

# CA1 – Food Search & Customer Analytics

30-Mar-2020 | Version 1.0

## Team 7

Alfred Tay Wenjie  
Li Xinlin  
Wang Zilong  
Xu Jiachen

Prepared for NUS-ISS Master of  
Technology in Intelligent Systems  
EBA5004 Practical Language  
Processing Group Project

# Table of Contents

Table of Contents	1
1. Introduction	2
2. Business Model and Target Market	2
3. Datasets	3
4. Solution Overview	3
5. Conversational User Interface	4
6. Sentiment and Aspect Mining	6
7. Customer Sentiment Analysis Dashboard	11
8. Information Extraction	11
9. Knowledge Graph	15
10. Fake Review Filtering	19
11. Technical Implementation Details	24
12. Conclusion	30
Reference	31
Annex A – MISC Supporting Files	32
Annex B – User Guide	33
Annex C – Test Scenarios	40
Annex D – Team Members Individual Reflection	51

## 1. Introduction

### 1.1 Business Opportunity

With the availability of information at their fingertips (through their smart phone), many consumers rely on online restaurant reviews to select their next dining experience. However, with the huge number of reviews, it is impossible to read through all the reviews before deciding on the perfect dining place for that very important date/meeting. Moreover, the Internet is littered with fake reviews designed to mislead consumers.

Similar information overload challenge exists for restaurant owners who wants to know what consumers are talking about their restaurant. It is simply too time consuming to manually gather, read through and compile the online reviews into actionable insights to improve their business.

### 1.2 Proposed Solution

The team proposes an Intelligent Language Processing System which identify and filter fake online restaurant and food reviews, perform information extraction and sentiment and aspect mining and provide relevant and concise information to Consumers through a conversational user interface and to Restaurant Owners through a customer sentiment analysis dashboard.

## 2. Business Model and Target Market

### 2.1 Business Model

The business model is for this service to be provided free of charge to consumers. The revenue will be generated from the \$19.90 monthly subscription fee for the customer sentiment analysis dashboard.

For restaurants with multiple outlets, a discounted price of \$9.90 will be offered for second and subsequent outlets.

### 2.2 Target Markets

The service will first be launched and tested in the local market where there is more familiarity. Feedbacks will be sought from users to improve the service before launching phase II to introduce the service to the rest of Asia. In phase II, the operation, sales and maintenance processes will be fine-tuned before embarking on phase III to make available the service to the rest of the world.

Phases	Phase Launch Schedule	Country/Region	Target Number Subscriber	Estimated Yearly Revenue
I	X	Singapore	1,000	\$238,800
II	X + 6 months	Asia	60,000	\$14.32 Million
III	X + 18 months	Global	100,000	\$23.88 Million

### 2.3 Potential Partners

Partnership agreements with food delivery services and restaurant booking services can be explored to deliver a seamless end-to-end experience for consumers.

### 3. Datasets

Tripadvisor and Yelp Reviews data are used for this project. Octoparse scraping software is used to automatically extract Singapore restaurants data from Tripadvisor.com.

Name	Rows	Columns	Description
Tripadvisor Restaurants	5,000	restaurant name, link, cuisine, overall rating, address, phone number, coordinate, and description	Basic information of 5000 restaurants. From Tripadvisor.com
Tripadvisor Reviews	20,000	restaurant name, reviews text, review ratings, user id, user comment, user helpful comment, and review time.	Reviews text and user behavioral information. From Tripadvisor.com
Yelp Reviews I	19,000	Reviews text	Raw review from Yelp Public Dataset without label
Yelp Reviews II	1,000	Reviews text, polarity	Reviews with labels. Created by Dimitrios Kotzias from UCI ML Repository.

### 4. Solution Overview

The high-level solution overview is depicted in figure 1. The system can be logically grouped into 3 main processes (from right to left): (1) Data Collection, Pre-processing and Filtering, (2) Information Extraction and Sentiment Mining, (3) Front End User Interface. The main natural language processing modules (from left to right) namely, (a) Conversational UI, (b) Sentiment and Aspect Mining (c) Information Extraction and (d) Fake Review Filtering are coloured in dark purple and will be discussed in more detail in subsequent sections.

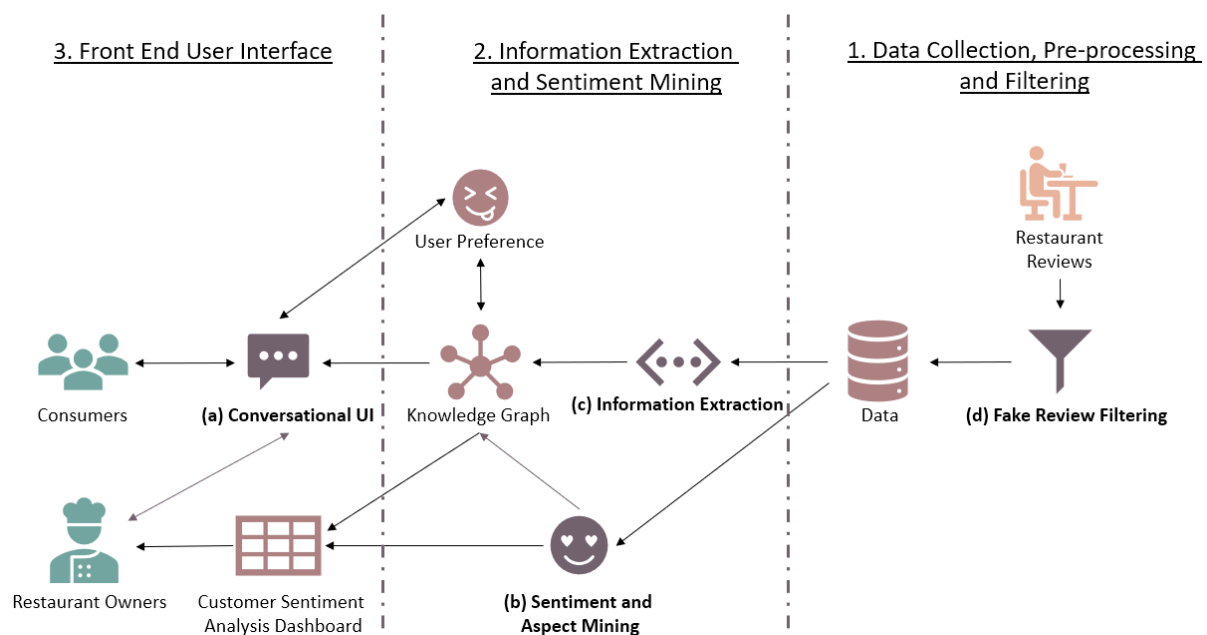


Figure 1: Solution Overview Diagram

In the Data Collection, Pre-processing and Filtering step, Restaurant Review data is crawled from the Internet using 3<sup>rd</sup> Party Robotic Process Automation software. The data is pre-processed before being passed into a Fake Review Filter to remove reviews which are likely to fake. Thereafter, the data is

stored in a structured database for downstream use. Information Extraction is performed to extract dishes names mentioned in the reviews. The extracted information together with restaurant name, cuisine and address are stored into a knowledge graph data structure to facilitate fast query response for the chatbot. Sentiment and Aspect Mining is performed to extract consumer sentiments on taste of food, perceived value in terms of price, quality of service and ambience of the restaurant. The mining results are formatted into an online dashboard to provide actionable insights and sentiment overview for restaurant owners and managers for each of their restaurants/outlets. The conversational UI, in the form of a chatbot interacts with the consumers and help them search for restaurants based on their requirements. The chatbot proactively ask for user feedbacks after the consumers have patronised a restaurant. It also has a feature to initiate a conversation to recommend restaurants to consumers based on their preference which is also stored in the knowledge graph.

## 5. Conversational User Interface

A custom-built task-oriented specialist chatbot has been developed to help consumers look for the most suitable restaurants for their dining needs. Figure 2 depicts the main components in the chatbot module. The implementation deploys a 2-layer intent detection architecture to take advantage of the high accuracy of the Finite State Transducer and the wide coverage of machine learning intent classification. A unique feature of this chatbot is that it's able to self-learn and improve its intent detection performance through interacting with users. This means that as more people interact with the chatbot, the chatbot automatically gets better at identifying the intent correctly. The details of how this works will be elaborated below.

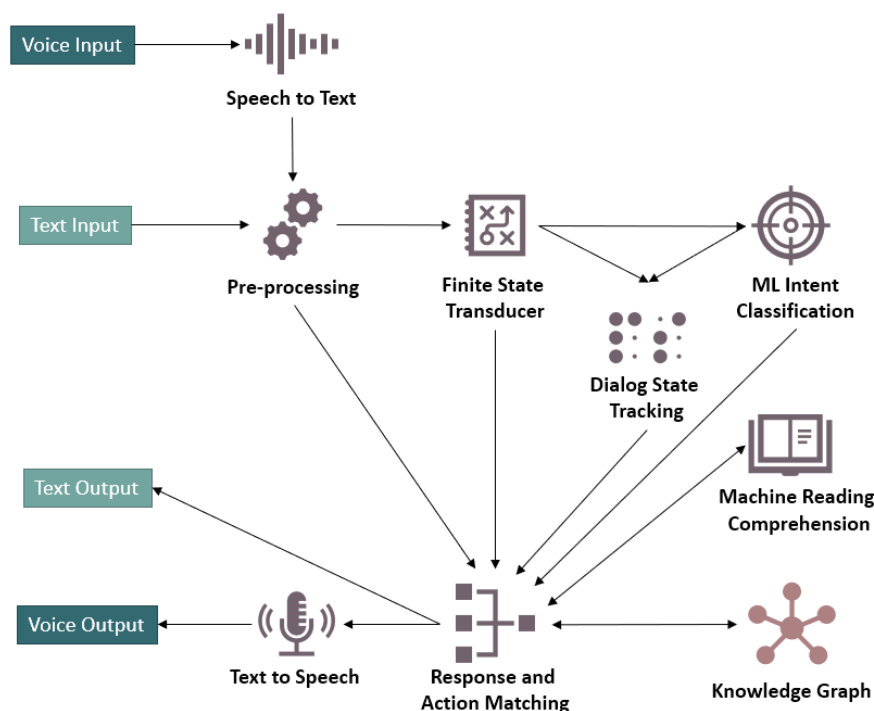


Figure 2: Chatbot Internal Components

**Intent Detection** – As mentioned above, there are 2-layers of intent detection. The first layer deploys Finite State Transducer where conceivable phrases are handcrafted into various possible paths. This first layer should be able to identify at least 80% of genuine use cases. Utterance which cannot be identified by the Finite State Transducer will be sent to the second layer Machine Learning Intent Classifier. Figure 3 below shows a small portion of the Finite State Transducer for the chatbot.

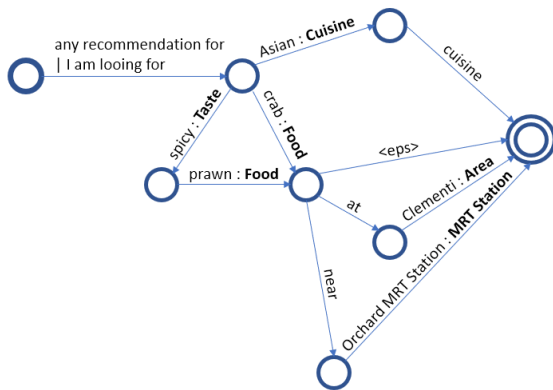


Figure 3 – Portion of FST

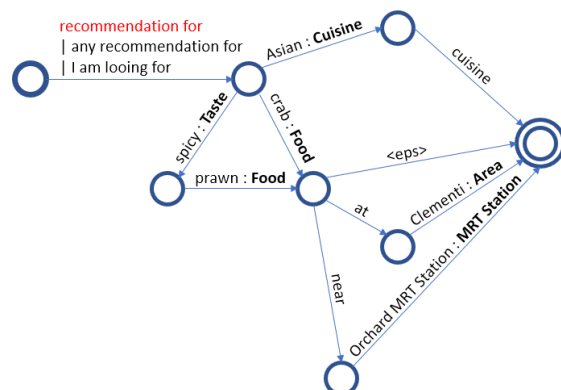


Figure 4 – FST update example

**Machine Learning Intent Classification** – A Random Forest Model is trained with feature engineered inputs to learn to classify the utterance into intent or intent groups. The inputs to the model include presence or absence of various entities (e.g. Food, Cuisine, Area) and presence or absence of unigram words in the training sentences. For utterance which are classified into intent groups, the specific intent may be further identified by applying rule-based presence or absence of certain entities (for example if ‘taste’ entity is present in a search for restaurant by food intent group, it means that the user wants to search by both food and taste) .

**Self-Learning Capability** – After an utterance has been classified by the machine learning model, the chatbot will check with the user to confirm that the intent has been classified correctly. If so, the appropriate path at the Finite State Transducer will be automatically updated with the previously unseen pattern (see example in figure 4 after chatbot Test Scenario 5 in annex C is executed) and the conversation flow with the user will continue. Thereafter, when any user uses similar utterance, the intent will be detected correctly without the need for clarification.

**Slot Extraction** – Slot extraction is accomplished using Finite State Transducer, dictionary approach and a small bit of regular expressions. The main entities to be extracted are: food (name of dishes), cuisine type, name of area, name of MRT stations, taste of food and restaurant name, which are distinctive and easily picked up using a dictionary of the possible words for each entity. Regular expression is used to pick up numeric values such as time and number of pax for restaurant reservation.

**Machine Reading Comprehension** – Machine Reading Comprehension using a pre-trained model [1] is deployed to answer general questions on Singapore food culture. The data source is from a Wikipedia page on Singapore Cuisine ([https://en.wikipedia.org/wiki/Singaporean\\_cuisine](https://en.wikipedia.org/wiki/Singaporean_cuisine)).

**Dialog State Tracking** – Dialog state tracking is implemented to track and maintain the context of the conversation. It is also used to control the conversation flow in order to get all the user’s requirements and feedbacks. A total of 27 states and their possible paths has been defined and implemented.

**Response and Action Matching** – The Response and Action Matching module acts as a central unit to match responses and trigger search actions or queries to Machine Reading Comprehension model based on user utterance, detected intent and current dialog state.

**Proactive Engagement** – Besides responding to user-initiated conversations, the chatbot has two proactive user engagement scenarios. The first one is getting feedback from users after he/she has made a reservation at and patronised a particular restaurant. The feedback could be added as another data point in the customer sentiment dashboard. The second one is making restaurant recommendations to users based on what the user had searched for in the past. The search parameter is selected from the list of previous search terms using roulette wheel selection method giving higher probability to more frequent search terms. The assumption is that if the user frequently searched for a particular dish or a particular area, it is very likely that he is interested to know about similar dining options.

**Adaptable Persona** – The chatbot has two persona types (casual and formal) to mirror the conversational style of the user. The two persona types are adapted based on the style of greeting the user starts the conversation with. This mirroring technique aims to make the users feel more comfortable talking to the chatbot.

**Pre-processing** – Simple pre-processing is done to convert user utterance to all small letters so that matching of phrases in the Finite State Transducer is not affected by case sensitivity. There were considerations to perform spell checks, but it would wrongly correct entity names which are not in the English dictionary.

## 6. Sentiment and Aspect Mining

Figure 5 shows the sentiment and aspect mining workflow from pre-processing to presenting actionable insights to eatery owners.

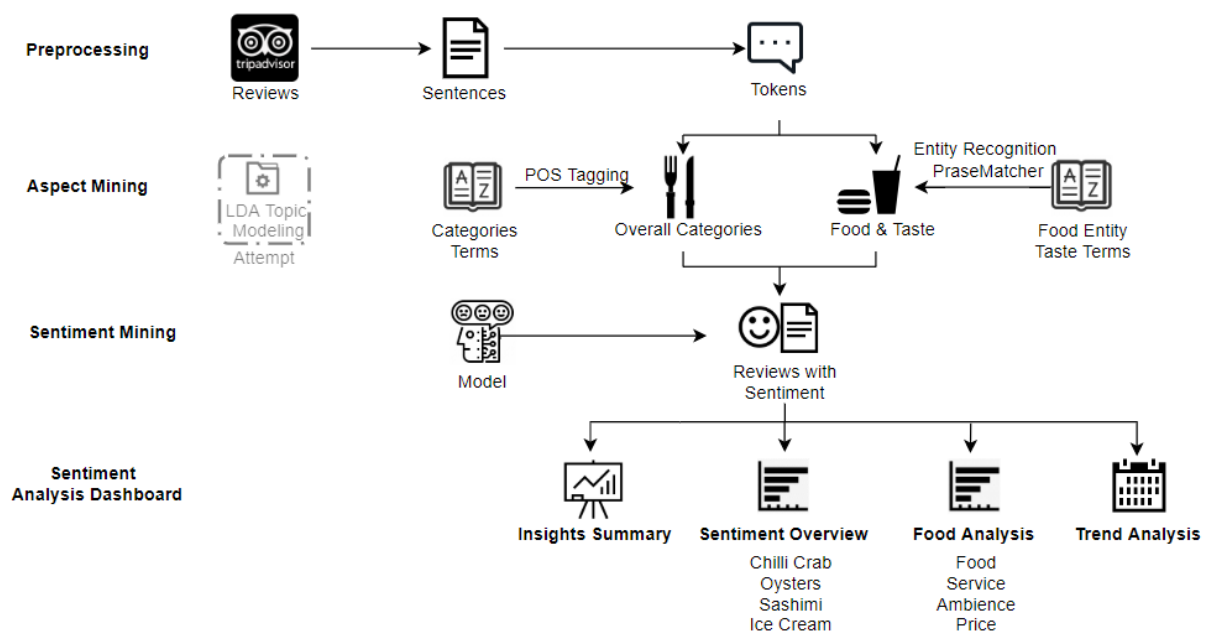


Figure 5: Sentiment and Aspect Mining Workflow

## 6.1 Pre-processing

The filtered reviews from the fake review filter are first converted into lower case, and SpaCy pre-processing pipeline is loaded into model, the pipeline includes Sentencizer and Tagger. Reviews are fitted into pipeline for batch process, reviews are first split into sentences and sentences are then tokenized into individual tokens. Tagger will perform POS tagging on each token for downstream aspect mining tasks.

## 6.2 Aspect Mining

Aspect mining is carried out at 2 levels. First to identify the 4 key aspects of a dining experience, namely: Food, Service, Ambience and Price that each sentence in a review is talking about. Second to identify and extract the taste aspect of different dishes in sentences talking about food.

For the first task, predefined restaurant terms are used for token matching. The terms include both explicit aspects such as nouns and implicit aspects such as adjectives and symbols. Each matched aspect is collected into the four corresponding categories: food, service, ambience and price. Some examples are list in figure 6 below.

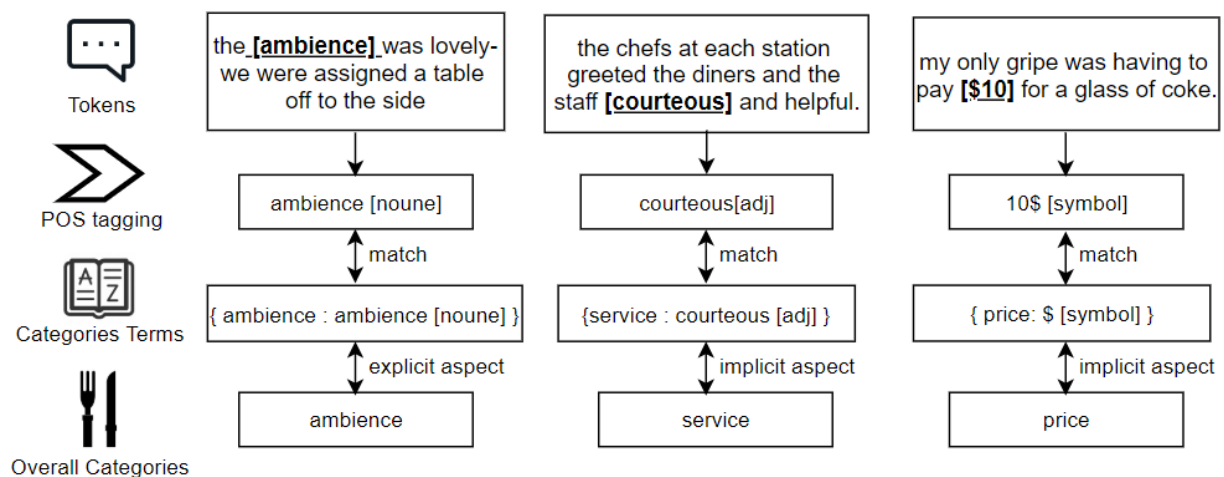


Figure 6: Aspect Mining Examples for Ambience, Service and Price

For the second task, food name entity obtained from the information extraction step (to be discussed later in section 8) is used to identify the food item in the sentences talking about food and pre-determined taste terms is used to pick up the taste aspect for the food item. The taste aspect is stored in the knowledge graph as a relationship between the food node and the restaurant node. Two examples of taste aspect mining are listed in figure 7.



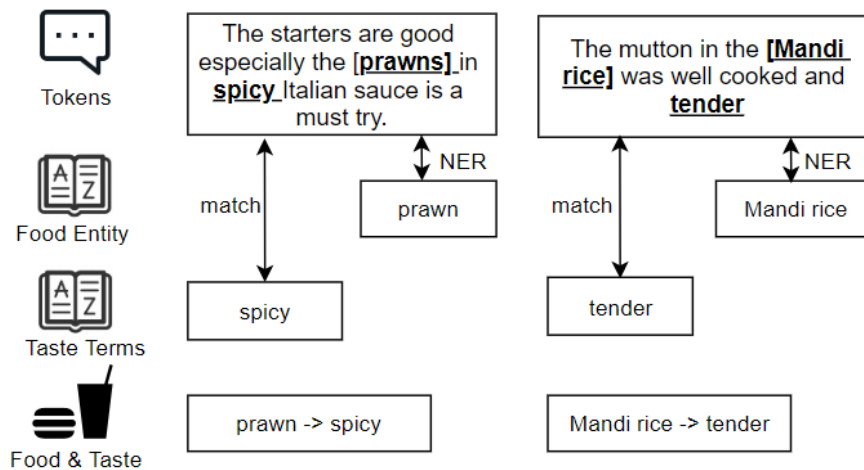


Figure 7: Aspect Mining Example for Taste of Food

An iterative approach is taken to build up and refine the categories terms and taste terms by performing the mining, examining the results and making adjustments (adding or removing) to the terms.

For the first task of identifying the 4 key aspects, LDA topic modelling has been explored but it did not produce good result.

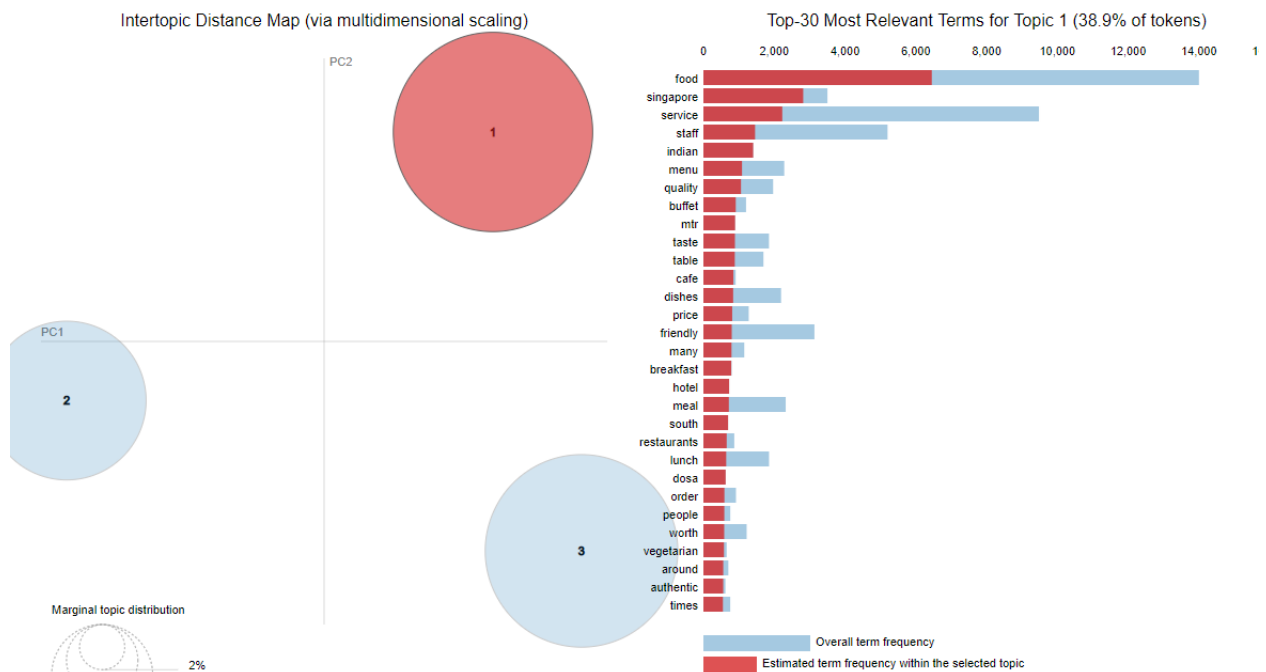


Figure 8: PyLDAvis Visualization for LDA Model Topic 1

**Performance** - From observation, topic 1 involves food, service-related reviews, it is hard to distinguish specific aspects of a restaurant. Terms such as 'quality', 'taste', 'friendly' and 'worth' seem to represent more subjective feelings from customers. In topic 2, meat-related food are frequently mentioned, accompanied by salad, wine, soup and bread. This topic is more likely to represent formal

meals. As for topic 3, it is similar to topic 1. Moreover, the terms such as 'birthday', 'ambience', 'special', and 'celebrate' might indicate more event-related information.

**Evaluation** - The LDA model doesn't cluster reviews as expected, the goal is to categorize different aspects of restaurants such as food, environment, service, etc. After investigating some of the reviews, three main reasons might cause this undesirable result.

i. Indistinguishable Content - Restaurant Reviews are naturally difficult to be summarized into categories even from human understanding. Customers are more likely to comment on a simple and general overview instead of a precise and detailed review. For example, review 'Amazing atmosphere, great food, kind and warm staff, wonderful view, well done Le Noir' is not easy to be categorized into a certain topic because it mentions a lot but with less detail.

ii. Imbalanced Categories - People are more likely to talk about food and service rather than ambience or price, these less frequently mentioned categories empirically require a larger number of clusters to be observed. However, the LDA Model is a soft-cluster so there is no objective metric to guarantee the result.

iii. Sparsity in Short Text - The conventional LDA model implicitly captures the document-level word co-occurrence patterns to reveal topics, and thus suffer from the severe data sparsity in short documents. The occurrences of words in short documents play less discriminative roles compared to lengthy documents where the model has enough word counts to know how words are related.

### 6.3 Sentiment Mining

Sentiment Mining is one of the core functions of the system, and the subsequent fine-grained analysis is performed based on the sentiment prediction result. A supervised transfer learning approach is used to train sentence level reviews and then predict each sentence as positive or negative polarity.

**Training Data** – The Yelp Public Dataset [2] does not provide sentiment labels directly. In order to find labels, a popular method is converting the rating of each review to sentiment polarity, the assumption is higher rating means more positive sentiment. However, for sentence-level task, one review with multiple sentences may have multiple sentiments towards different aspects. Sentence level labels are required. The final training dataset consists of two sets of data combined.

	From UCI ML Repository (labelled data)	From Sentiment Analysis API	Overall
Positive	500	14776	15276
Negative	500	5224	5724
Overall	1000	19000	20000

UCI ML Repository has a dataset created by Dimitrios Kotzias [3] that includes 1000 Yelp Reviews sentences. The selected sentences with clearly positive or negative connotations present good data quality. As for another part, after comparing multiple sentiment analysis API, Google Cloud Natural Language API [4] is chosen to predict 19000 selected sentences, the results are labelled with positive or negative polarity based on sentiment score.

**Pre-trained Weights** – The pre-trained weight 'en\_vectors\_web\_lg-2.1.0' [5] provided by SpaCy includes 1,070,971 keys with 300 dimensions unique vector each. The original source is trained by

GloVe [6] unsupervised learning algorithm, the training is performed on aggregated global word-word co-occurrence statistics from a corpus. The pretrained weights help the team initialize the model to achieve better accuracy and reduce huge computational requirements.

**Training Model (Tuning)** – First load the pre-trained weight into the SpaCy pipeline, then map the embedding between words and integers, then a 4 layers model is stacked using Keras sequential model.

**Embedding Layer** - Initialize embedding with the same shape as the pre-trained weight.

**Time Distributed Dense Layer** – Time Distributed Dense applies a same Dense (fully-connected) operation to every timestep of a 3D tensor.

**Bidirectional LSTM layer** – Bi-directional LSTM is an extension to typical LSTM that can enhance performance of the model on sequence classification problems. Bi-directional layer wrapper provides the implementation of Bi-directional LSTM in Keras. In this layer, hidden units are set to 64, and dropout is implemented.

**Dense layer** - The final layer is a dense layer with Sigmoid activation function, the output is a 1-dimension result.

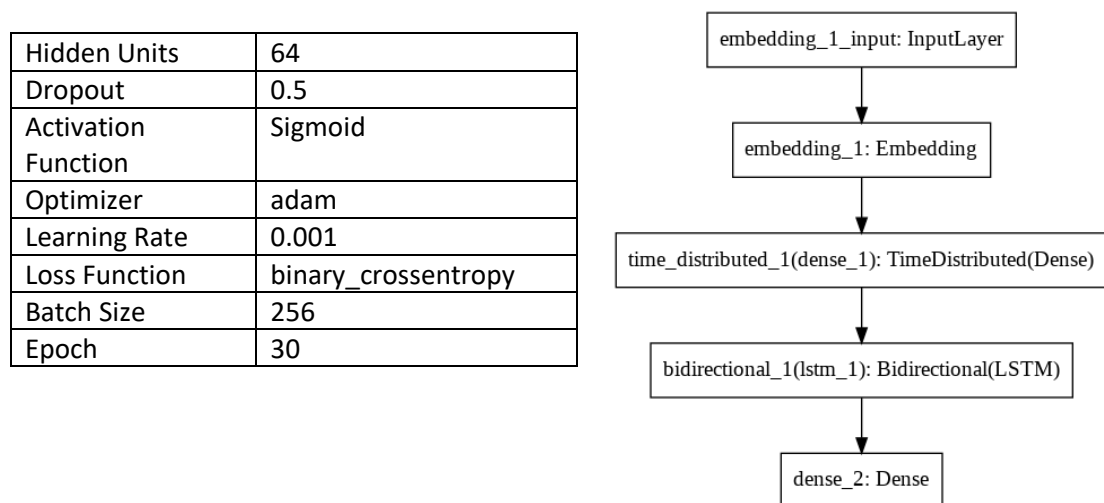


Figure 9: Model Structure & Parameters

**Evaluation** – The model is saved and deployed on SentimentAnalyser Class, which is then added into SpaCy Pipeline for batch processing. After 30 epochs of training, the training accuracy is around 0.87, and validation accuracy is around 0.85.

```

Epoch 29/30
104823/104823 [=====] - 205s 2ms/step - loss: 0.3036 - acc: 0.8665 - val_loss: 0.3468 - val_acc: 0.8538
Epoch 30/30
104823/104823 [=====] - 206s 2ms/step - loss: 0.3032 - acc: 0.8673 - val_loss: 0.3457 - val_acc: 0.8550
  
```

## 6.4 Further Exploration

Heuristic rules-based frequency approach is applied for the system due to its simplicity and efficiency, the candidate aspects are almost always the most important aspects of the restaurant. Due to time limitation, there are still several good techniques waiting to be explored.

**Unsupervised Approach** - Beside calculating direct frequency, Point-wise Mutual Information (PMI) can be used to compute the co-occurrence strength and thus indicate aspects of subjects. Context-Free Grammar Parsing and dependency parsing could be also helpful but might also fail if sentence is badly structured and grammatically incorrect, which is common in reviews.

**Supervised Approach** - SemEval-2014 [7] provides 3044 annotated sentences that are manually labelled into categories for Aspect Based Sentiment Analysis. Conditional Random Field (CRF) and LSTM could be an ideal training model.

## 7. Customer Sentiment Analysis Dashboard

**Insights Summary** – A rule-based insights summary is generated to provide eatery owners with a quick summary of actionable insights at the top of the dashboard before they dive in to look at the details. It covers the most representative information from all 4 categories.

**Sentiment Overview** - For each of the 4 categories, the number of positive and negative reviews are presented along with a score calculated based on polarity and samples of most representative positive and negative reviews. For price category, an additional metric called 'Valuation' is used to provide actionable insight for restaurant owners. The assumption is that if the price is frequently considered cheap with high positive sentiment, the food might be undervalued. Similarly, if price is frequently considered expensive with high negative sentiment, the food might be overvalued. By looking at these reviews, restaurant owners may adjust prices to gain more profit based on price elasticity of demand.

**Food Analysis** - For each food item, the report shows the number of positive and negative reviews, a score calculated based on polarity, and samples of most representative positive and negative reviews.

**Trend Analysis** - The trend analysis helps restaurant owners track and compare with previous performance. The initial thinking is to generate trend data on a monthly basis but after testing, we realized that the less frequently mentioned aspects such as ambience and price does not have sufficient data points to produce meaningful trends. Thus, the approach taken is to generate trend analysis only when the number of reviews crosses a certain threshold.

## 8. Information Extraction

In order to allow users to search for restaurant by food name and taste, obtaining the relevant information is the first step in building a database for the chatbot. For food name information extraction, namely the name entity extraction task has been accomplished by combing multiple methods. Figure 10 depicts the main techniques applied in this part. Since there is no well annotated corpus, rule-based method like lexical pattern matching and structural pattern recognition have been implemented first to extract food name. Using the output from the rule-based approach to build the annotated corpus, a Bi-directional LSTM-CRF model has been trained using the corpus to seek further extension of the information extraction task. The details of how this works will be elaborated below. For taste information extraction, aspect mining techniques has been applied as discussed in the sentiment and aspect mining section.

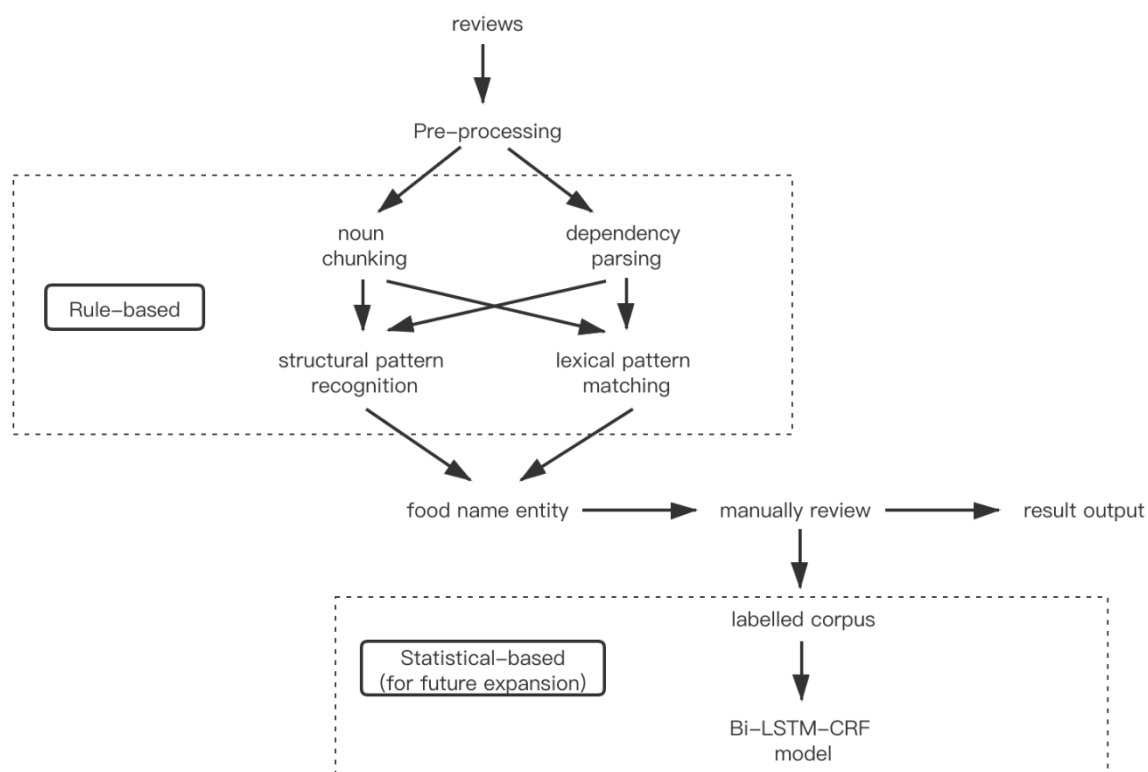


Figure 10. Information Extraction techniques

**Pre-processing** – After some data exploration tasks, the reviews from Tripadvisor have been cleaned through several processes, including deleting empty rows, removing duplicate rows, etc. Based on the final results to be given separately by restaurant, the clean reviews were grouped by restaurant name and saved into a list.

**Rule-based method** – The main technique used in this section is dependency parsing, besides basic natural language process tasks include sentence segmentation, tokenization, part-of-speech tagging, lemmatization and noun chunking have been implemented to help identify food name entities. In order to facilitate understanding, a typical example is described below in figure 11:

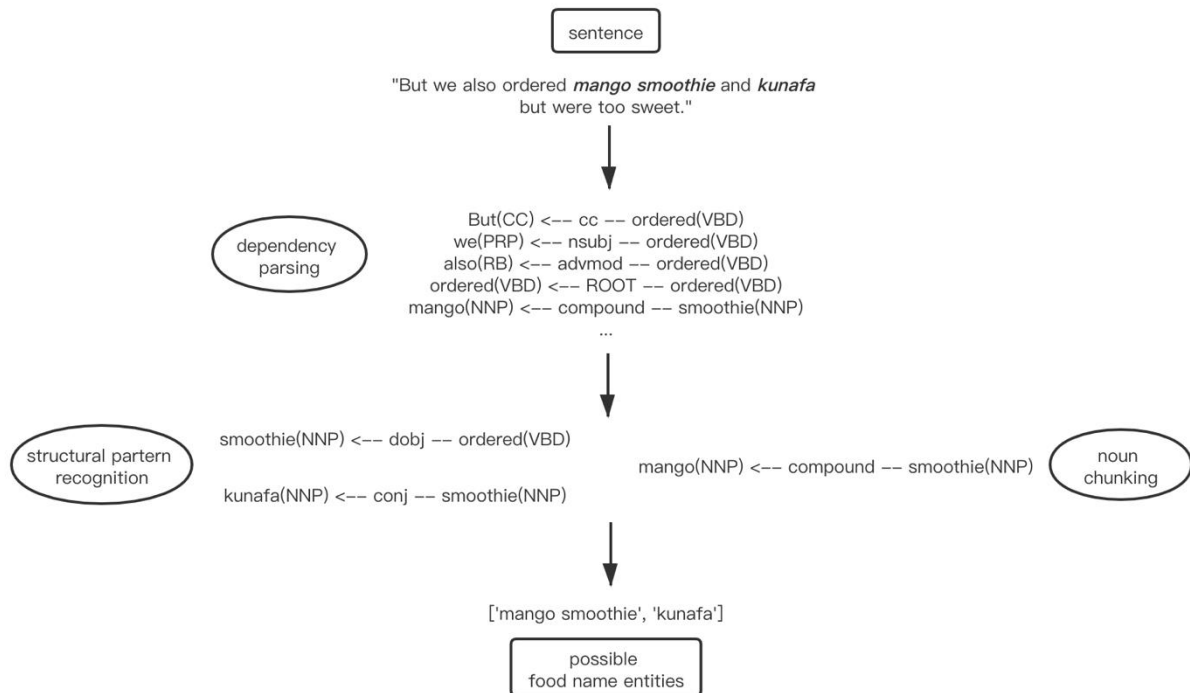


Figure 11. a typical example of main method

In addition to the techniques shown in this example, the following methods also have been implemented to identify the dishes name: (a) Filter all distinct words and phrases in the product category from spacy named entity recognition results, and then extract the noun and noun phrases in the data, get the possible food entities by ranking their frequency, (b) Sift out the entities in irrelevant category with the help of the spacy named entity recognition results (Figure 12 shows an example).

DATE  
['a sad week', 'last 2', 'Saturday', 'Tuesday', 'a long work day', 'one day', 'some days', 'weekly', 'last Wednesday', '1-2 days']  
EVENT  
['15 Oct.', 'the Rugby World Cup']  
TIME  
['4 hours', '8pm', 'all evening', 'a late afternoon', 'late afternoon', 'a few minutes later', 'the other night', 'an incredible evening', 'a great night', 'night', 'this afternoon', 'every night', 'a few minutes', 'afternoon', 'every minute', 'seconds', 'this evening', 'Monday evening', 'our first night', 'every hour', 'last minute', 'a memorable night', 'Year end night', 'evening', '5 minutes', '20 minutes']  
FAC  
['views!Awesome', 'Le Noir MBS', 'again!Nice', 'Le Noir Bar', 'Le Noir', 'Le Noir @ MBS', 'The Le Noir', 'Le NoirCame', 'the Le Noir Bar!', 'Clarke Quay']  
LOC  
['Le Noir MBS', 'the Singapore River', 'Marina bay', 'Marina Bay Sands', 'Marina', 'Le Noir', 'Marina Bay', 'bar!Great', 'marina bay', 'the marina bay', 'marina bay Le Noir Bar']

Figure 12. spacy NER result example

Since there is no labelled data for evaluation, the review of the outcome dishes name entities is done manually.

**Statistical-based method** -- Based on the label data obtained in the first way, a hybrid approach combining a bidirectional LSTM model and a CRF model [8] has been trained for possible future use when the data volume is significantly larger. Figure 13 depicts the structure of the model.

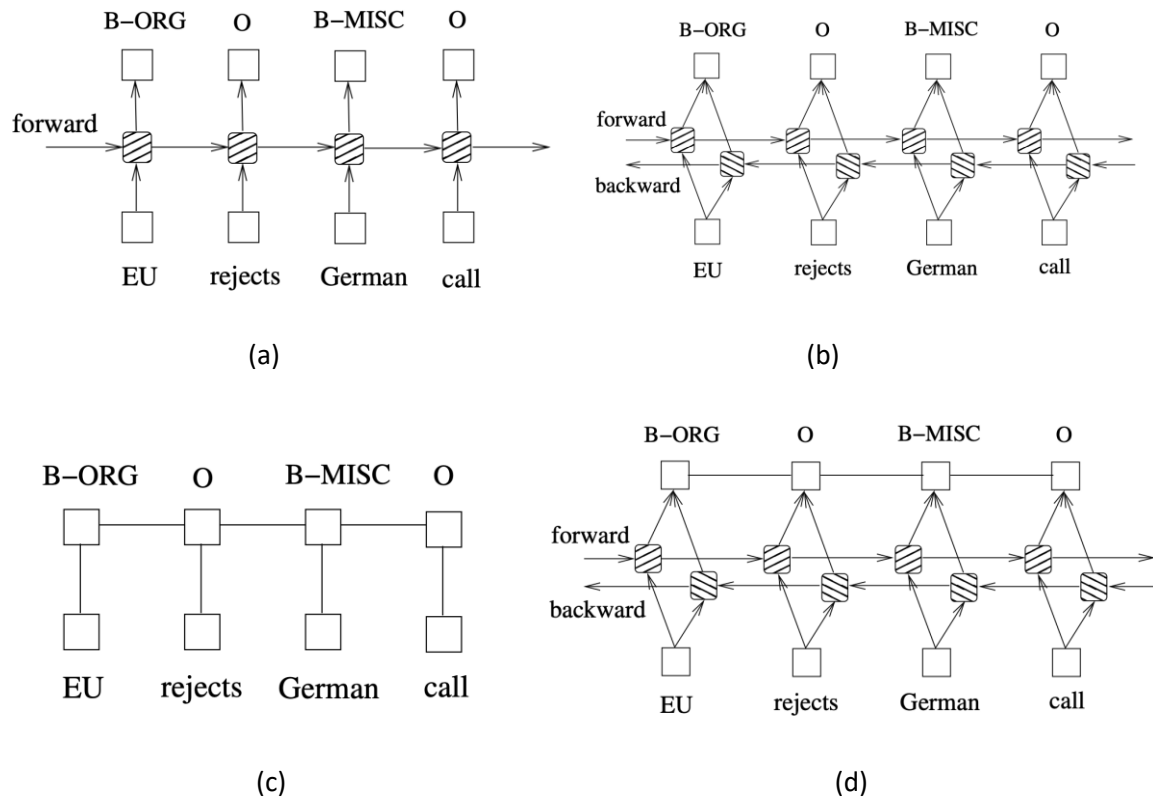


Figure 13. (a) LSTM network (b) Bidirectional LSTM network  
(c) CRF network (d) Bi-LSTM-CRF model

Source: <https://arxiv.org/pdf/1508.01991v1.pdf>

(a) Data pre-processing – The sentences which contain the dishes name entities have been filtered out as the corpus. The labels of those sentences have been created using a BIO annotation scheme. The reviews' sentences and labels have been mapped to a sequence of numbers. After that, in order to feed into the Bi-LSTM-CRF model, all texts have been processed to the same length.

(b) Model training -- The detailed model architecture is shown below.

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 50, 40)	387080
bidirectional_3 (Bidirection	(None, 50, 100)	36400
dropout_3 (Dropout)	(None, 50, 100)	0
bidirectional_4 (Bidirection	(None, 50, 100)	60400
dropout_4 (Dropout)	(None, 50, 100)	0
time_distributed_2 (TimeDist	(None, 50, 4)	404
crf_2 (CRF)	(None, 50, 4)	44
Total params: 484,328		

Trainable params: 484,328

Non-trainable params: 0

(c) Model evaluation – Due to the small amount of valid data, namely the reviews containing the dishes name, the accuracy of the model trained with this corpus is very low. However, when using a well annotated data set with a large amount of data to train the model, the accuracy is higher than 90%, which proves that the model can achieve good performance and can be built in advance to prepare for possible future use when there is more data.

## 9. Knowledge Graph

Considering the user preference learning feature and the complex query for highly connected data requirements of the chatbot, in addition to the basic CRUD functions, the database for chatbots should have good performance in creating and querying relationships. Compared with traditional relational databases, graph databases are particularly efficient for deep and complex query performance. The table below shows the deep query performance comparison of MySQL and Neo4j:

Query depth	MySQL execution time (s)	Neo4j execution time (s)	Amount of data returned
2	0.016	0.01	~2500
3	30.267	0.168	~110000
4	1534.505	1.359	~600000

(<https://www.manning.com/books/neo4j-in-action>)

Hence, a knowledge graph has been designed using Neo4j to facilitate fast query response for the chatbot as part of good user experience. The information extracted together with data crawled from Tripadvisor and obtained from the user interface have all been cleaned and saved into the graph database based on the basic node and relationship structure. The details about the knowledge graph will be elaborated below.

**Graph database schema** -- Figure 14 and the table below depicts the schema of the knowledge graph.

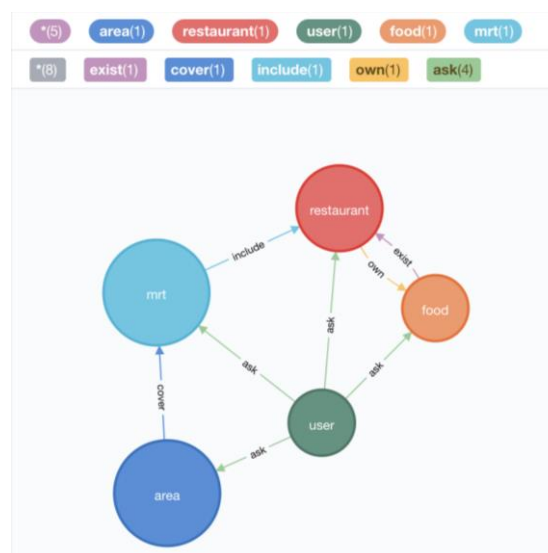


Figure 14. Knowledge graph schema

(circle: Node, line: Relationship)



Schema	Label	Description
Node	area	The planning areas of Singapore
	mrt	Singapore MRT stations
	restaurant	Restaurants in Singapore (Tripadvisor)
	food	Dishes extracted from the restaurants' reviews
	user	Service user
Relationship	cover	area -> mrt
	include	mrt -> restaurant
	own	restaurant -> food
	exist	food -> restaurant
	ask	user -> area
		user -> mrt
		user -> restaurant
		user -> food

**Graph database data** -- Graph database stores data through nodes, nodes properties, relationships and relationships properties, the details about the data stored in graph will be divided into nodes relevant and relationships relevant and elaborated below:

#### 1. Nodes relevant

Node Label	Node Properties	Description	Data Example
area	name	Uniquely identifies the area	{ "name": "ang mo kio", "direction": "North-East" }
	direction	At which part of Singapore	
mrt	name	Uniquely identifies the MRT/LRT station	{ "name": "bishan mrt station", "stn_no": "CC15" "line": "Circle Line" "coordinate": "1.35130868,103.84915", }
	stn_no	Station number	
	line	In which line	
	coordinate	North latitude, East longitude	
restaurant	name	Uniquely identifies the restaurant	{ "coordinate": "1.301108,103.859909", "address": "66 Bussorah Street, Singapore 199479 Singapore", "res_id": 8, "rating": 4.5, "name": " positano risto", "link": "https://www.tripadvisor.c om.sg/Restaurant_Review-g294265- d13152787-Reviews-Positano_Risto- Singapore.html", "cuisine": "italian, pizza", "description": "Food and ambiencc italian, Pizza, European, Vegetarian Friendly, Vegan Options,
	res_id	Uniquely identifies the restaurant	
	link	Link to Tripadvisor detail page	
	cuisine	style or method of cooking	
	rating	Restaurant rating in Tripadvisor	
	address	Restaurant address	
	from	In which area, and distance to some of the landmarks	
	tel	Restaurant telephone number	
	coordinate	North latitude, East longitude	

	description	Restaurant descriptions in Tripadvisor	HalalAwesome LasagnaThe seafood (mussels, prawns, calamari and scallops) were definitely fresh an...Special Occasion Dining, Romantic, Groups, Business meetingsBirthday treat from sis and broinlawalso a great place to celebrate a birthday or special occasion.", "from": "Central Area/City Area,1.1 km from Little India", "tel": "+65 6292 1866" }
food	name	Uniquely identifies the dish	{  "name": "tom yam kung", "cuisine": "asian, thai" }
	cuisine	style or method of cooking	
user	id	Uniquely identifies the user	{  "name": 123456000 }

## 2. Relationship relevant

Relationship start node	Relationship Label	Relationship end node	Relationship Properties	Description
user	ask	area	count	User mentioned this area when consulting our chatbot.  count: number of times users asked
user	ask	mrt	count	User mentioned this MRT station when consulting our chatbot.  count: number of times users asked
user	ask	restaurant	count	User mentioned this restaurant when consulting our chatbot.  count: number of times users asked
user	ask	food	count	User mentioned this food when consulting our chatbot.  count: number of times users asked
area	cover	mrt		The planning area covers the MRT station
mrt	include	restaurant	distance	The restaurant is closest to the MRT station.  distance: distance from restaurant to MRT station
restaurant	own	food		The restaurant has the food
food	exist	restaurant	taste	The food exists at the restaurant

				taste: the taste of the food in this restaurant
--	--	--	--	---

**Graph database data source** – The main natural language processing processes in this part include: (a) the food relevant information, which have been extracted from the reviews crawled from Tripadvisor, mentioned in the last section. (b)The ‘taste’ property of the ‘exist’ relationship have been extracted from the sentiment mining section.

In addition, the ‘distance’ property of the ‘include’ relationship is calculated by using the coordinate of the MRT station and the restaurant with the help of one python library called geopy.

The data source of the planning area and the MRT station in Singapore are both from Wikipedia. ([https://en.wikipedia.org/wiki/Planning\\_Areas\\_of\\_Singapore](https://en.wikipedia.org/wiki/Planning_Areas_of_Singapore) , [https://en.wikipedia.org/wiki/List\\_of\\_Singapore\\_MRT\\_stations](https://en.wikipedia.org/wiki/List_of_Singapore_MRT_stations) )

Except for the data mentioned above, all other data is crawled from the Tripadvisor.

**The amount of data in the knowledge graph** – The table below shows the amount of data in the graph database.

Schema	Label	Count
Node	area	56
	mrt	188
	restaurant	3028
	food	227
	user	30
Relationship	cover	141
	include	3028
	own	365
	exist	365
	ask	11

**Partial knowledge graph example** – Figure 15 shows part of the knowledge graph.

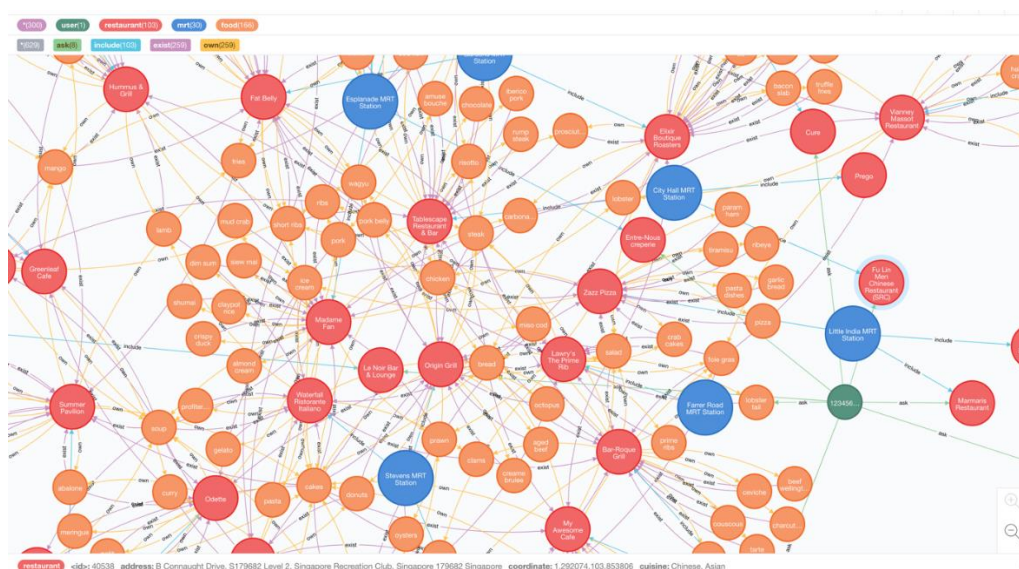


Figure 15. partial knowledge graph example

**Functional interface of graph database** – In order to allow chatbot and web application to easily access data in graph, several interfaces have been created, listed in table xxx.

Function name	Input	Output	Description	For
<b>create_user</b>	user id	NA	create user node	Web application
<b>query_res</b>	query dictionary	restaurant list (each item in the list is a dictionary, containing all information of one restaurant, i.e. restaurant node's properties)	search restaurants based on query and return all the information of the restaurants matched	chatbot
<b>query_user</b>	user id	1.reference list (each item in the list is a dictionary, containing all information of all the area, mrt, restaurant, food the user asked before, i.e. those nodes' properties) 2.count list (Corresponding number of queries) 3.type list (Corresponding type of the reference)	According to the user id, get all the historical preference data of the user to make private recommendations to the user.	chatbot
<b>user_rel</b>	id, reference fictionary	NA	Update the user's preference information. If the 'ask' relationship is already existing, update the relationship's 'count' property; else, create new 'ask' reference.	chatbot
<b>filter_res</b>	two restaurant list	NA	Get restaurants information that exists in both restaurant lists.	chatbot

## 10. Fake Review Filtering

### 10.1 Overview

Fake review detection of our system is the first components after data crawling. The aim of this component is to filter as many as possible fake reviews and push those non-fake reviews into the database for other components like sentiment analysis dashboard to use, which is very important and essential because our system, especially generating reports for eatery owners, needs real reviews data written by real customers who actually been to the eatery to generate truly effective reports, otherwise, our system will be distracted by those review generated by machine or written by anonymous online workers (called Turkers) leading to waste computing resources and generate non-reliable recommendations and analysis results. Our teams uses several methods of feature engineering on the dataset including word2vector, behavioural features extraction, imbalance data processing and we also trained three types of models, including supervised learning, semi-supervised

learning and deep learning, to compare the performance and chose the best one. All these models are trained on the same dataset, labelled restaurant reviews of Yelp Dataset. The final model we use in the system gives 0.67 F1 score and 93% Recall on the test dataset.

## 10.2 Problems and Solution

The dataset or the data source of the sentiment analysis dashboard is the review data from restaurant recommendation platforms like TripAdvisor and Yelp and those data are now widely used by individuals and organizations for their decision making. However, due to the reason of fame and influence, some people try to cheat the system by add many spam review or fake review to promote or to demote some restaurant. In recent years, fake review detection has attracted significant attention from both the business and research communities. The main idea is to use machine learning or deep learning method to differentiate the fake review and the non-fake review, which is a classification problem. Although the topic has attracted significant attention and many researchers or teams are doing that topic, it is still a very challenging task because there are two major problems in training of fake review classification model.

**Limited or no labelled dataset** – Obtaining labelled fake reviews for training is difficult because it is very hard if not impossible to reliably label fake reviews manually. Even if someone did manually labelled review data, it would be expensive and too small to train a robust model. Some method is to use other review generation models or platforms like Amazon Mechanical Turk (AMT) to generate reviews and labelled those reviews as fake review. But those data are widely different from the real review data, which appears on the Yelp and TripAdvisor and it will be very easy to use some model to differentiate those data, but those models are unreliable or useful for our system or any other real-world business application. Fortunately, we found some papers on fake review detection from some excellent researchers, and we also got the Yelp Review Dataset which has the label with fake and non-fake. It helped us a lot because the dataset is collected from Yelp and labelled based on the Yelp Fake Review strategy, which is to display different types (fake and non-fake) of reviews on different webpage. Furthermore, Since the Yelp Dataset and the TripAdvisor data which we need to conduct fake review detection are in the same restaurant domain, we don't need to use transfer learning and we can directly use those labelled Yelp data to train the model.

**Imbalanced Dataset** – For review of restaurant domain, regardless of date collection from which platform. The common problem is that the data is imbalanced, because usually the number of non-fake review is many times more than fake review number, so we have to handle the imbalanced data problem during the data pre-processing. Though the imbalanced data handling method will hurt the model accuracy, the aim of this components is to let the model differentiate the fake review and we need to focus on the F1 Score of the model and let it find as many as possible fake reviews from the data as possible.

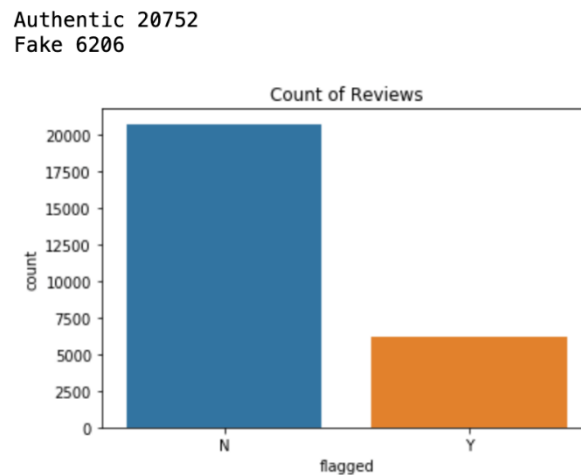
## 10.3 Data Pre-processing and Feature Engineering

### Data Overview

The dataset we use for training the fake review detection models is almost the same as the ones we use for the sentiment analysis dashboard which is the Yelp Dataset. The only difference between those two datasets is that for fake review detection we have the Yelp Dataset with fake or non-fake labels.

The Yelp dataset contains two types of reviews, hotels and restaurants. Both types of review have reviews, business owners and reviewers data profile.

For the fake or non-fake review label, there are 4 types of labels, Y, N, YR, NR. Reviews with Y/N: Reviews obtained from the restaurant page wherein all Y reviews from the filtered section and N reviews from the regular page. Reviews with YR/NR: Reviews obtained from the reviewer profile page. In our model training we only use the Y and N. [9]



## Feature Engineering

For any machine learning or even deep learning model training, feature engineering is important and even more important than fine-tuning at the end of model training, especially in the Nature Language Processing field. However, in the field of fake review detection research, the most difficult part is not the feature engineering of review itself, Word2vec technology is usually used for word and sentence representation, like unigram, bigram or skipgram. There are many methods but the result is not very good. Therefore, based on the papers we have referred to, we decided to conduct a more in-depth study on Yelp Dataset.

**word-2-vec** – For the word-2-vec processing, we choose two ways to process our review data, one is use the word2vec model trained on our dataset and then use that model to vectorize the review content, the other method is to use the Glove Twitter 100 Dimension pre-trained word2vec model to vectorize the review content. We did not do anything like POS tagger or Bigram

**Behavioural feature** – For the behavioural feature extraction, we referred to the Liu Bing’s paper and extracted and used the following five behavioural features, Maximum Number of Reviews, Review Count Percentage of Positive Reviews, Review Length, Review deviation and Maximum content similarity [9]

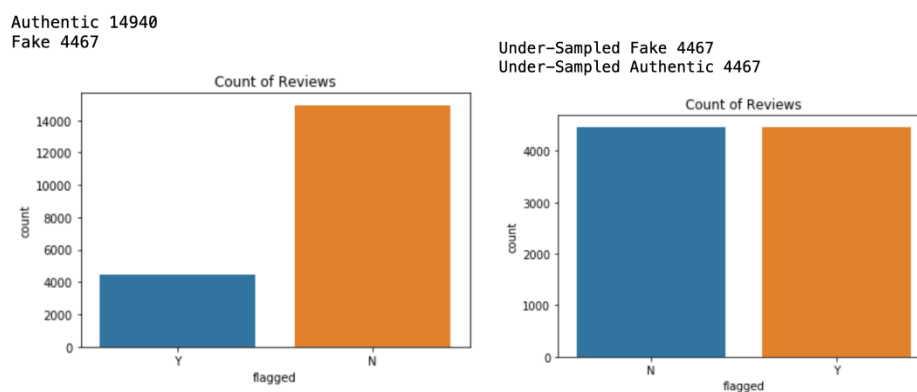
**Review Count Percentage of Positive Reviews** - Opinion spamming can be used for both promotion and demotion of target businesses. This feature is the percentage of positive (4 or 5 star) reviews. Non-spammers on the other hand show a rather evenly distributed trend where a varied range of reviewers who have different percentage of 4-5 star reviews. This is reasonable because in real-life, people (genuine reviewers) may have different levels of rating nature.

**Review deviation** - This feature analyses the amount that spammers deviate from the general rating consensus.

**Maximum content similarity** - Crafting a new fake review every time is time consuming. To examine whether some posted reviews are similar to previous reviews, we compute the maximum content similarity (using cosine similarity) between any two reviews of a reviewer. Content similarity is another metric where opinion spamming is exhibited quantitatively.

Due to the different data structure between Yelp dataset and Tripadvisor dataset, we cannot extract or use more behavioural features to train on the Yelp dataset. We will extend the Tripadvisor dataset in the future.

**Imbalance data handling** – For the imbalance data handling, we use the under-sampling method, which is to sample the same number of non-fake review data as the fake review data in the dataset. This method is only performed on the training dataset and the validation and test dataset retain the original data proportion.



## 10.4 Model Training and Testing Result

For the model training, we experimented with three types of models, supervised learning, semi-supervised learning and deep learning. Supervised learning, semi-supervised learning we both use the same word2vec pretrained model and behavioural features because we want to compare the semi-supervised learning and supervised learning. For the deep learning model, we use the Bi-LSTM with pre-trained embedding model which is the same as the machine learning model.

Semi-supervised learning is an approach to machine learning that combines a small amount of labelled data with a large amount of unlabelled data during training. Semi-supervised learning falls between unsupervised learning (with no labelled training data) and supervised learning (with only labelled training data). In our semi-supervised learning training process, we use the validation set and the unlabelled data in the semi-supervised learning.

Models	Accuracy	Precision	Recall	F1
Random-Forest pretrained-word2vec				
No-Undersampling with BF	0.85	0.73	0.58	0.65
Word2Vec with BF	0.82	0.57	0.91	0.70
Word2Vec	0.78	0.52	0.88	0.65
BF	0.82	0.57	0.90	0.70

Models	Accuracy	Precision	Recall	F1
Semi-Random-Forest pretrained-word2vec				
No-Undersampling with BF	0.85	0.74	0.58	0.65
Word2Vec with BF	0.79	0.52	0.93	0.67
Word2Vec	0.73	0.46	0.93	0.61
BF	0.79	0.52	0.94	0.67

Models	Accuracy	Loss	Epoch
Bi-RNN(GRU)			
Word2Vec with BF	0.58	0.79	10
Word2Vec with BF	0.62	0.65	20

## 10.5 Conclusion

According to the table and team discussion, we choose the semi-supervised random-forest model to be integrated into our system. We have found that in the fake review detection, machine learning performance is better than deep learning. More training time or a larger training corpus may lead to better results. However, under the observation of our training results, deep learning does not have better advantages over machine learning. Meanwhile, due to the large environmental requirements of deep learning, we cannot integrate the model into our system.

In fake review detection model building and training process, our teams learned a lot of knowledge about word2vec and how to migrate the pre-trained word2vec model in the Pytorch model, how to create more features and concatenate them with the word embedding output. We realized that in this domain, only using the word2vec or word embedding methods cannot get a good performance, because the feature in the review content is limited by the review itself, like length and style. But using the Behaviour Features is different, it is to differentiate the fake reviewer by their behaviour, like the number of written reviews and the similarity between each of their review. That is another way to perform feature engineering by adding the reviewers feature into the review data, which let us realize the impact of data pre-processing is far greater than for the choice of the model.

## 10.6 Further Exploration

**Behaviour feature** - Adding behaviour features into the data made a huge improvement on our model performance. Without doing further feature extraction in the word2vec process, the model can achieve 86% accuracy (91% if we use all features in Yelp Dataset), it also let us know in the limited time and training resources, further extracting feature from the data is much better than using a more sophisticated model on improve the model's performance. However, due to the difference between the Yelp dataset and Tripadvisor dataset, we were unable to use more features in the Yelp data, like reviewer's fan count, friend count. Therefore, we needed to further optimize Tripadvisor crawler script to obtain more reviewers' information, so as to further improve the model performance.

**Word Embedding and further engineering** - In the process of modelling this component, our team did not further expand word2vec and word embedding, such as Bigram or N-gram, but used the pre-trained Glove word2vec model. The reason is we want to compare the effects of using self-built and pre-trained models on classification problems and we did not have enough time and resources to train a word2vec model on a large corpus from scratch. Furthermore, we want to compare the gap between



machine learning and deep learning using the same word2vec model, but for this reason, our deep learning model actually does not have enough depth for extracting enough feature. Therefore, we will make adjustments for word embedding from two dimensions. One is using POS tagger and Bigram to do further extraction on the basis of word2vec for improving the performance of machine learning model. The other is to add more depth and training time for the deep learning model.

## 11. Technical Implementation Details

### 11.1 System Architecture Overview

The figures below depict the high-level system architecture and the operation activities for consumer and eatery owner processes.

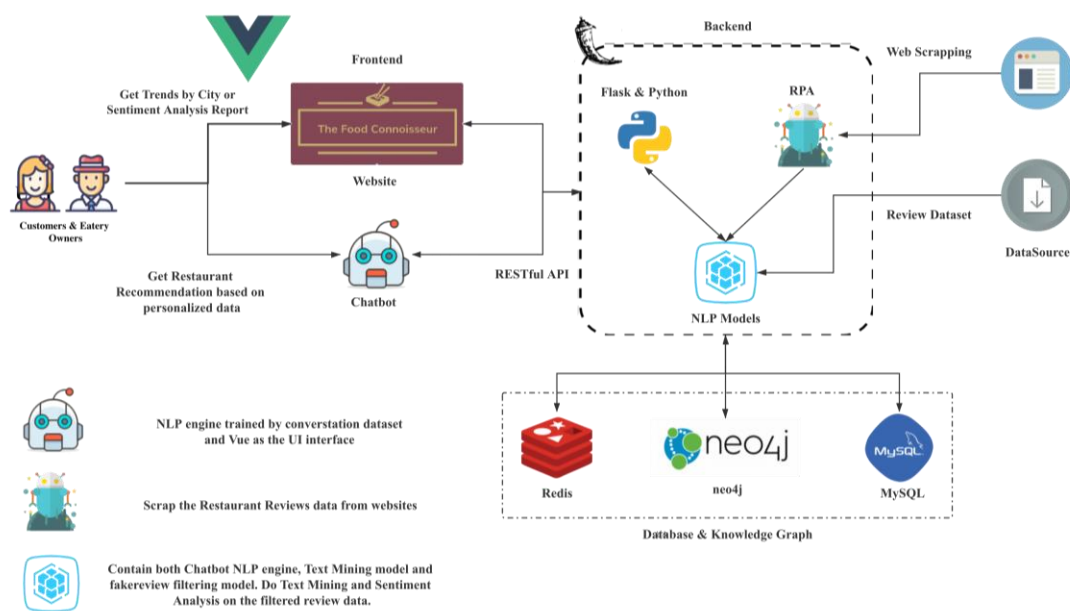


Figure 16: System Architecture Diagram

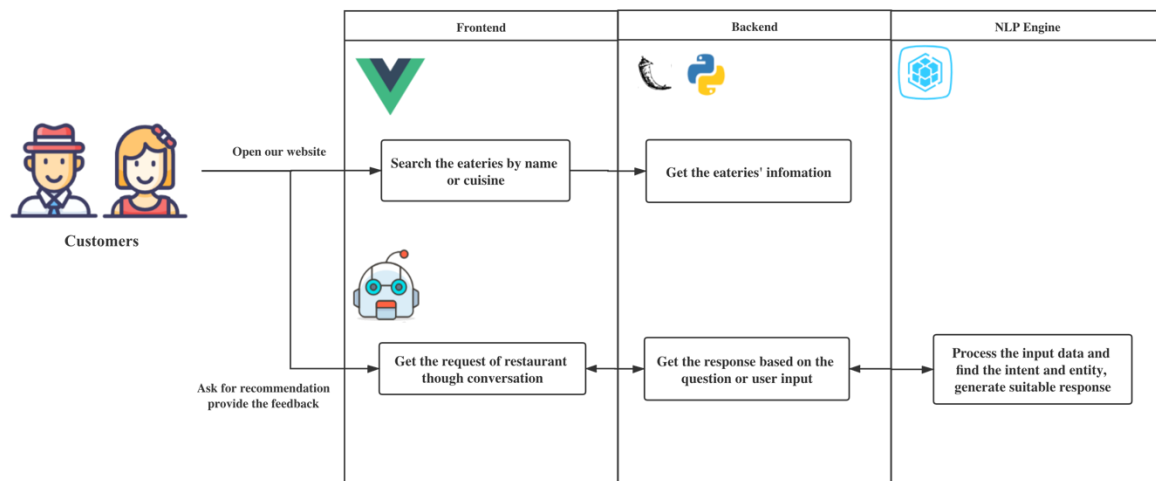


Figure 17: Consumer Operation Activity Diagram

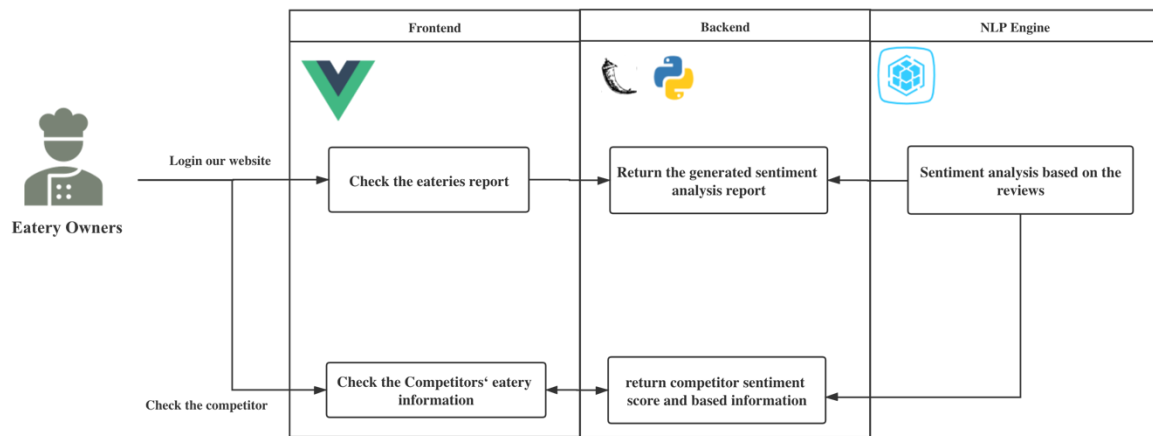


Figure 18: Eatery Owner Operation Activity Diagram

The system is a hybrid solution made up of multiple frameworks and databases including Vue.js, Flask, Neo4j, Redis and MySQL. It contains a web application and multiple NLP modules discussed in previous sections. The frontend and backend adopt a loosely coupled architecture design which makes it modular with high scalability and maintainability. The frontend is a single-page application developed using Vue.js and Element.js, and the backend is a web service that provides RESTful API calls developed using Flask. For integration with other parts of system, the backend needs to have all the dependencies and components connected to the models and databases. The components to connect includes Redis, Neo4j, MySQL, models for Fake review filtering, model for conversational UI, model for information extraction and model for sentiment and aspect mining.

## 11.2 Web Application

The web application is a complicated component in the system. Its backend needs to invoke interfaces from other components and to provide interfaces for other components to access system's main database. In addition, its frontend needs to be designed not only for user requirements, but also for interface and webpage design according to integration requirements of chatbot and sentiment dashboard. Based on those requirements, the team has designed and implemented a functional and efficient web application with high interoperability. The design and implementation of the web application can be split into frontend and backend.

- Design and implement frontend using Vue.js and Node.js
  - Common components for Chatbot and Sentiment Dashboard frontend, such as user registration and user login.
  - Integrate and display components for Chatbot such as feedback and recommendation.
  - Integrate and display components for Sentiment Dashboard
- Design and implement backend using Flask
  - Connection components for Redis, Neo4j and MySQL
  - Invoke interface for Chatbot and Sentiment Dashboard.

Due to the frontend and backend separation architecture, the system's frontend and backend have their own development and deployment environment. The frontend development environment is Node.js and deployment environment is Nginx. The backend development and deployment environment both uses Python 3.6. For security and user permissions, the system carries out

authorization and restriction of permissions based on pages and API requests respectively for the frontend and backend, ensuring the security of the system.

### Front End Design and Implementation

For the design and implementation of the frontend interface, the popular progressive JavaScript framework, Vue.js, and the Lin-cms-vue frontend solution is selected. The Lin-cms-vue frontend solution is selected for UI design which provides many useful components and tools. The UI is mainly designed using Element.js, which is a Vue based component library. For the Integration and display components of the Chatbot, Vue dynamic binding feature is used to import the chatbot as a component into the frontend and access the backend using the RESTful API interface. For the Integration and display components of the Sentiment Dashboard, Element.js and Echart.js is used to design the UI interface. Element.js is used to build the layout of the Sentiment Dashboard and Echarts.js is used to design the line chart, radar chart and bar chart. In order to make it more intuitive for eatery owners and consumers to query information and location of eateries, the team has designed and customized map charts and markers by using AMap's Web JS API.

Due to the Vue.js framework, the frontend architecture follows the MVVM Architecture which allows separation of user interface logic and business logic. Each of the webpage is composed of different components, which make the frontend system can be very scalable

In terms of security, the system uses page security authentication based on user's action scope and present different pages through different action scope of each user. This is to ensure that there will be no unauthorized operation. At the same time, the system also uses Token as the authentication method to interact with the backend to ensure that the backend RESTful API will not be called without permission.

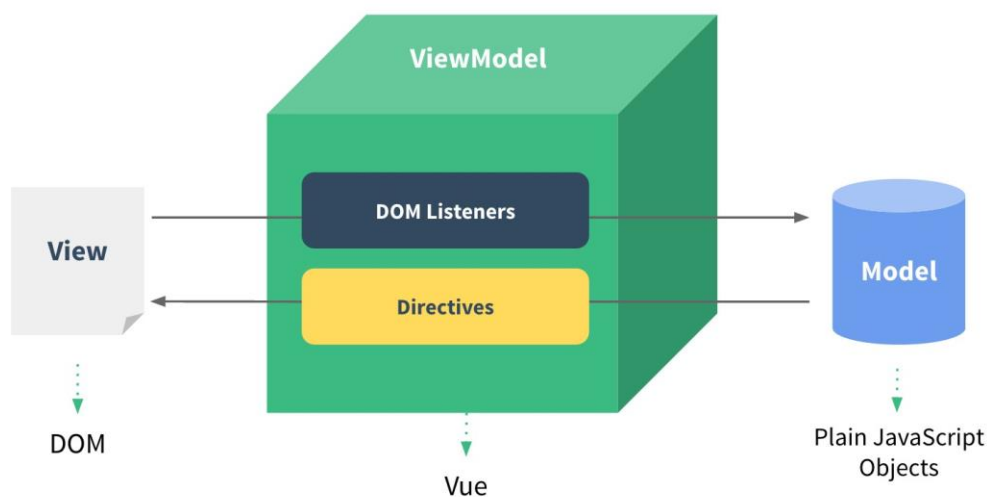


Figure 19. MVVM Framework

### Back End Design and Implementation

The system's backend uses arguably the most popular web application python library, flask, to build a RESTful API web service, which is very suitable for backend web applications that are separated from the frontend. For backend development, a combination of Blueprint and Redprint (modified Blueprint) is used to build the RESTful API and the security authentication based on user permissions to ensure the security of RESTful API calls.

The backend system is mainly responsible for providing the RESTful API service for the frontend to get data through calling various models and databases. Common data for Chatbot and Sentiment Dashboard is stored in MySQL database and accessed via SQLAlchemy ORM framework for easy storage and retrieval. ViewModel is used to modify the selected data from the database before displaying on the frontend. Multithreading and HTTP request is used to interact with the backend layer, making sure that the database operations will not affect the user experience in the frontend. A customized Redis Control Component is implemented to query and update the conversation data from the chatbot.

The Sentiment Dashboard also taps on the SQLAlchemy ORM framework to connect to the MySQL database to store the values for each eatery and to provide RESTful API interface for the frontend components to retrieve these values.

### 11.3 Chatbot Architecture

The conversational UI also follows the frontend and backend separation architecture. The frontend of the conversational UI is an independent web application developed using Vue.js, which makes it easy to be integrated with the web application frontend. Chatbot backend has three key components namely, conversation store, conversation protocol and chatbot logic.

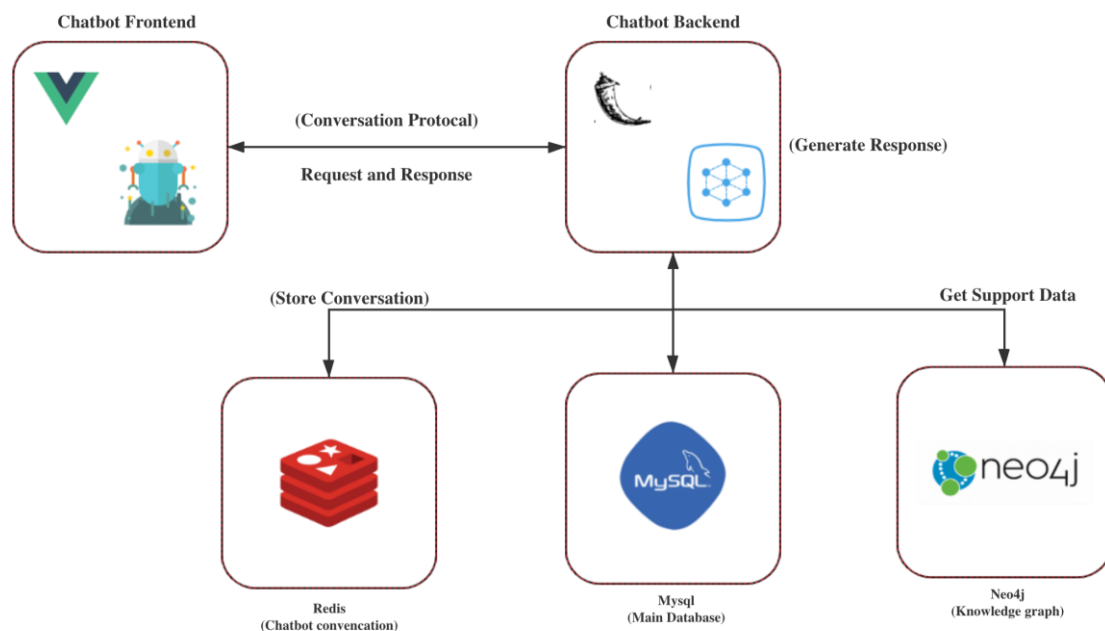


Figure 20. Chatbot Architecture

**Frontend user interface** – The user interface is mainly developed and implemented using Vue.js and the team has conducted in-depth secondary development by referring to dialogflow-web-v2 on GitHub (<https://github.com/mishushakov/dialogflow-web-v2>), which uses Vue.js to implement the UI and frontend functions of Dialogflow. The team spent a lot of effort modifying and customizing it to avoid having to use the paid features.

Our implementation includes:

1. Modify the UI interface to meet the overall style and functional needs of our web application
2. Modify the underlying logic and interaction protocol. Although the github library implemented the majority of dialogflow functions, we need to modify the logic and

interaction protocol because it encapsulates the interface to access dialogflow which involves a backend paid service.

3. Research on the conversation protocols of dialogflow and Action on Google to understand and modify it to adapt to our system requirements.
4. Modify and write the interface code to access the backend and replace the Dialogflow interface to connect our own chatbot backend service.

The end result is a Chatbot user interface which is the only open source development on github based on Vue.js using dialogflow UI style and interaction protocol, but not required to connect to any paid Chatbot platform. Axios library is used to interact with the backend, which is very convenient for linking to various Chatbot backend. The team will make this open source to benefit future learners and the larger NLP community.

**Conversation store** – This component is responsible for storing the user conversation data along with some related system data. Conversation data is used for Dialog State Tracking and to store Entity Data. The team has designed a custom conversation data structure and uses the high-performance NoSQL database Redis to store that data, which can handle a large number of data requests.

**Conversation protocol** – The conversation protocol is for frontend and backend interaction. Because we use the customized dialogflow conversation protocol, we also use the Flask-assistant library which already has the Majority of Actions on Google components. We do not need to generate the JSON files following the Actions on Google Protocol by ourselves. But unfortunately, the library doesn't provide or create all the components, so our team decide to improve and modify it by adding some new function and class into it. We have read the source code and understand how this library generates and assembles the response and make them follow the Action on Google Protocol. After that, we added and modified the original code to fill in missing components. Though this library was not written by our team, we are very familiar with its logic and structure after the implementation.

**Chatbot logic** – Discussed in Section 5.

## 11.4 Database Design

The database design considers user and system integration requirements.

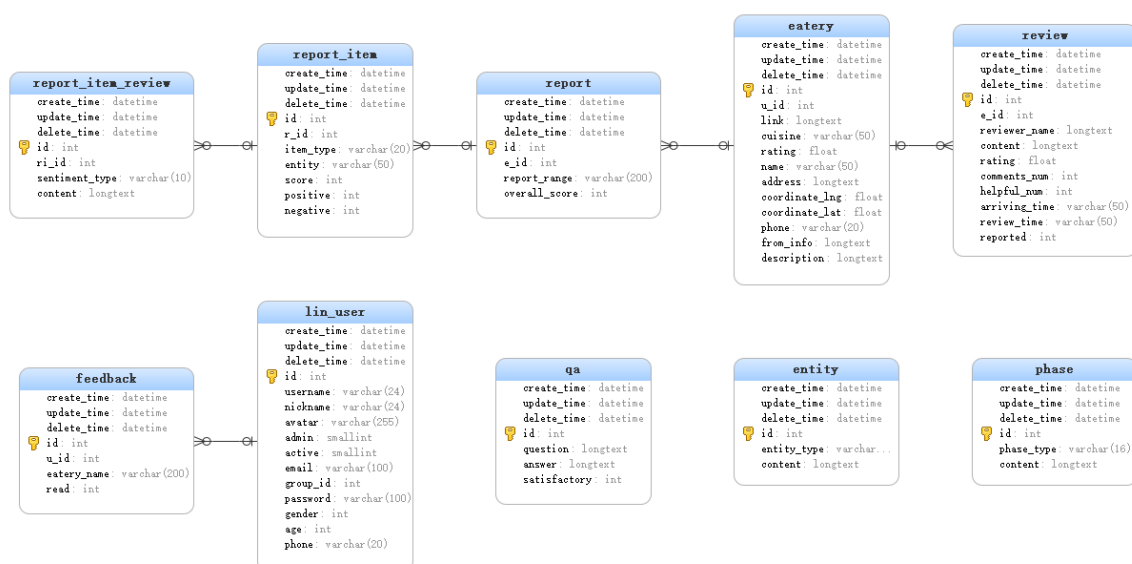


Figure 21. Database ER

As shown in the figure above, ten tables has been designed. These tables can be divided into three categories according to different functions namely: common functions of web applications data, conversational UI data and sentiment dashboard data.

- Common functions of web applications data
  1. Lin\_user is the table to store the user profile including id, username, nickname, age, password and so on, which uses modified lin-cms framework user table
- Conversational UI data
  1. Entity is the table to store the entity data including the entity\_type and content which is used in intent detection and entity extraction of the Conversation UI Backend.
  2. Phase is the table to store the phase data including the phase\_type and content which is used in intent detection of the Conversation UI Backend.
  3. Feedback is the table to store the feedback checking which is used in the Conversation UI Backend for checking whether to send a notification to the user requesting a feedback or not.
- Sentiment dashboard data
  1. Eatery is the table to store the eatery data from the web scrapping including the eatery name, cuisine, address, rating and so on.
  2. Review is the table to store the review data from the web scrapping filtered by fake review detection model including the eatery name, cuisine, address, rating and so on.
  3. Report, report\_item, report\_item\_review are three tables to store the sentiment analysis results.

## 11.5 System Integration & Deployment

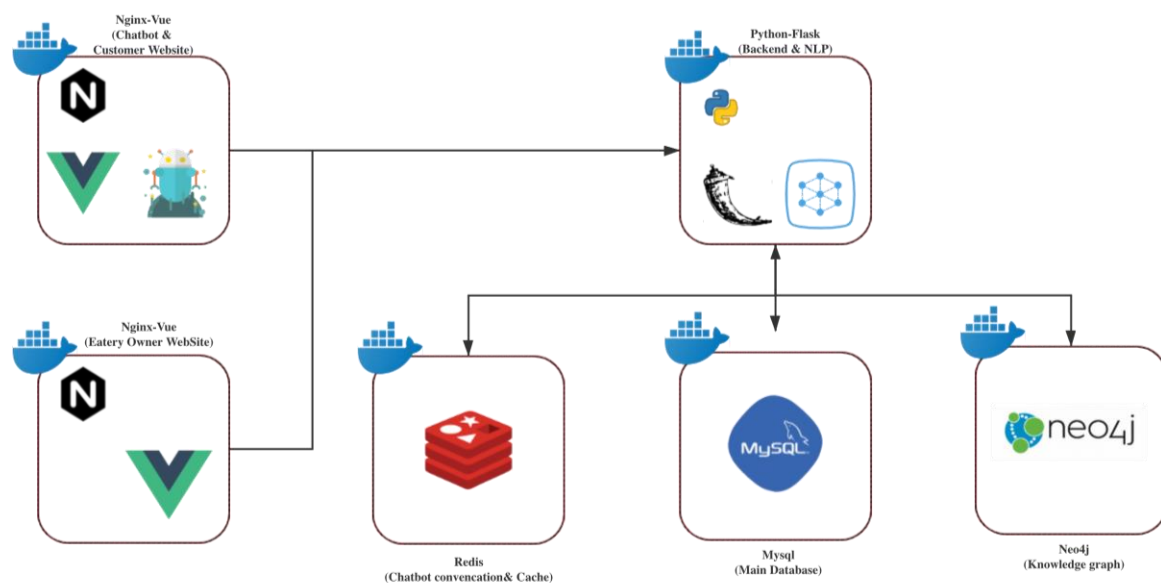


Figure 22. Deployment Structure

For the system development, all the system components need to be integrated together, including web application components, knowledge graph component (Neo4j), sentiment analysis component, chatbot backend component, Redis and MySQL database. Each component is the realization of different functions, some of which have dependencies on another component. In addition, because the system uses docker as the development tools, there is a need to specify those dependencies in deployment.

There are two steps in system integration. The first step is to handle the interdependence between components by conducting dependency analysis and integration by functions:

Conversation UI depends on following components: Customer frontend, backend RESTful API and response generation for Chatbot, Redis for storing the conversation, MySQL for storing support data, and Neo4j to provide the knowledge graph information.

Sentiment Dashboard depends on following components: Eatery Owner frontend, backend RESTful API for report, MySQL for storing the report, review and eatery data, sentiment analysis components, fake review components.

Based on the dependency analysis, the dependencies are organised and integrated in Flask Backend, Web Application Frontend and Chatbot Frontend.

The second step is to handle the dependency at application level, which is mainly the dependency between each docker container in development, namely the encapsulation of the system architecture. In this step, we mainly deal with the dependency of network and port dependencies between each docker container and the dependency of application start-up sequence.

The components dependency table is as below:

No.	Container	Dependence
1	Customer Frontend	Backend
2	Eatery Owner Frontend	Backend
3	Backend	Redis MySQL Neo4j
4	Redis	
5	MySQL	
6	Neo4j	

The system development architecture is designed based on these dependencies and uses the container orchestration tool of docker-compose to deploy and control the start-up sequence of each container. Containers can interact with each other through the network of docker. Different components can also be used to deploy on different servers (like docker swarm clusters).

## 12. Conclusion

The team has identified a business opportunity in providing a subscription-based Customer Sentiment Analysis Dashboard for eatery owners as well as free to use chatbot for consumers to search for restaurants.

The team went on to successfully design and implement an Intelligent Language Processing System which identify and filter fake online restaurant and food reviews, perform information extraction and sentiment and aspect mining and provide relevant and concise information to Consumers through a chatbot and to Restaurant Owners through a properly formatted customer sentiment analysis dashboard.

Through this process, the team has learnt immensely on the application of natural language processing techniques and how to incorporate them to provide useful services to the users. As an incidental, the team has developed an open source front end chatbot user interface.

## Reference

[1] Pretrained BERT model trained on SQuAD dataset

<https://arxiv.org/abs/1810.04805>

<https://raipurkar.github.io/SQuAD-explorer>

[2] Yelp Dataset Reviews: <https://www.yelp.com/dataset/documentation/main>

[3] Sentiment Labelled Sentences Dataset: 'From Group to Individual Labels using Deep Features', Kotzias et. al., KDD 2015.

[4] Google Cloud Natural Language API <https://cloud.google.com/natural-language/docs/analyzing-sentiment>

[5] Word vectors From Spacy <https://github.com/explosion/spacy>  
models/releases/tag/en\_vectors\_web\_lg-2.1.0

[6] GloVe: <https://nlp.stanford.edu/projects/glove/>

[7] SemEval-2014 <http://alt.qcri.org/semeval2014/task4/index.php?id=data-and-tools>

[8] Bidirectional LSTM-CRF Models for Sequence Tagging <https://arxiv.org/pdf/1508.01991v1.pdf>

[9] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. What Yelp Fake Review Filter Might Be Doing. Proceedings of The International AAAI Conference on Weblogs and Social Media (ICWSM-2013), July 8-10, 2013, Boston, USA.



## Annex A – MISC Supporting Files

No.	Filename	Description
1	Intent_Classification_Random_Forest_Model.ipynb	Script to train Intent Classification Model
2	chatbotTrainingPhases.csv	Training data for Intent Classification Model
3	LDA_Topic_modeling.ipynb	Exploration of aspect mining using LDA model
4	food_taste.ipynb	Find food taste for building knowledge graph
5	Sentiment_train.ipynb deep_learning_keras.py	Transfer Learning for sentiment score prediction
6	Google_Cloud_API.ipynb	Generate sentiment score as labeled reviews data
7	Tripadvisor_v2.otd	Script of Octoparse software to scrape Tripadvisor data
8	IE_ruleBased_statisticalBased.ipynb	Information extraction script, including both rule-based method and statistical based method.
9	Data Expectation.ipynb	Because the dataset we used for the Fake Review Detection cannot be public, please follow this notebook to get and generate the data.
10	PLP_CA_FakeReviewDetection_Filtering.ipynb	Fake review detection script for filtering fake reviews from Tripadvisor data with semi-supervised random forest model.

## Annex B – User Guide

### 1.0 Abstract

This document is about how to deploy the Food Search & Customer Analytics. Considering the system's architecture and connection between every component, our team choose the docker as our system deployment environment. Using a docker can further improve the portability of our system and can be deployed, so long as has the docker and python3.6 environment any server can only through a line command can complete the deploying and running of the whole system, at the same time because it is based on the docker and docker - compose deployment, each function module is encapsulated as a docker container, so you can compile each module of the enclosed update, improve the maintainability and expansibility of the system

### 2.0 Deployment Structure

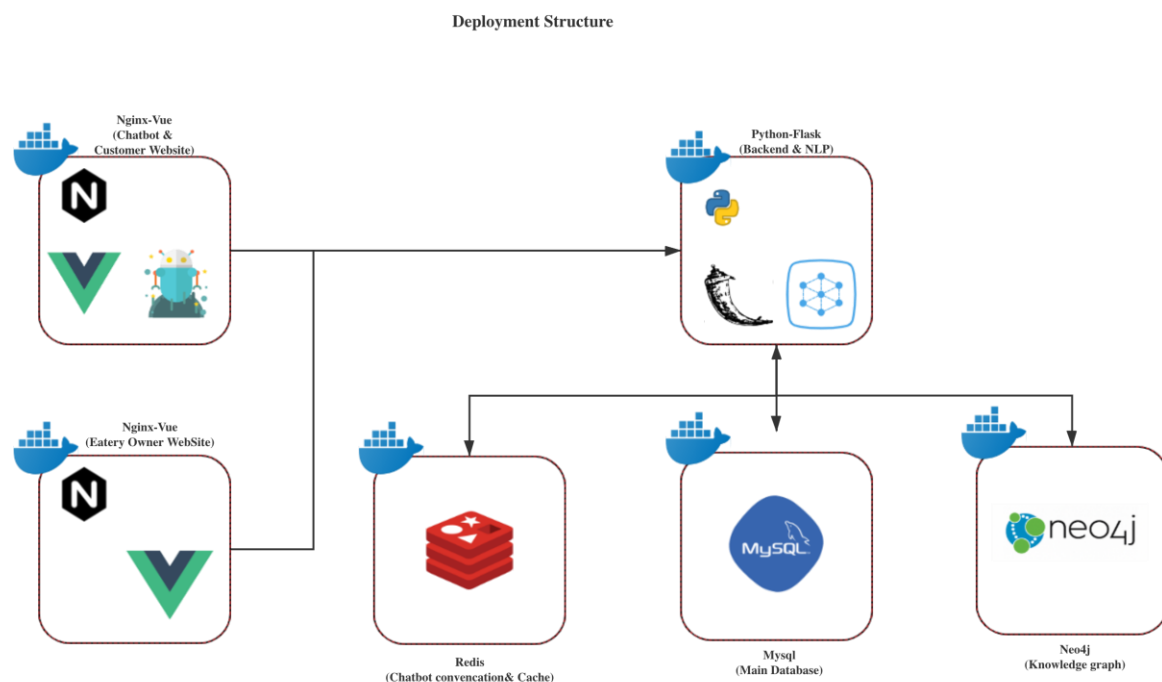


Figure 1 Deployment Structure

In the deployment structure of this system, our team use four docker images to package and deploy components.

**Nginx-Vue:** this is a docker images based on official Nginx docker image. We use the Nginx to proxy our Single-Page Application based on Vue and we use two Nginx-Vue containers to deploy the frontend for customers and the frontend for eatery owners respectively.

**Python-flask:** this is a docker images based on official python3.6 docker image. For offline deployment, our team pre-install all the python requirements of the backend and packaged it to be a new docker images -- Python-flask. And because there are several machine learning, deep learning and knowledge

graph components in the system, the image for the backend also has their dependence, like deeppavlov, Keras, Tensorflow and spacy.

MySQL: this is a docker images based on official MySQL 5.6 docker images. To prevent garbled code problems, our team modified the setting of MySQL and packaged it to be a new docker image — fofshsf/mysql-utf8.

Redis: this is a docker images of official Redis. Redis in the system is mainly responsible for the conversation data storage and query.

Neo4j: this is a docker images of official Neo4j 3.5.14, which is very good and convenient knowledge graph components. Neo4j in the system is mainly responsible for the graph creation, storage and query.

### 3.0 Dependencies

Because the system has multiple machine learning and deep learning models, it has high requirements for deployment and running environments

Local Machine or Virtual Machine:

- CPU Cores: 2 cores
- Minimum Memory: 10 G
- Recommend Memory: 16 G
- Docker Environment: version: 18.09.0
- Python environment: version: Python3.6

### 4.0 Key Files for Deployment

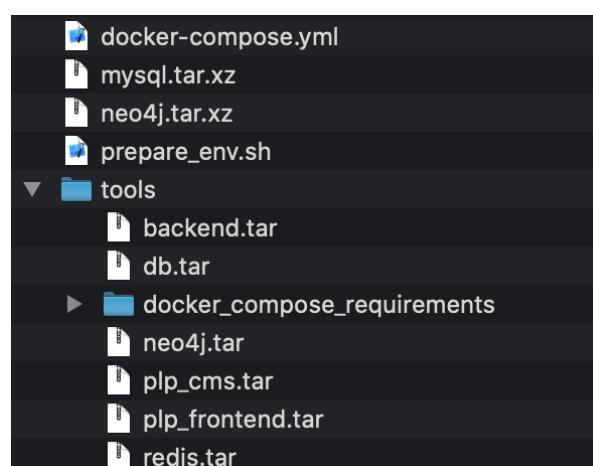


Figure 2 Files of Deployment

The docker-compose.yml, prepare\_env.sh, all the files in the tools folder are the most important files in the deployment package. Mysql.tar.xz and Neo4j.tar.xz contain all the data the system needs for the deployment.

The docker-compose.xml is to control our internal system's start and stop, which needs the python3.6 environment.

```
version: '2.1'
services:
  mysql4ca:
    image: tofshsf/mysql-utf8
    ports:
      - "3306:3306"
    volumes:
      - "/mysql/data:/var/lib/mysql"
      - "/mysql/conf/my.cnf:/etc/mysql/my.cnf"
    environment:
      MYSQL_ROOT_PASSWORD: 123456
    restart: always

  redis4ca:
    image: redis
    ports:
      - "6379:6379"
    restart: always

  neo4j4ca:
    image: neo4j:3.5.14
    ports:
      - "7474:7474"
    volumes:
      - "/neo4j/data:/data"
      - "/neo4j/logs:/logs"
      - "/neo4j/import:/var/lib/neo4j/import"
      - "/neo4j/plugins:/plugins"
    environment:
      NEO4J_AUTH: neo4j/test
    restart: always

  backend:
    image: plp_backend_prod
    ports:
      - "5000:5000"
    depends_on:
      - mysql4ca
      - redis4ca
      - neo4j4ca
    mem_limit: 10G
    memswap_limit: 10G
   oom_kill_disable: true
    restart: always

  frontend:
    image: plp_frontend
    restart: always
    ports:
      - "8080:80"
    depends_on:
      - backend

  cms:
    image: plp_cms
    restart: always
    ports:
      - "8081:80"
    depends_on:
      - backend
```

Figure 3 docker-compose.yml

The prepare\_env.sh is the shell script to prepare the environment of deployment and start our system.

```
#!/bin/sh

echo "loading docker images"
docker load < ./tools/db.tar
docker load < ./tools/plp_frontend.tar
docker load < ./tools/plp_cms.tar
docker load < ./tools/neo4j.tar
docker load < ./tools/redis.tar
docker load < ./tools/backend.tar

echo "unzip data"
tar -xvf mysql.tar.xz
tar -xvf neo4j.tar.xz

echo "install docker-compose"
pip install docker-compose -f ./tools/docker_compose_requirements/requirements.txt

docker-compose down
docker-compose up
```

Figure 4 prepare\_env.sh

The tools folder contains all the required python libraries by docker-compose and all the docker images our system needs. This folder is very prepared for the offline deployment.

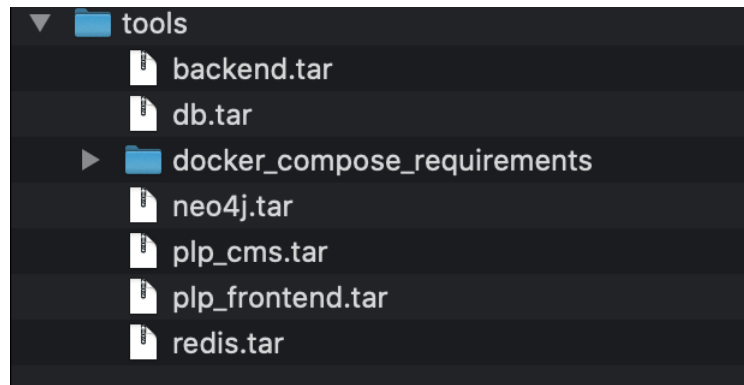


Figure 5 tools folder

## 5.0 Deployment Process

For the deployment, our team use the ISS-VM provided by Sam, which is for the docker and python3.6 environment, if you have your own server with docker and python3.6, you can pass the step of VM download.

If you have trouble with the deployment process, you can see the deployment video (<https://youtu.be/HmR8BYN2fGQ>) or you can email with any member in the teams we are very happy to help you!

1. Please download and install iss-vm according to this document

<https://github.com/telescopeuser/iss-vm/blob/master/User%20Guide%20for%20iss-vm.pdf>

Double Click the iss-vm-v17 to open the virtual machine

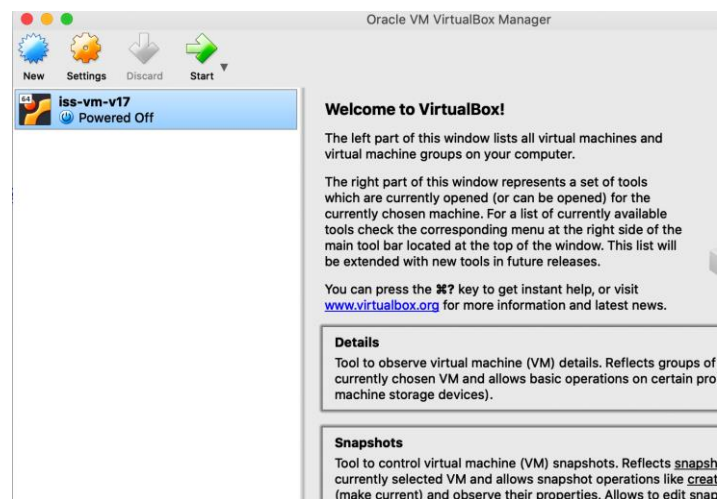


Figure 6 Install ISS-VM

2. Please download the system's docker Deployment Package (It is on the Google Drive, because it is too big for the GitHub)

<https://drive.google.com/open?id=1dQ185hNQqPTvm8oeAsfmCbopUZcS4ZCA>

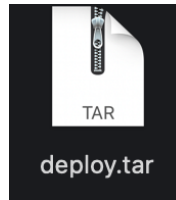


Figure 7 deploy.tar

3. Right click on VM desktop and click `Open a Terminal` to Open a Terminal
4. Please use `mv /home/iss-user/Downloads/deploy.tar /home/iss-user/Desktop` command to move the deployment package to the Desktop

```
iss-user@iss-vm: ~  
(base) iss-user@iss-vm:~$ mv /home/iss-user/Downloads/deploy.tar /home/iss-user/  
Desktop  
(base) iss-user@iss-vm:~$
```

Figure 8 mv /home/iss-user/Downloads/deploy.tar /home/iss-user/Desktop

5. Please use `cd /home/iss-user/Desktop` command to change current location

```
iss-user@iss-vm: ~/Desktop  
(base) iss-user@iss-vm:~$ mv /home/iss-user/Downloads/deploy.tar /home/iss-user/  
Desktop  
(base) iss-user@iss-vm:~$ cd /home/iss-user/Desktop/  
(base) iss-user@iss-vm:~/Desktop$
```

Figure 9 cd /home/iss-user

6. Please use `tar -xvf deploy.tar` to extract the deployment file

```
iss-user@iss-vm: ~/Desktop
(base) iss-user@iss-vm:~/Desktop$ tar -xvf deploy.tar
./_deploy
./deploy/
./deploy/._tools
./deploy/tools/
./deploy/._DS_Store
./deploy/.DS_Store
./deploy/._neo4j.tar.xz
./deploy/neo4j.tar.xz
./deploy/._prepare_env.sh
./deploy/prepare_env.sh
./deploy/._mysql.tar.xz
./deploy/mysql.tar.xz
./deploy/._docker-compose.yml
./deploy/docker-compose.yml
./deploy/tools/._db.tar
./deploy/tools/db.tar
./deploy/tools/._neo4j.tar
./deploy/tools/neo4j.tar
./deploy/tools/._redis.tar
./deploy/tools/redis.tar
./deploy/tools/._backend.tar
./deploy/tools/backend.tar
```

Figure 10 tar -xvf Deploy.tar

7. Please use `cd deploy` to enter the extracted deployment file

```
iss-user@iss-vm: ~/Desktop/deploy
(base) iss-user@iss-vm:~/Desktop$ cd deploy/
(base) iss-user@iss-vm:~/Desktop/deploy$ ls
docker-compose.yml  mysql.tar.xz  neo4j.tar.xz  prepare_env.sh  tools
(base) iss-user@iss-vm:~/Desktop/deploy$
```

Figure 11 cd Deploy/ITAS

8. Please use `bash prepare\_env.sh` command to prepare the environment and start the system

It will take about 5-8 minutes depending on the system to prepare the docker and python environment, then start the system. Because the system contains some very large models, please wait patiently for it.

```
iss-user@iss-vm: ~/Desktop/deploy
(base) iss-user@iss-vm:~/Desktop/deploy$ bash prepare_env.sh
loading docker images
e0db3ba0aaaa: Loading layer 58.51MB/58.51MB
605f8f2fe1e5: Loading layer 338.4kB/338.4kB
4dac9b6b28ce: Loading layer 10.44MB/10.44MB
1a527f11e03e: Loading layer 4.472MB/4.472MB
cee57cdf5101: Loading layer 1.536kB/1.536kB
76db703007bc: Loading layer 46.15MB/46.15MB
6b5c7baa4da8: Loading layer 34.3kB/34.3kB
802f9b63e008: Loading layer 3.584kB/3.584kB
52313adc2e10: Loading layer 320MB/320MB
3a31b7a7e513: Loading layer 15.87kB/15.87kB
a28a6d28cd00: Loading layer 1.536kB/1.536kB
8af118e1d6b2: Loading layer 3.584kB/3.584kB
Loaded image: fofshsf/mysql-utf8:latest
f2cb0ecef392: Loading layer 72.48MB/72.48MB
71f2244bc14d: Loading layer 58.11MB/58.11MB
55a77731ed26: Loading layer 3.584kB/3.584kB
cd4de7a6bc02: Loading layer 2.048kB/2.048kB
b1d6b9bdf0b5: Loading layer 9.04MB/9.04MB
6aa1286aaeb6: Loading layer 3.584kB/3.584kB
Loaded image: plp_frontend:latest
b8624c70e0ce: Loading layer 131.1kB/10.16MB
```

Figure 12 bash prepare\_env.sh

```
iss-user@iss-vm: ~/Desktop/deploy
backend_1 | WARNING:tensorflow:From /usr/local/lib/python3.6/site-packages/dee
ppavlov/core/models/tf_model.py:54: The name tf.train.Saver is deprecated. Pleas
e use tf.compat.v1.train.Saver instead.
backend_1 |
backend_1 | WARNING:tensorflow:From /usr/local/lib/python3.6/site-packages/ker
as/backend/tensorflow_backend.py:1238: calling reduce_sum_v1 (from tensorflow.py
thon.ops.math_ops) with keep_dims is deprecated and will be removed in a future
version.
backend_1 | Instructions for updating:
backend_1 | keep_dims is deprecated, use keepdims instead
backend_1 | 2020-03-29 12:58:22.595667: W tensorflow/core/framework/allocator.
cc:107] Allocation of 1285165200 exceeds 10% of system memory.
backend_1 | 2020-03-29 12:58:23.293548: W tensorflow/core/framework/allocator.
cc:107] Allocation of 1285165200 exceeds 10% of system memory.
backend_1 | 2020-03-29 12:58:23.898757: W tensorflow/core/framework/allocator.
cc:107] Allocation of 1285165200 exceeds 10% of system memory.
backend_1 | done
backend_1 | done
backend_1 | done
backend_1 | * Serving Flask app "app.app" (lazy loading)
backend_1 | * Environment: development
backend_1 | * Debug mode: on
backend_1 | * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Figure 13 deployment done

If you can see the above screenshot, the system has started. Congratulation! Now you can open the Chrome and go to our two website.

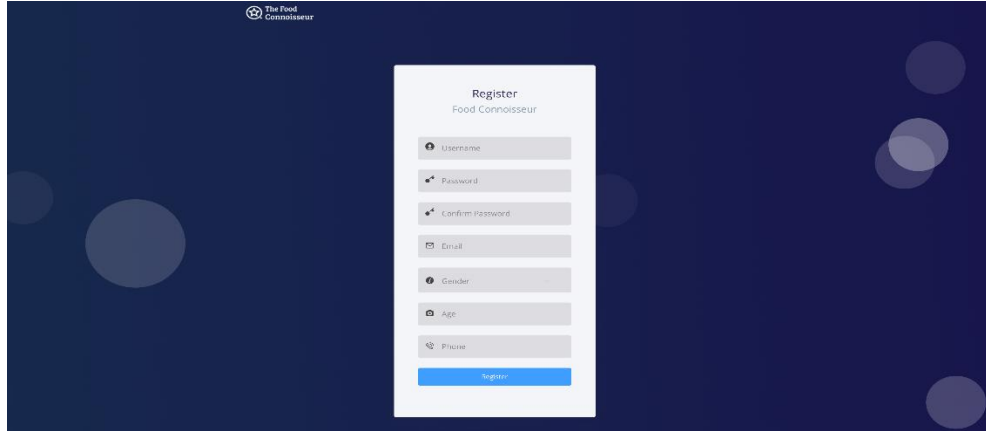
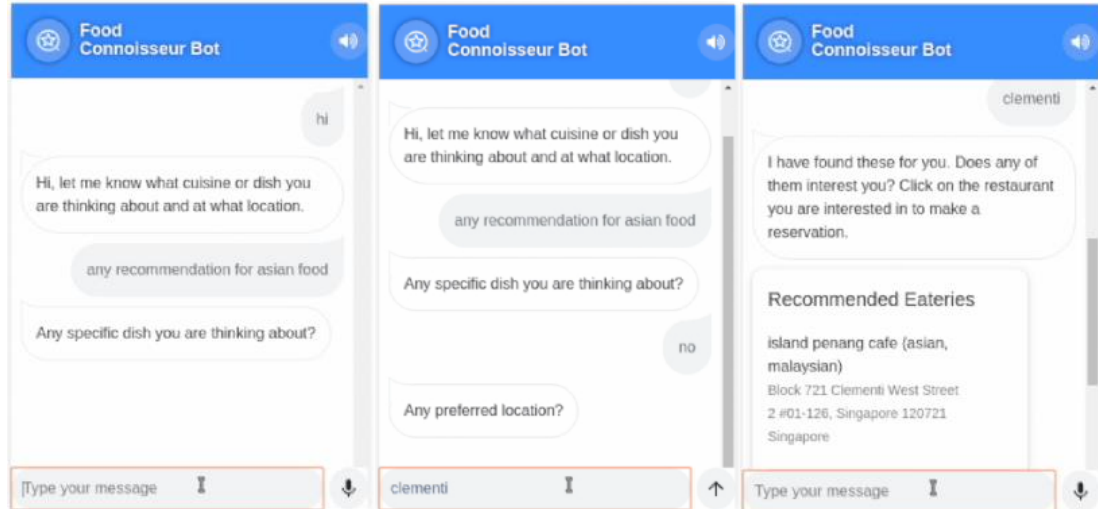
localhost:8080 Customer Website

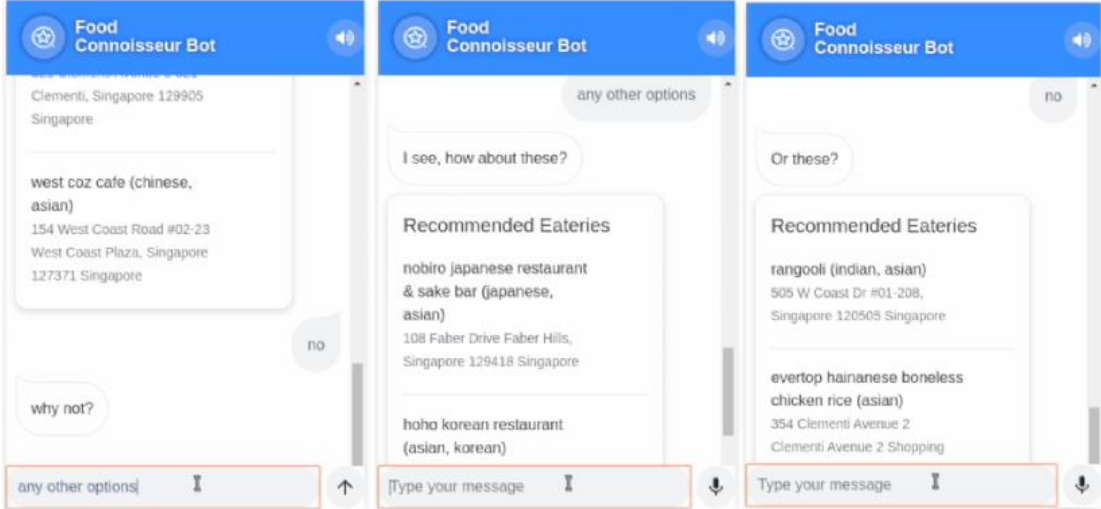
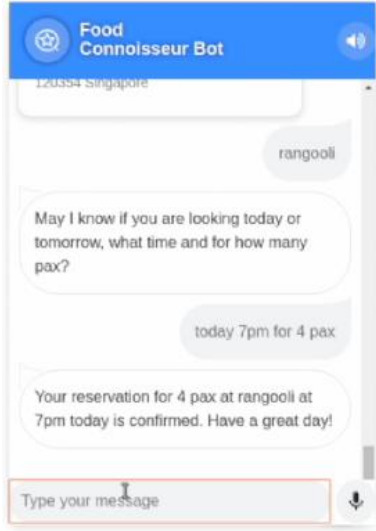
localhost:8081 Eatery Owner Website

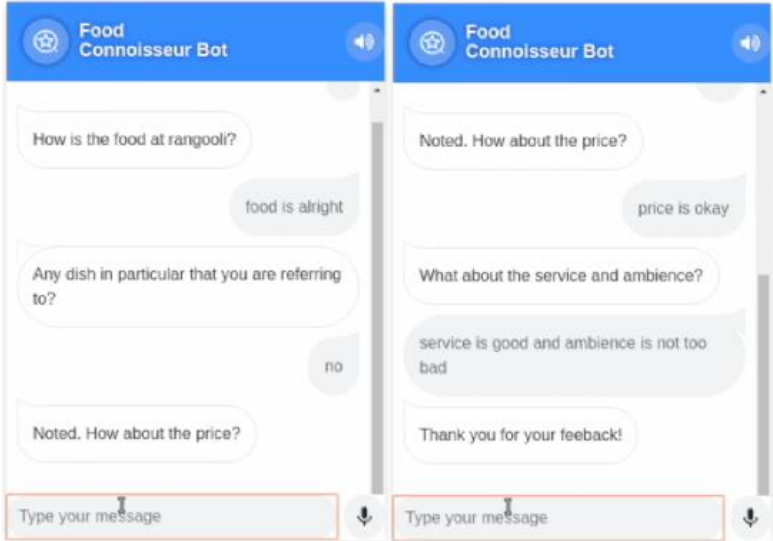
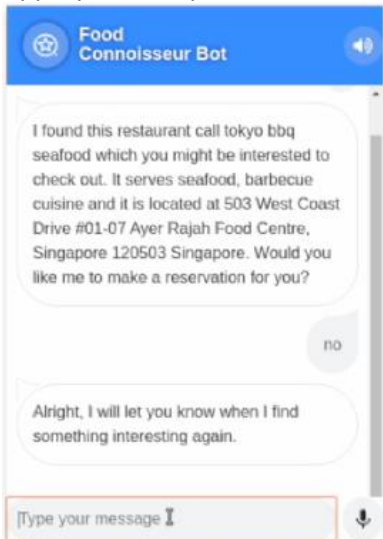
Please refer to the User Guide of the system to use the system.

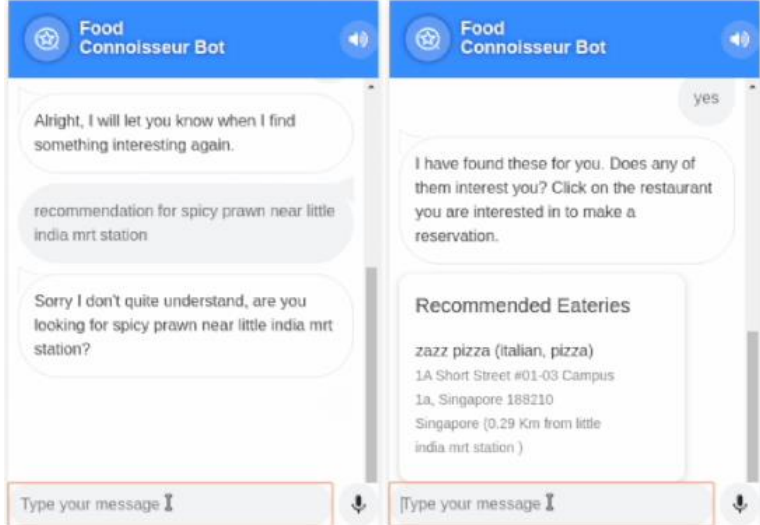
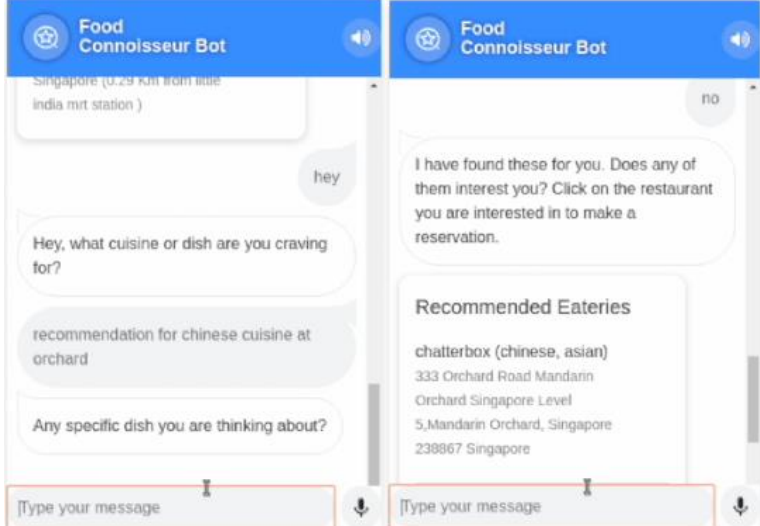


## Annex C – Test Scenarios


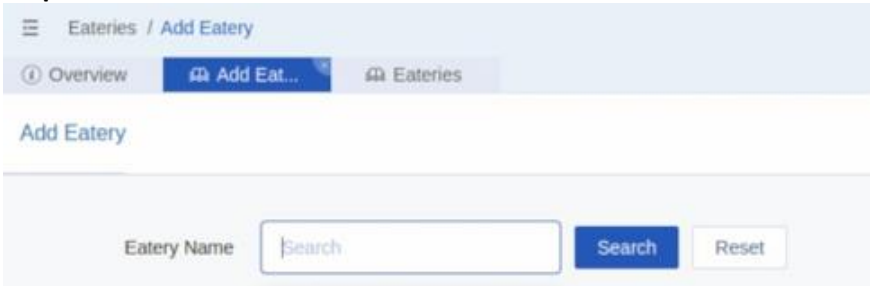
Scenario	Steps and Input Values	Expected Output
<b>Chatbot</b> <b>Scenario 1: User Registration</b>	<i>Inputs:</i> <ol style="list-style-type: none"> <li>1. <i>enter username</i></li> <li>2. <i>enter password of at least 6 characters</i></li> <li>3. <i>re-enter password</i></li> <li>4. <i>enter email</i></li> <li>5. <i>enter gender</i></li> <li>6. <i>enter age</i></li> <li>7. <i>enter phone</i></li> </ol>	 <p>Successful registration and logged in into the application.</p>
<b>Chatbot</b> <b>Scenario 2: Search by cuisine and location and make reservation</b>	<p>Step 1: Give requirements to Food Connoisseur Bot</p> <p><i>Inputs:</i></p> <ol style="list-style-type: none"> <li>1. <i>hi</i></li> <li>2. <i>any recommendation for asian food</i></li> <li>3. <i>no</i></li> <li>4. <i>clementi</i></li> </ol>	

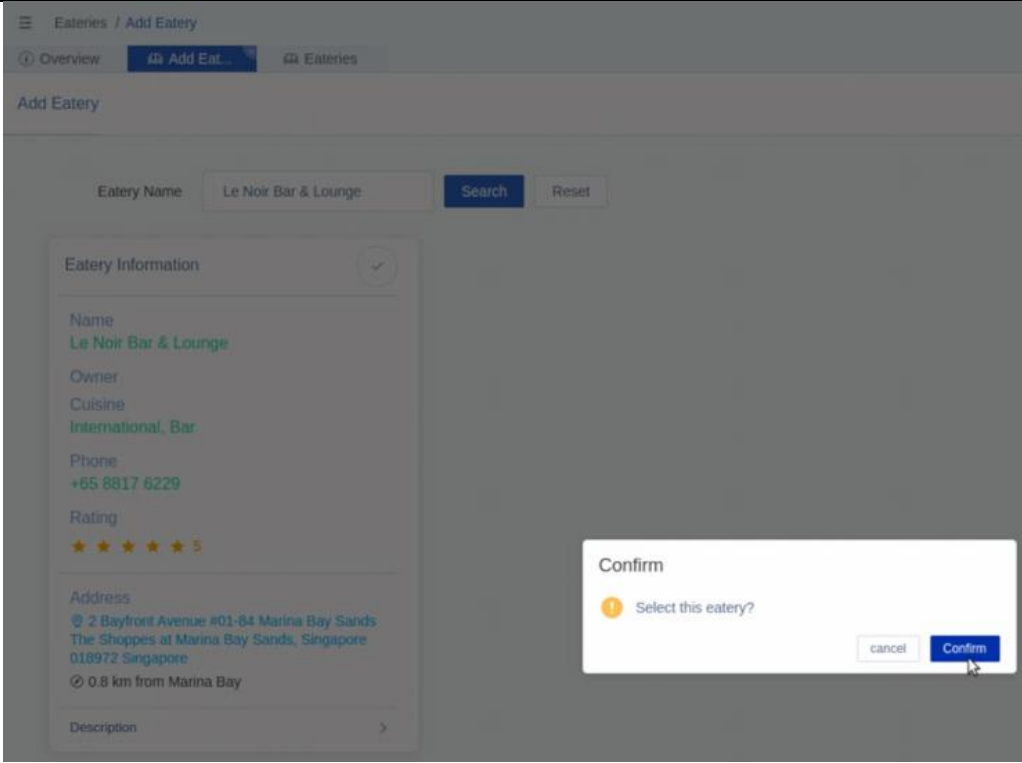
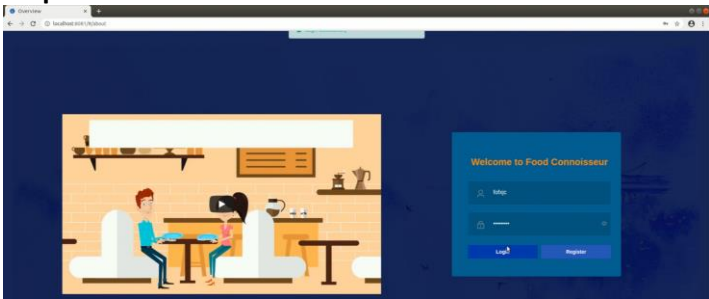
	<p>Step 2: Look through recommendations</p> <p><i>Inputs:</i></p> <ol style="list-style-type: none"> <li>1. <i>no</i></li> <li>2. <i>any other options</i></li> <li>3. <i>no</i></li> </ol>	<p>Appropriate responses from Food Connoisseur Bot.</p> 
	<p>Step 3: Select restaurant and give reservation details</p> <p><i>Inputs:</i></p> <ol style="list-style-type: none"> <li>1. <i>click on a restaurant</i></li> <li>2. <i>today 7pm for 4 pax</i></li> </ol>	<p>Appropriate responses from Food Connoisseur Bot.</p> 

<b>Chatbot Scenario 3: Give Feedback</b>	<p><i>Steps/Inputs:</i></p> <ol style="list-style-type: none"> <li><i>1. close chatbot window and click on browser refresh button after successfully making reservation.</i></li> <li><i>2. launch chatbot window</i></li> <li><i>3. food is alright</i></li> <li><i>4. no</i></li> <li><i>5. price is okay</i></li> <li><i>6. service is good and ambience is not too bad</i></li> </ol>	<p>Appropriate responses from Food Connoisseur Bot.</p> 
<b>Chatbot Scenario 4: Proactive Restaurant Recommendation</b>	<p><i>Steps/Inputs:</i></p> <ol style="list-style-type: none"> <li><i>1. close chatbot window and click on browser refresh button after at least one successful search and clearing any outstanding feedback.</i></li> <li><i>2. launch chatbot window</i></li> <li><i>3. no</i></li> </ol>	<p>Appropriate responses from Food Connoisseur Bot.</p> 

<p><b>Chatbot Scenario 5:</b> <b>Search by Taste, Food and MRT Station</b></p>	<p><i>Steps/Inputs:</i></p> <ol style="list-style-type: none"> <li><i>1. recommendation for spicy prawn near little india mrt station</i></li> <li><i>2. yes</i></li> </ol>	<p>Appropriate responses from Food Connoisseur Bot.</p>  <p>The screenshot shows two chat windows. The left window shows the bot's initial response: 'Alright, I will let you know when I find something interesting again.' followed by a recommendation for 'zazz pizza (italian, pizza)' at '1A Short Street #01-03 Campus 1a, Singapore 188210', which is '0.29 Km from little india mrt station'. The right window shows the bot asking 'I have found these for you. Does any of them interest you? Click on the restaurant you are interested in to make a reservation.' and displaying the same recommendation for 'zazz pizza'.</p>
<p><b>Chatbot Scenario 6:</b> <b>Casual Persona and Self Learning Capability</b></p>	<p><i>Steps/Inputs:</i></p> <ol style="list-style-type: none"> <li><i>1. hey</i></li> <li><i>2. recommendation for chinese cuisine at orchard</i></li> <li><i>3. no</i></li> </ol>	<p>Appropriate responses from Food Connoisseur Bot.</p>  <p>The screenshot shows two chat windows. The left window shows the bot's initial response: 'Singapore (0.29 Km from little india mrt station )'. The right window shows the bot asking 'I have found these for you. Does any of them interest you? Click on the restaurant you are interested in to make a reservation.' and displaying a recommendation for 'chatterbox (chinese, asian)' at '333 Orchard Road Mandarin Orchard Singapore Level 5, Mandarin Orchard, Singapore 238867 Singapore'.</p>

<p><b>Chatbot</b></p> <p><b>Scenario 7:</b></p> <p><b>Questions on</b></p> <p><b>Local Food</b></p> <p><b>Culture</b></p>	<p><i>Steps/Inputs:</i></p> <ol style="list-style-type: none"> <li><i>1. where does the locals mostly eat at?</i></li> <li><i>2. what is hawker centre?</i></li> <li><i>3. why do Singaporean eat at hawker centre?</i></li> <li><i>4. what are the popular hawker centre?</i></li> <li><i>5. what is food to Singaporean?</i></li> </ol>	<p><b>Appropriate responses from Food Connoisseur Bot.</b></p> <div data-bbox="927 229 1229 427"> <p>Food Connoisseur Bot</p> <p>where does the locals mostly eat at?</p> <p>hawker centres, coffee shops or food courts rather than restaurants</p> </div> <div data-bbox="1240 229 1543 427"> <p>Food Connoisseur Bot</p> <p>what is hawker centre?</p> <p>Hawker centres, or open air food courts</p> </div> <div data-bbox="1554 229 1856 427"> <p>Food Connoisseur Bot</p> <p>why do Singaporean like to eat at hawker centre?</p> <p>due to convenience, a wider range of options and affordability</p> </div> <div data-bbox="927 464 1229 662"> <p>Food Connoisseur Bot</p> <p>what are the popular hawker centre?</p> <p>Telok Ayer Market, Lau Pa Sat and Newton Food Centre</p> </div> <div data-bbox="1240 464 1543 662"> <p>Food Connoisseur Bot</p> <p>what is food to Singaporean?</p> <p>food is viewed as crucial to its national identity and a unifying cultural thread</p> </div>
---	---	---

Scenario	Steps and Input Values	Expected Output
<b>Sentiment Dashboard</b> <b>Scenario 1:</b> <b>Register Eatery</b>	<p><b>Step 1</b> After registering user account, log in</p> <p><b>Step 2</b> Add eatery name</p>	<p><b>Step 1</b></p>  <p>The illustration shows a warm, orange-toned cafe interior. Two people, a man and a woman, are sitting at a small table, looking at their phones. In the background, there are shelves with coffee-making equipment. Overlaid on the right side is a dark blue login/register screen with the text 'Welcome to Food Connoisseur'. It features input fields for 'tester' and a masked password, along with 'Login' and 'Register' buttons.</p> <p><b>Step 2</b></p>  <p>The screenshot shows a web interface for adding a new eatery. At the top, there's a breadcrumb 'Eateries / Add Eatery' and a navigation bar with 'Overview', 'Add Eat...', and 'Eateries' tabs. Below this is a section titled 'Add Eatery'. It contains a form with a label 'Eatery Name', a text input field with the placeholder 'Search', a blue 'Search' button, and a 'Reset' button.</p>

	<p><b>Step 3</b></p> <p>Click search button, the corresponding Eatery Information Card will be shown, and click confirm button to generate analysis dashboard</p>	 <p>The screenshot shows a web application interface for adding a new eatery. At the top, there's a navigation bar with 'Overview', 'Add Eat...', and 'Eateries'. Below this is the 'Add Eatery' form. It has a search bar with 'Le Noir Bar &amp; Lounge' entered and a 'Search' button. A 'Reset' button is also present. Below the search bar, there's a card titled 'Eatery Information' with a checkmark icon. The card contains the following details: Name (Le Noir Bar &amp; Lounge), Owner, Cuisine (International Bar), Phone (+65 8817 6229), Rating (5 stars), Address (2 Bayfront Avenue #01-84 Marina Bay Sands, The Shoppes at Marina Bay Sands, Singapore 018972), and a note that it's 0.8 km from Marina Bay. A 'Description' field is at the bottom. A 'Confirm' dialog box is overlaid on the right, asking 'Select this eatery?' with 'cancel' and 'Confirm' buttons.</p> <p>Successful entered into the Overview Page</p>
<p><b>Sentiment Dashboard Scenario 2: Check Overview Page</b></p>	<p><b>Step 1</b></p> <p>log in</p>	<p><b>Step 1</b></p>  <p>The screenshot shows a login page for 'Food Connoisseur'. On the left, there's an illustration of a person sitting at a table in a cafe. On the right, there's a login form with fields for 'Email' and 'Password', and buttons for 'Log In' and 'Register'.</p>

**Step 2**  
Check Overview Page

Section 1  
User information

Section 2  
A sample of incoming reviews in prompt style

Section 3  
Overall sentiment score for service, food, price, and environment categories

Section 4  
Click the Competitor Button from Section 1. Explore the map and click each blue pinpoint to check competitor information

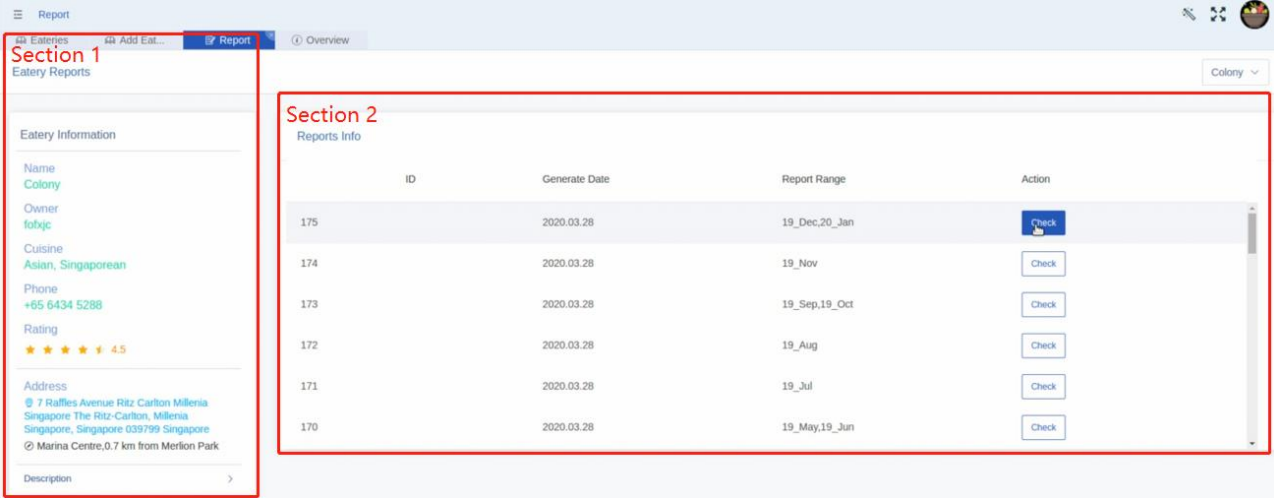
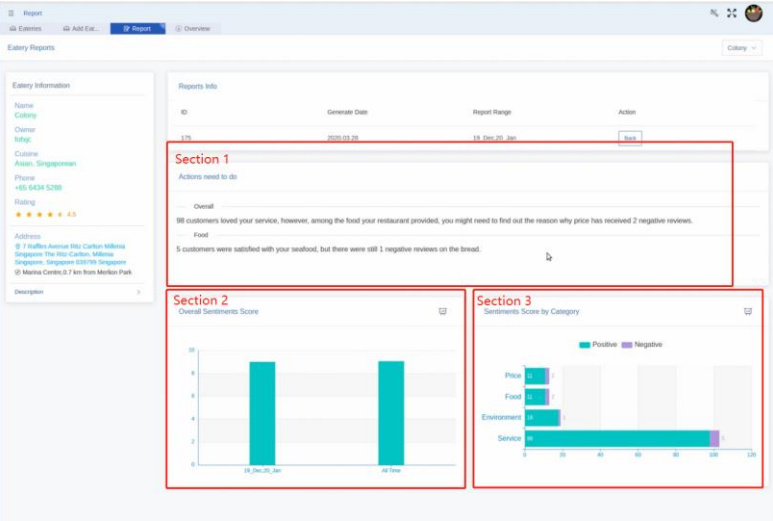
**Step 3**  
Click Reports button to enter Scenario 3: Analysis Dashboard

**Step 2 and Step 3**

The screenshot displays a web application interface for a food business. The top navigation bar includes links for Overview, Eateries, Add Eat..., Report, and Overview. The main content area is divided into several sections:

- Section 1: Personal Info** (highlighted with a red border) shows the user's profile for 'fofxjc', a Food Connoisseur in Singapore, with a join date of 2020.03.08 and 2 eateries listed.
- Section 2: Reviews** (highlighted with a red border) displays a sample of incoming reviews in a prompt style, such as 'fantastic food and wonderful service!' and 'excellent customer service, food is decent, amazing experience with our staff'.
- Section 3: Sentiment Scores** (highlighted with a red border) shows overall sentiment scores for Service (9/10), Food (10/10), Price (10/10), and Environment (10/10).
- Section 4: Competitors** (highlighted with a red border) features a map of Singapore with blue pinpoints indicating competitor locations. A tooltip for 'The Wine & Gourmet Friends' shows its address as 48 Bukit Pasoh Road 01-01, Singapore 089859, and a rating of 4.5.



<p><b>Sentiment Dashboard</b> <b>Scenario 3:</b> <b>Choose Analysis Dashboard</b></p>	<p><b>Step 1</b> check basic information in section 1</p> <p><b>Step 2</b> Click the Check button in section 2 to enter the corresponding Analysis Dashboard</p>	<p>User interface</p> 
<p><b>Sentiment Dashboard</b> <b>Scenario 4:</b> <b>Check Analysis Dashboard</b></p>	<p><b>Step 1</b> Check Section 1 to 3 for Actions need to do, Overall Sentiments Score, and Sentiment Score by Categories</p> <p><b>Step 2</b> In section 3, click each Price, Food, Environment and Service elements in bar chart to expand detail sections.</p>	<p><b>Step 1 and Step 2</b></p> 

### Step 3

In Service section, click face icon to switch positive and negative reviews.

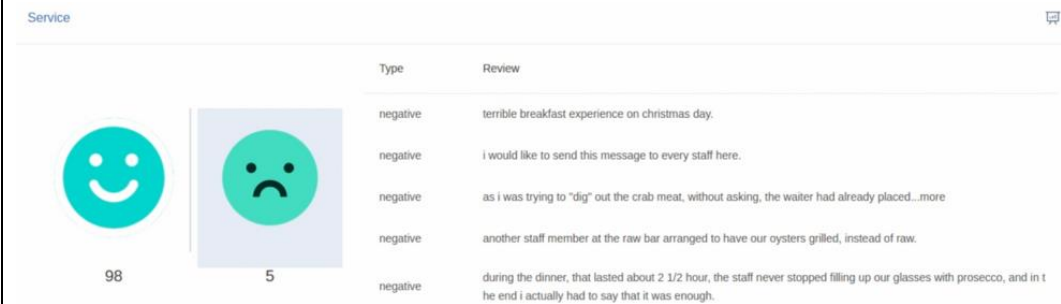
### Step 4

In Food section, click each food item to check its corresponding reviews.

### Step 5

In Price section, click Negative, Positive, Valuation elements in radar chart to check corresponding reviews

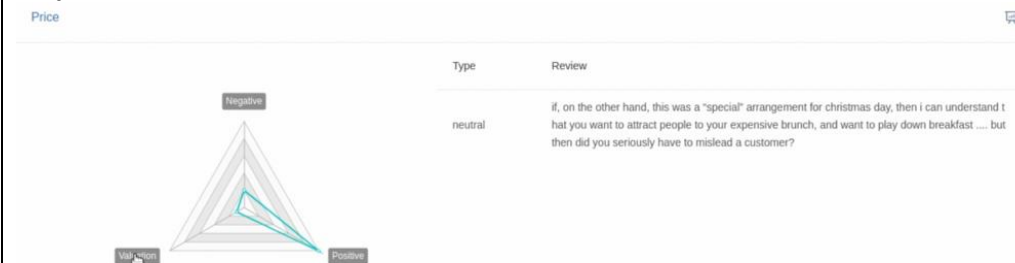
### Step3



### Step 4




### Step 5




### Step 6

In Environment Section, click leaves icons to switch negative and positive reviews

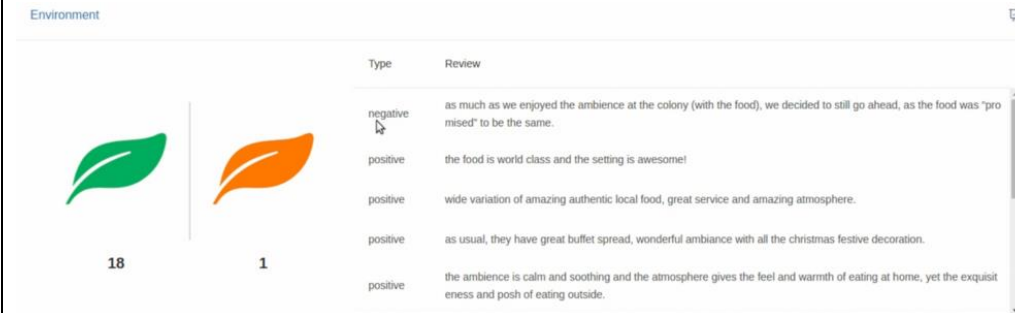
### Step 7

Click  button In sections 2 and 3 to expand trend analysis line chart

### Step 8

switch  Service - Pos button to show or hide corresponding trend analysis line

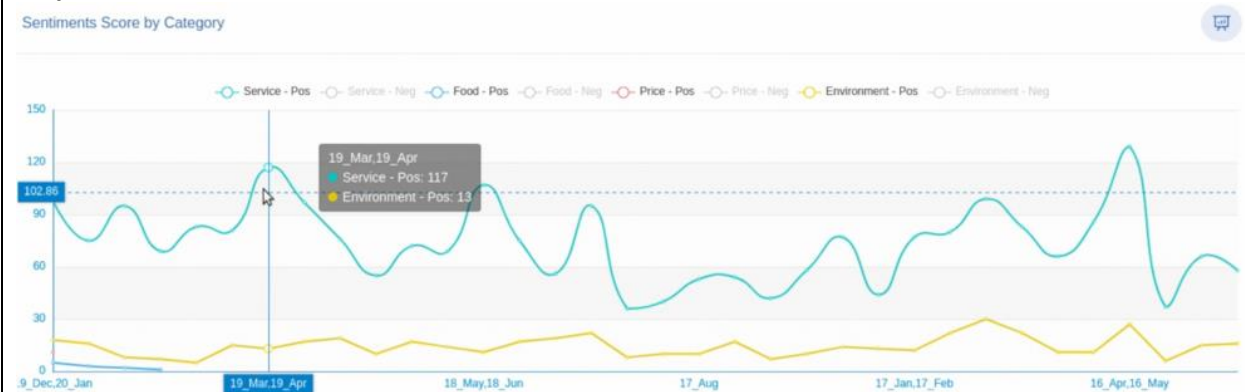
### Step 6



### Step 7



### Step 8



## **Annex D – Team Members Individual Reflection**

## CA1 – Individual Project Report

Your Name:	Alfred Tay Wenjie
Certificate:	Graduate Certificate in Practical Language Processing

### 1. Personal contribution to the group project

Ideate and formalize business use case and conceptualized business model. Perform overall solution design. Design and implement conversational user interface in the form of a specialist chatbot. Implement from scratch a Finite State Transducer (FST) for intent detection and entity extraction mainly using python list data type. Handcrafted phrases for the FST. Conceptualised and implemented a 2-layer intent detection and clarification architecture which gives the chatbot its self-learning capability. Build and trained a machine learning model for the 2<sup>nd</sup> layer intent classification. Incorporated pre-trained machine reading comprehension model for the chatbot to answer local food culture related questions. Facilitated project discussions, scoping of work and tracking of project progress.

### 2. What you have learnt from the project

I have learnt about the various techniques used in natural language processing from pre-processing to named entity extraction to text classification. I have learnt that the NLP techniques largely falls into rule-based, statistic-based and deep learning. Rule-based techniques are effective when the domain is narrow and it is very accurate as long as the rule covers the scenarios. The downside is that it is almost impossible to cover all scenarios. On the other hand, statistical techniques can be more generally applied but requires large training corpus to achieve high accuracy and labelled data is not always readily available. One possible design pattern is to deploy rule-based technique together with and before statistic-based technique to take advantage of the high accuracy of rule-based technique and the wider coverage of statistic-based technique.

### 3. How you can apply the knowledge and skills in other situations

One area in most organization where NLP can be very useful is in the area of knowledge management. This is especially so for larger organizations with thousands of employees. A central NLP knowledge management system would be able to ingest learning points in real-time and immediately push these learning points to employees at other sites before the same costly mistake recurs in the organization. A conversational user interface can be used to interact with the employee to collect and dispatch the learning points. It can also get immediate feedback from employee on the relevancy of the dispatched learning points and use that data point to subsequently improve the performance of the system.

## CA1 – Individual Project Report

Your Name:	Li Xinlin
Certificate:	Graduate Certificate in Practical Language Processing

### 1. Personal contribution to the group project

At the beginning of the project, we chose the overall topic, set up the proposal, defined the objective and split tasks together. Then I am mainly responsible for information extraction and knowledge graph.

For information extraction, I mainly extract the food name entities from the reviews crawled. I used both rule-based method and statistical method. For the rule-based method, I combined several methods, including dependency parsing, noun chunking, structural pattern recognition and so on, extracted dishes names entities and reviewed by myself manually about the result accuracy since no labelled corpus to reference. Then I built a labelled corpus used the results I got, using BIOES-style annotation scheme. Using this corpus, after some preprocessing works, I trained a Bi-directional LSTM-CRF model for possible future use when the data volume is significantly larger.

For knowledge graph, at the beginning, I did related research, including the advantages and necessity of knowledge graphs. Based on the excellent relational search capabilities of graph databases, it is necessary to use graph databases. I also investigated the differences between different graph databases, and finally chose to use neo4j based on its highly scalable, fast execution, high availability and so on. Then I designed the structure of the knowledge graph database, including the node's types, node's properties, relationship's types and relationship's properties. After that, I did some data processing works in preparation for importing into the database, for example, calculating which restaurants are near each subway station and their distance from the restaurant based on the coordinates. Then import all the relevant data into database. I also wrote several querying and dynamic creating functions for chatbot and other part of our system to use.

At last, I also finished writing the report of the Information Extraction section and the Knowledge graph section.

### 2. What you have learnt from the project

Through the CA, I have a deeper understanding about natural language processing tasks that not just artificial intelligence plays an important role in it, linguistics knowledge is also significant. For specific domain like our project's food entities extraction domain, there is no well labelled data can be used to train supervised model, the rule based on the linguistic shows its power. And I also learned about the Bi-directional LSTM-CRF model which is a state-of-art-statistical NER model.

And I also get familiar about how to design, build, query and use Neo4j, a typical graph database which is particularly powerful in terms of deep and complex queries than traditional relational databases.

### **3. How you can apply the knowledge and skills in other situations**

With the dramatic growth of text information on web pages, information extraction is particularly important, which is to extract knowledge from semi-structured and unstructured text sources. I firmly believe that the information extraction experience brought to me by this project will always come in handy in future work.

With the rapid development of industries such as social networking, e-commerce, finance, retail, and the Internet of Things, the real society has woven a large and complex network of relationships, and traditional databases are difficult to handle relational operations. The relationship between the data that the big data industry needs to process increases geometrically with the amount of data. The Graph database I learned in this project which supports massive complex data relationship operations is well suited for this type of problems, can be applied widely.

## CA1 – Individual Project Report

Your Name:	Wang Zilong
Certificate:	Graduate Certificate in Practical Language Processing

### 1. Personal contribution to the group project

Brainstorm business scenario and implementation techniques;  
Scraped data from restaurants portals; collected existing labeled data from public dataset; created additional labeled data using API;  
Preprocess data for multiple NLP tasks;  
Explored topic modeling method for classification;  
Trained transfer learning model for sentiment analysis task;  
Perform aspect mining for fine-grained restaurants analysis task;  
Recorded demo video, wrote test cases and reports

### 2. What you have learnt from the project

I have applied what I have learnt from Practical Language Processing Graduate certificate and gained practical experience on how to perform natural language processing. From module 1, I learned how to preprocess textual data, categorize text, and extract information. From module 2, I applied sentiment mining methodology into practice; learned how to extract entities and aspects from opinions. From module 3, I gained hands-on experience in building and tuning machine learning models for NLP tasks. Apart from this, I have also gone through how to build an end-to-end intelligent NLP system from defining project scope to final user interface representations.

### 3. How you can apply the knowledge and skills in other situations

For document classification, sentiment mining, and aspect mining techniques, I can combine these methods and use the result as investment contribution factors for reference. It will be useful to analyze massive economic and finance-related media information and provide trend analysis, public opinion analysis, and event-driven analysis.

The knowledge graph from our project has a lot of potentials, it can be also deployed on many other scenarios, for example by building the graph-structured database, it could dig relationship information for complicated anti-fraud detection. The conversational UI is another promising area where more intelligent responses could yield more potential commercial value.



## CA1 – Individual Project Report

Your Name:	Xu JiaChen
Certificate:	Graduate Certificate in Practical Language Processing

### 1. Personal contribution to the group project

In this group project, I am mainly responsible for whole system architecture, integration and deployment, including two frontend website design and implementation, Conversation UI frontend and backend design and implementation without the response generation part, web application backend design and implementation, design and implementation of the integration and deployment structure. In the meantime, I also responsible for the Fake Review Detection part, including looking for the dataset, data preprocessing and feature extraction, model training and application.

### 2. What you have learnt from the project

What I learned in this project is mainly divided into two aspects:

Intelligent System Architecture:

In the intelligent system architecture level, I learnt how to really design a system which is deeply integrated with machine learning or deep learning technologies. Unlike previous project, I am responsible for the chatbot and sentiment analysis component integration. Those two components needs real-time processing, which means the system needs to be able to do real-time response for user's question and generate sentiment analysis report. That is very challenging for the system architecture. After spending lots of efforts on researching, I used the Redis and Singleton Mode to meet this requirement.

Getting of deeper understanding of Nature Language Processing:

Through the training of fake review detection model, the data I used to train the model in the project came from professor Liu Bing's papers and dataset. I spent a lot of time reading his papers and learned a new data processing method, change a different perspective on data processing, such as using only review itself may not get a good model performance, maybe add some feature about the reviewer. That is a new aspect for me, in the past, I thought the NLP data preprocessing and feature engineering should only focus on the text itself, such as POS tagger. From this aspect, I learned a new way of data processing, which is multi-domain multidimensional data integration and feature extraction. I also learned that it is necessary to read the papers frequently and understand the latest research results.

### 3. How you can apply the knowledge and skills in other situations

I think what I learnt from this project about the intelligent system architecture design and the integration of AI technology can be very easy to apply to most of commercial project. Now there are a lot of company and the team hopes to add intelligent components into their own system, such as

add a Conversational UI, like chatbot, sentiment analysis or fake review detection. For these requirements and projects, I can use what I learnt from this project about how to structure a intelligent system with high availability to help them to intelligentize the system or some component of their system, because from a system architecture level, machine learning or deep learning is a tool for implementation of certain components or function. With that intelligent system architecture-level vision is very important for the growth of a full stack programmer. For the knowledge and skills about the Nature Language Processing in this project, not only word2vec and word embedding, but also better understand and application of the PyTorch and the architecture of those state-of-art NLP models. With those skills, I may not be able to model and process an entire NLP project, but I can better understand the project and know how to choose the technology to process different types of data. At the same time, I can modify or optimize those state-of-art according to what I have learnt, so as to meet the needs of future projects