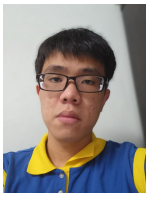
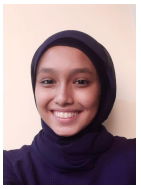
UNIVERSITI MALAYSIA PAHANG

BSD2513 ARTIFICIAL INTELLIGENCE
GROUP PROJECT
TITLE: RESTAURANT'S CHATBOT
LECTURER: DR KU MUHAMMAD NA'IM KU KHALIF

MATRIC ID	NAME	PHOTOS
SD20019	LEONG TENG MAN	
SD20024	SHAESTA KHAN BINTI SHEH RAHMAN	
SD20040	SOON KIEN YUAN	
SD20042	WAN MAISARAH BINTI WAN ALIAS	

1.0 Executive Summary

1.1 Project Description:

By 2025, the market for AI that can communicate with people is expected to be worth $13.9 billion. Organizations all over the world are clearly adopting chatbots, virtual agents, and other conversational AI applications in large numbers. However, when it comes to adopting these new technologies, the restaurant industry lags behind.

Most restaurants use robot waiters and order machines. This project uses task-oriented chatbots since no Malaysian restaurants have them. Real-world apps employ it to help customers achieve a goal, such making a restaurant reservation, ordering pizza, or booking a ticket. This chatbot reduces order and delivery times. It'll be like the restaurant employs a self-service order system, making the procedure efficient as guests take turns purchasing meals. The restaurant wouldn't require a server to accept orders, saving money. Chatbots may offer effective customer service by automatically answering basic inquiries.

1.2 Problem to Be Solved

1. To speed up the order and delivery process
2. To reduce labour expense
3. To improve the dining experience for customers.

1.3 Data Description

The dataset is the data we make ourselves, as none of the websites provide the suitable dataset that can be used for our programme. The data consists of 3 key elements , which are tags, questions to be asked by customers and answers from the restaurants. The data type is string or text. The dataset is written in JSON form.

2.0 Summary of the Project Context and Objectives

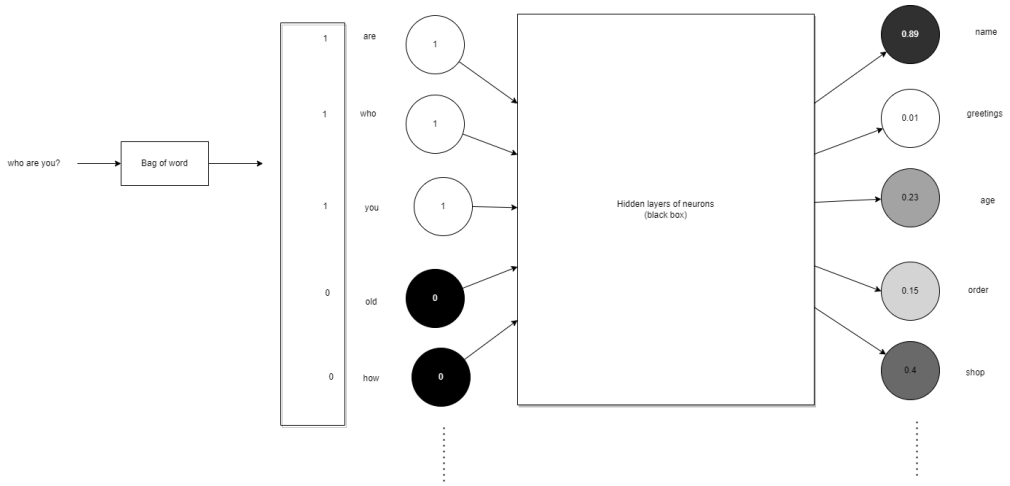
2.1 Summary of the Project Context

Chatbots are one of the most important real-world commercial applications of artificial intelligence (AI) software that exists today. Businesses all over the globe are employing them to eliminate the need for costly human engagement with clients. This project is about communicating with customers of restaurants or cafes by using deep learning. These methods can ease the process of taking orders and save time too. In the code, we are using packages such as tensorflow, and numpy. After running the code, there will be “You're talking with the bot (type quit to quit)!” and an answer in the box providing you want to quit or not. If your answer is “I didn't get that, try again,” then the code will run again and ask the same question until you type in "quit,” at which point the chatbot will stop.

2.2 Objectives

1. To save time for food ordering.
2. To reduce the number of workers in order to cut costs.
3. To incorporate AI into our daily routines.

3.0 Methodology



The figure above shows a simplified idea of an ML-based chatbot. Indeed, the chatbot is more like a classifier trying to understand the intents behind a sentence. The program chooses a random response based on intent with the highest activation. The model used is deep neural network. The data must be preprocessed into an array of values between 1 and 0 for input and output.

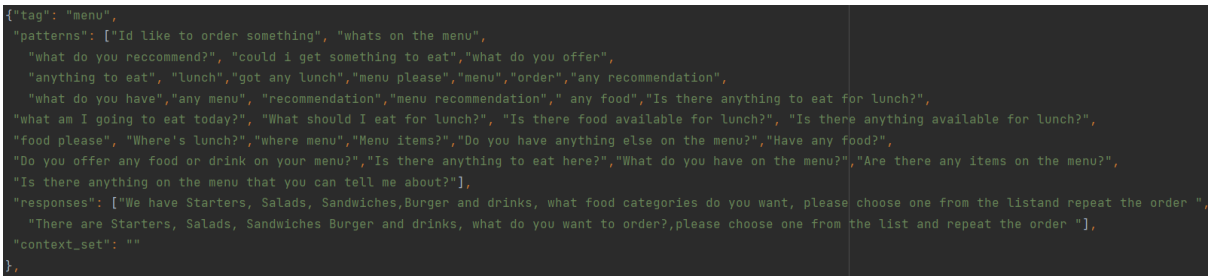
3.1 Data Collection

First of all, we will create a chatbot dataset in JSON format. The dataset contains 3 key elements. They are tag, patterns, and responses. The definition of the three key elements in the dataset is as follows:

Data elements	Definition
Tag	To group together instances of text that are related and then utilise those groups as targeted outputs when training neural networks.
Patterns	Patterns which are or may be expected as being submitted by end-users as inputs
Responses	Patterns that are or may be provided to end-users as outputs from the chatbot in their interactions with it

(Table 3.1.1: Definition for each data element)

Next, let us illustrate what the dataset will look like. One of the parts of the dataset is depicted in image 3.1.2 as below.



(Image 3.1.2: Part of the dataset)

3.2 Data preprocessing

3.2.1 Tokenization and Stemming

During this stage of the tokenization process, we perform the processes that convert pattern data elements into single words.

Stemming is a method of comparing words that have the same root term. It will be easier for us to reduce the size of our vocabulary if we connect similar terms with the words that give them their underlying meaning. The image 3.2.1 below shows the code of tokenization and stemming.

```
class doc_struct:
    def __init__(self, tag, words):
        self.tag = tag
        self.words = words

words = []
labels = [intent["tag"] for intent in intents]
labels = list(set(labels))
docs = []

for intent in intents:
    for pattern in intent['patterns']:
        wrds = nltk.word_tokenize(pattern)
        words.extend(wrds) # concatenate 2 list
        docs.append(doc_struct(intent["tag"], wrds))
```

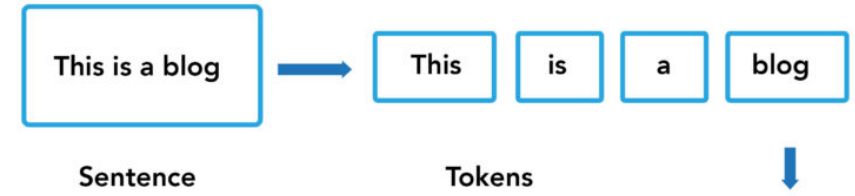
```
words = [stemmer.stem(w.lower()) for w in words if w != "?"] # stemmer: extract root word
words = sorted(list(set(words))) # remove duplicates
labels = sorted(labels)
```

```
'''
Representation: bag of words
for example, wrds= ['a', 'apple', 'eat', 'you', 'i']
sent = ['a', 'apple', 'i']
bag(sent, wrds) = [1, 1, 0, 0, 1]
whose 1 denotes existence of corresponded word, 0 is not.
'''
training = []
output = []
out_empty = [0 for _ in range(len(labels))]

for doc in docs:
    bag = []
    wrds = [stemmer.stem(w) for w in doc.words]
    for w in words:
        if w in wrds:
            bag.append(1)
        else:
            bag.append(0)
    output_row = out_empty[:]
    output_row[labels.index(doc.tag)] = 1

    training.append(bag)
    output.append(output_row)
```

(Image 3.2.1: Tokenization and Stemming)

3.2.2 Generate BOW (Bag of Words)

Bow is a technique that can convert words into numbers. It does this by constructing vector embeddings from the tokens that were created in the phase before it. This information is fed into the neural network model to help it understand what is being written. The image 3.2.2 illustrates the BOW process. In image 3.2.3, it shows the code for BOW.



Original word	BOW Vector			
This	1	0	0	0
is	0	1	0	0
a	0	0	1	0
blog	0	0	0	1

(image 3.2.2: BOW process illustration)

```
[ ]: def bag_of_words(sent, words):
        bag = [0 for _ in range(len(words))]
        sent_wrds = nltk.word_tokenize(sent)
        sent_wrds = [stemmer.stem(w.lower()) for w in sent_wrds]

        # counting freq for each word
        for word in sent_wrds:
            for i, w in enumerate(words):
                if w == word:
                    bag[i] = 1
        return np.array(bag)
```

(Image 3.2.3: code for BOW)

3.3 Tensorflow text classification algorithm

We will use the Tensorflow deep learning library (TFlearn) to build a text classification model. In this project, we will implement a deep neural network that is provided by Tensorflow. The image 3.3.1 shows the code of a deep neural network classification algorithm.

```
In [ ]: training = np.array(training)
        output = np.array(output)

In [ ]: # Building AI
        tensorflow.compat.v1.reset_default_graph() # this is required to run afterwards
        net = tflearn.input_data(shape=[None, len(training[0])])

        net = tflearn.fully_connected(net, 8) # 8 neurons
        net = tflearn.fully_connected(net, 8) # 8 neurons
        net = tflearn.fully_connected(net, len(output[0]), activation="softmax") # output layers
        net = tflearn.regression(net)

        model = tflearn.DNN(net)

In [ ]: model.fit(training, output, n_epoch=1000, batch_size = 8, show_metric = True)
        model.save("model.tflearn")
```

(Image 3.3.1: code for text classification model)

3.4 Evaluation of the model

Interaction with the chatbot serves as the basis for determining how well the model works. When testing the chatbot and interacting with it, one thing that will be taken into consideration is the question of whether or not it is possible to achieve the desired outcome by categorising the output.

3.5 Chatbot window

We borrowed a feature from the internet that lets a user talk to a bot through text. The function makes sure that the chat window stays open until it is told to close or break. Tim is the name of our text bot. The algorithm for this function is as in Image 3.5.1 as follows:

```
In [ ]: def chat():
            print("You're talking with the bot (type quit to quit)!")

            while True:
                user_inp = input("You: ")
                if user_inp.lower() == "quit":
                    break

                results = model.predict([bag_of_words(user_inp, words)])[0]
                results_idx = np.argmax(results)
                tag = labels[results_idx]
                if results[results_idx] > 0.7:

                    for corspd in dat['intents']:
                        if corspd['tag'] == tag:
                            responses = corspd['responses']
                    

                    print(rd.choice(responses))
                else:
                    print("I didn't get that, try again")
        chat()
```

(Image 3.5.1: code for Chatbot window)

4.0 Results and Discussion

In this section, we will discuss the features of the application for Chatbot for restaurants or cafes, as well as the guided user interface (GUI) of the program.

```
chat()
You're talking with the bot (type quit to quit)!
You: hi
Hello!
You:
  @media print {
    .ms-editor-squiggles-container {
      display:none !important;
    }
  }
  .ms-editor-squiggles-container {
    all: initial;
  }I would  like to order
We have Starters, Salads, Sandwiches,Burger and drinks, what food categories do you want, please choose one from the listand re
peat the order

You:
```

This is the first output that will be displayed once a user inputs a greeting such as "Hello", "Hi" and so on. Then the chat bot will reply back. Next the user will ask "i would like to order". The chat bot will reply with the "what's on the menu" and list all the courses available for the user to choose.

```
You: I would  like to order
There are Starters, Salads, Sandwiches Burger and drinks, what do you want to order?,please choose one from the list and repeat
the order
You: Starters please
We have Seafood Chowder - RM7.95, Mozzarella Sticks - RM4.95, Onion Rings - RM4.95 and Potato Fries - RM4.95
You:
  @media print {
    .ms-editor-squiggles-container {
      display:none !important;
    }
  }
  .ms-editor-squiggles-container {
    all: initial;
  }1 Potato Fries and 2 Onion Rings
order created

You:
```

```
You: salads
we have Greek Salad - RM19.95,Spinach Salad - RM 15.50, and chesszy salad - RM 12.90
You:
  @media print {
    .ms-editor-squiggles-container {
      display:none !important;
    }
  }
  .ms-editor-squiggles-container {
    all: initial;
  }i would like to have 1 chessy salad
order created

You:
```

Then the user will select the course they want. After that, the chatbot will display the menu within the courses, and subsequently the order will be created. The user can still choose

another course they want to add. As an example, if the user chooses to add salad, the chat bot will display all the menu items in the course. After that, the user will choose the food they wish to order. Finally, the order will be created.

```
You: give me the menu again
We have Starters, Salads, Sandwiches,Burger and drinks, what food categories do you want, please choose one from the listand re
peat the order
You: burger
we have Black and Blue - RM29.95,Tuna cheese  - RM 20.50, and Carribean Jerk  - RM 22.90
You: one black and blue please
your orders is created
You: drinks
cola - RM 5.00, orange juices - RM 10.90, special juices - RM 20.00, watermelon - RM10.30, water - RM3.00
You: one cola and 2 orange juices
Yes, done ordered

You:
```

The users can request for the chatbot to display the courses that the restaurant has. After that, the chat bot will repeat itself by displaying the courses. The users will choose the course that they feel like having. Repeating the same step, the chat bot will display the menu. The order will be created once the user chooses the food that they want. Next, the user wants to proceed with the drinks. The chat bot will list the drinks menu. The user will choose the drink and the order will be created.

```
You:
  @media print {
    .ms-editor-squiggles-container {
      display:none !important;
    }
  }
  .ms-editor-squiggles-container {
    all: initial;
  }operation hour?
We are open 7am-4pm Monday-Friday!
You: thank you very much
I didn't get that, try again
```

The chat bot will answer all questions regarding the restaurant. For example, in this case, the user asked for the restaurant's operating hours. The chat bot will display the answer. Lastly, if the question is not related to the restaurant, the chat bot will display the notice "i didn't get that, try again".

5.0 Conclusion

In conclusion, this system is beneficial to the customers as well as the owners of the restaurants. We are still getting used to the system, so there are some constraints on how the languages can be used at the moment. However, this can be improved in the future. The system would make a major contribution towards society, the environment, and systems. This system can produce more software developers who can improve the system. Our country can become a developed country with their assistance. For example, if their project is the first advanced project, they can entice an investor to invest in it. Besides, it can raise environmental awareness by disseminating information to many people. More people are responding to chatbots because of the use of natural language processing. Lastly, chatbots are a type of digital assistant that automates routine support tasks to improve business efficiency.

6.0 References

1. By Great Learning Team -. (2022, May 25). *Introduction to Chatbot | Artificial Intelligence Chatbot Tutorial*. GreatLearning Blog: Free Resources What Matters to Shape Your Career! https://www.mygreatlearning.com/blog/basics-of-building-an-artificial-intelligence-chatbot/

2. Tim, T. W. (2019, May 27). *Python Chat Bot Tutorial - Chatbot with Deep Learning (Part 1)* [Video]. YouTube. https://www.youtube.com/watch?v=wypVcNIH6D4&feature=youtu.be

3. Lee, G. (2022b, January 29). *Chatbots: AI tech can be useful in real-world business applications right now*. International Accounting Bulletin. https://www.internationalaccountingbulletin.com/extra-categories/editors-pick/a-chatbot-carol/#:%7E:text=Chatbots%20are%20one%20of%20the,expensive%20human%20interaction%20with%20customers.

4. Gupshup.io. (2022, February 16). *6 Advantages of Restaurant Chatbots*. Gupshup. https://www.gupshup.io/resources/blog/6-advantages-of-restaurant-chatbots

5. Damien, A. (2022). *TFLearn | TensorFlow Deep Learning Library*. TFLearn: Deep Learning Library Featuring a Higher-Level API for TensorFlow. http://tflearn.org/

6. Lyons, K. (2017). Intelligent Chatbot.

http://norma.ncirl.ie/2697/1/karllyons.pdf

Appendix

https://github.com/taimoon/AI-School-Project

```
import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()

import numpy as np
import tflearn
import tensorflow
import random as rd
import json

with open("intents_bot2.json") as f:
	dat = json.load(f)
	intents = dat['intents']

class doc_struct:
	def __init__(self, tag, words):
		self.tag = tag
		self.words = words

words = []
labels = [intent["tag"] for intent in intents]
labels = list(set(labels))
docs = []

for intent in intents:
	for pattern in intent['patterns']:
		wrds = nltk.word_tokenize(pattern)
		words.extend(wrds) # concatenate 2 list
		docs.append(doc_struct(intent["tag"], wrds))

words = [stemmer.stem(w.lower()) for w in words if w != "?"] # stemmer: extract root word
words = sorted(list(set(words))) # remove duplicates
labels = sorted(labels)

'''
Representation: bag of words
for example, wrds= ['a', 'apple', 'eat', 'you', 'i']
sent = ['a', 'apple', 'i']
bag(sent, wrds) = [1, 1, 0, 0, 1]
whose 1 denotes existence of corresponded word, 0 is not.
'''
training = []
output = []
out_empty = [0 for _ in range(len(labels))]
```

```
for doc in docs:
	bag = []
	wrds = [stemmer.stem(w) for w in doc.words]
	for w in words:
		if w in wrds:
			bag.append(1)
		else:
			bag.append(0)
	output_row = out_empty[:]
	output_row[labels.index(doc.tag)] = 1

	training.append(bag)
	output.append(output_row)

training = np.array(training)
output = np.array(output)

# Buidling AI
tensorflow.compat.v1.reset_default_graph() # this is required to run afterwards
net = tflearn.input_data(shape=[None, len(training[0])])

net = tflearn.fully_connected(net, 8) # 8 neurons
net = tflearn.fully_connected(net, 8) # 8 neurons
net = tflearn.fully_connected(net, len(output[0]), activation="softmax") # output layers
net = tflearn.regression(net)

model = tflearn.DNN(net)

model.fit(training, output, n_epoch=1000, batch_size = 8, show_metric = True)
model.save("model.tflearn")

def bag_of_words(sent, words):
	bag = [0 for _ in range(len(words))]
	sent_wrds = nltk.word_tokenize(sent)
	sent_wrds = [stemmer.stem(w.lower()) for w in sent_wrds]

	# counting freq for each word
	for word in sent_wrds:
		for i, w in enumerate(words):
			if w == word:
				bag[i] = 1
	return np.array(bag)

def chat():
	print("You're talking with the bot (type quit to quit)!")

	while True:
		user_inp = input("You: ")
		if user_inp.lower() == "quit":
```

```
                    break

                results = model.predict([bag_of_words(user_inp, words)])[0]
                results_idx = np.argmax(results)
                tag = labels[results_idx]
                if results[results_idx] > 0.7:

                    for corspd in dat['intents']:
                        if corspd['tag'] == tag:
                            responses = corspd['responses']

                    print(rd.choice(responses))
                else:
                    print("I didn't get that, try again")
chat()
```