herokuapp.com. The second URL, https://git.heroku.com/example.git, is the remote git repository URL; by default, the Heroku create command automatically adds a git remote named "heroku" pointing at this URL.

Next, extract all the contents in the folder "Chatbot", obtained from section A.2.1. Copy and paste all the extracted contents into the directory "example" where the git was initialised in this section.

A.4 Installing Required Python Packages

Before you are ready to go, you will need to install all the required python packages specified in requirements.txt. These are the packages necessary for the installation of technology stack mentioned in Section 3.5. Whenever you work on the project, make sure you are in the Python virtual environment created in Step B.3.3. You can go to the virtual environment by running the workon command, as mentioned earlier. After going to the environment, navigate to the project home directory myfyp/ by using cd command in PyCharm Terminal. Next, install all required packages by running the following command:

pip install -r requirements.txt

You may list all unused dependencies by running the command below and remove them using pip uninstall.

pip-autoremove –list

A.5. Database ClearDB

A.5.1 ClearDB Add-On for Heroku

Run the following command to add ClearDB to your application:

```
$ heroku addons:create cleardb:ignite

-----> Adding cleardb to sharp-mountain-4005... done, v18 (free)
```

Retrieve your database URL by issuing the following command:

```
$ heroku config | grep CLEARDB_DATABASE_URL

CLEARDB_DATABASE_URL => mysql://adffdadf2341:adf4234@us-cdbr-
east.cleardb.com/heroku_db?reconnect=true
```

Copy the value of the CLEARDB_DATABASE_URL config variable.

Based on the database url config variable, we can obtain the login credentials for usage in Mysql workbench. The login credentials are formatted as shown below.

```
CLEARDB_DATABASE_URL => mysql://[username]:[password]@[host]/[database
name]?reconnect=true
```

A.5.2 Cloning of Database

Install MySql WorkBench and login using the credentials obtain in Section A.5.1.

After logging into the server, head to the "Server" tab and select "Data Import". Choose the option "Import from Self-Contained File" and change the directory of this option to the directory where the chatbot.sql file is at. This .sql file should by right, be at the directory where the git was initialized in section A.2.1. Once the directory has been set, click on "Start Import" and it should start uploading the file.

A.6. Creating an App in Dialogflow

1. Go to the Actions on Google Developer Console.

2. Click on **Add/import project**, enter "YourAppName" for the project name, and click **Create Project**.
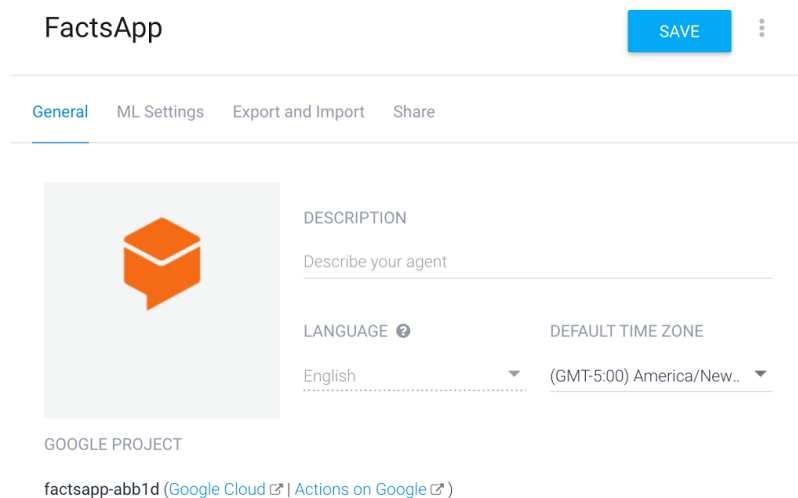
3. Click on **Skip** in the upper-right corner.

4. Click **Build > Actions** in the left nav.

5. Click **Add your first Action**

6. On the **Custom intent** card, click **Build**.

7. The Dialogflow console appears with information automatically populated in an agent. Click **Create** to save the agent.

After finishing the above 7 steps to create your agent, in the left navigation, click on the **gear icon** to the right of the agent name. Next, click on the **Export and Import** tab. Click **RESTORE FROM ZIP** and select the FoodOrder.zip file which is found when extracting the contents in section A.2.1.

Following that, type RESTORE in the text box, click **RESTORE**, then **DONE**. When the restoration is finished, the following screen evident below, appears. Take note of your Action project's ID; you'll need it to deploy the sample's fulfillment in the future.

A.7 Deploying Scripts to Cloud Service Provider

Open Command line and change the directory to the folder where the git was initialized in section A.2.1. Next, type the following commands in sequence to deploy the scripts to the Heroku Server. Firstly, to add all the relevant files to Git, run the command below.

git add .

Next, to save the changes to the local repository, run the command below.

git commit -m "first commit"

Lastly, to deploy the app to Heroku, we can push the code from the local repository's master branch to your Heroku remote with the command below.

git push heroku master

Appendix B Generic Messaging Template with Next Page Function

Future developers can follow this template to add the next page function in the last card, so that they can display more items rather than the restricted quantity of 9 items.

```
def foodpage1(accesstoken, sender):

  ACCESS_TOKEN = accesstoken

  r = requests.post("https://graph.facebook.com/v2.6/me/messages",

         params={"access_token": ACCESS_TOKEN},

         data=json.dumps({

           "recipient": {

             "id": sender

           },

           "message": {

             "attachment": {

               "type": "template",

               "payload": {

                 "template_type": "generic",

                 "elements": [

                   {

                     "title": "aglio olio chicken chop , $ 5.0",

                     "image_url": "http://4.bp.blogspot.com/-
7UwaNrcWZqI/TrFHDzzFHzI/AAAAAAAAKd0/UtaHvJhV0QE/s1600/Grilled+Chicken+
with+Pasta+Aglio+E+Olio+October+31st%252C+2011.jpg",

                     "subtitle": "This dish is made by lightly sauteeing sliced,
minced, or pressed garlic in olive oil, sometimes with the addition of dried red chili
flakes, and tossing with spaghetti topped with chicken chop",
```

```
        "buttons": [

          {

            "type": "postback",

            "title": "Order now",

            "payload": "aglio olio chicken chop"

          }

        ]

      },

      {

        "title": "aglio olio chicken cutlet , $ 5.0",

        "image_url": "http://cookdiary.net/wp-
content/uploads/images/Chicken-and-Noodles.jpg",

        "subtitle": "This dish is made by lightly sauteeing sliced,
minced, or pressed garlic in olive oil, sometimes with the addition of dried red chili
flakes, and tossing with spaghetti topped with chicken cutlet.",

        "buttons": [

          {

            "type": "postback",

            "title": "Order now",

            "payload": "aglio olio chicken cutlet"

          }

        ]

      },

      {


        "title": "bak chor mee , $ 4.0",
```

"image_url": "https://mtc1-dydfxmh.netdna-ssl.com/wp-content/uploads/2017/03/P1040774-1300x867.jpg",

"subtitle": "Bak Chor Mee (肉脞面) literally means "minced meat and noodles" in the Teochew dialect.",

"buttons": [

    {

        "type": "postback",

        "title": "Order now",

        "payload": "bak chor mee"

    }

]

},

{

    "title": "bak kut teh , $ 5.0",

    "image_url": "https://i.hungrygowhere.com/business/42/60/12/00/a2-bkt_594x0_crop_594x372_0668d8d8a6.jpg",

    "subtitle": "Singapore Teochew Bak Kut Teh is a version of pork ribs tea with a clear garlicky and peppery broth. Only a few ingredients and very easy to prepare.",

    "buttons": [

        {

            "type": "postback",

            "title": "Order now",

            "payload": "bak kut teh"

        }

    ]

                },

                {

                    "title" : "bee hoon , $ 1.2",

                    "image_url" : "https://s3-ap-southeast-1.amazonaws.com/afc-
prod/thumbnails/standard_mobile/5415/0286/1193/tn-HFM-Braised-Pig-Trotter-Bee-
Hoon.jpg",

                    "subtitle" : "Rice vermicelli are a thin form of rice noodles.
They are sometimes referred to as rice noodles, rice sticks, or bee hoon, but they
should not be confused with cellophane noodles which are a different Asian type of
vermicelli made from mung bean starch or",

                    "buttons": [

                        {

                            "type": "postback",

                            "title": "Order now",

                            "payload": "bee hoon"

                        }

                    ]

                },

                {

                    "title" : "beef , $ 4.0",

                    "image_url" : "https://www-cuisinesolutions-com-
production.s3.amazonaws.com/media/products/slicedbeef/original.jpg",

                    "subtitle" : "Beef is the culinary name for meat from cattle,
particularly skeletal muscle.",

                    "buttons": [

                        {

                            "type": "postback",

```json
                  "title": "Order now",

                  "payload": "beef"

               }

            ]

         },

         {

            "title" : "biryani , $ 4.0",

            "image_url" : "http://www.ndtv.com/cooks/images/mutton-
biryani-new.jpg",

            "subtitle" : "Biryani, also known as biriyani, biriani, birani or
briyani, 'spicy rice' is a South Asian mixed rice dish with its origins among the Muslims
of the Indian subcontinent.",

            "buttons": [

               {

                  "type": "postback",

                  "title": "Order now",

                  "payload": "biryani"

               }

            ]

         },

         {

            "title" : "bun bo hue , $ 5.0",

            "image_url" :
"https://media.foody.vn/res/g65/642530/prof/s/foody-mobile-ba-thu-jpg-537-
636255976625248994.jpg",

            "subtitle" : "Bún bò Huế or bún bò is a popular Vietnamese
soup containing rice vermicelli (bún) and beef (bò). Huế is a city in central Vietnam
```

associated with the cooking style of the former royal court. The dish is greatly admired for its balance of spicy, sour, sal",

```
            "buttons": [

                {

                    "type": "postback",

                    "title": "Order now",

                    "payload": "bun bo hue"

                }

            ]

        },

        {

            "title" : "char kway teow , $ 3.0",
```

"image_url" : "https://4scoin37ye-flywheel.netdna-ssl.com/wp-content/uploads/2014/03/charkwayteow2.jpg",

"subtitle" : "Char kway teow, literally 'stir-fried ricecake strips', is a popular noodle dish in Malaysia, Singapore, Brunei and Indonesia. The dish is considered a national favourite in Malaysia and Singapore",

```
            "buttons": [

                {

                    "type": "postback",

                    "title": "Order now",

                    "payload": "char kway teow"

                }

            ]

        },

        {

            "title" : "More food in the next page!",
```

```
                    "image_url": "https://i5.walmartimages.com/dfw/4ff9c6c9-
28f7/k2-_327d5c10-c756-40e0-90cf-d071d67b05a5.v2.jpg",

                        "buttons":[

                            {

                                "type": "postback",

                                "title": "View More food",

                                "payload": "View More food page 2"

                            }

                        ]

                    }

                }

            }

        }),

        headers={'Content-type': 'application/json'})

    if r.status_code != requests.codes.ok:

        print(r.text)
```

```python
def foodpage1(accesstoken, sender):
    ACCESS_TOKEN = accesstoken

    r = requests.post("https://graph.facebook.com/v2.6/me/messages",
        params={"access_token": ACCESS_TOKEN},
        data=json.dumps({
            "recipient": {
                "id": sender
            },
            "message": {
                "attachment": {
                    "type": "template",
                    "payload": {
                        "template_type": "generic",
                        "elements": [
                            {
                                "title": "aglio olio chicken chop , $ 5.0",
                                "image_url": "http://4.bp.blogspot.com/-
7UwaNrcWZqI/TrFHDzzFHzI/AAAAAAAAKd0/UtaHvJhV0QE/s1600/Grilled+Chicken+with+Pasta+Aglio
+E+Olio+October+31st%252C+2011.jpg",
                                "subtitle": "This dish is made by lightly sauteeing sliced, minced, or pressed
garlic in olive oil, sometimes with the addition of dried red chili flakes, and tossing with spaghetti
topped with chicken chop",
                                "buttons": [
                                    {
                                        "type": "postback",
                                        "title": "Order now",
                                        "payload": "aglio olio chicken chop"
                                    }
                                ]
                            },
                            {
                                "title": "aglio olio chicken cutlet , $ 5.0",
```