TOPIC:INSIGHT INTO CHATBOTS

DONE BY-

1NH20IS118 PRATHIKSHA R K

1NH20IS125 RADHIKA AJITH

CHAPTER 1

INTRODUCTION

1.1 Introduction

We live in a world where automation of work is the new normal. Activities that once were required to be done manually and repeatedly now can be automated by the simple use of loops and pre-defined code. Chatbots are a wonderful example of such automation as we can improve customer service and reduce the time taken for the same.

1.2 Motivation of the project

The primary motivation for the project was derived from the experience of the students in ordering from fast-food restaurants which required order to happen in person or in-person calling. The ongoing pandemic has made this an issue not just of convenience but also of safety. Other than this, an inquisitive insight into the working of a chatbot was also the primary driving force for the project.

1.3 Problem Definition

To create, design, and implement a chatbot that can handle or automate customer inquiry and orders using python when implemented in a website.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing System

Existing systems use the following frameworks for natural language processing-

1. NLTK:

The NLTK library is a Python framework that is used for Natural Language Processing. It consists of various libraries that are designed to handle various aspects of natural language processing, such as text tokenization, stop world removal, and clustering.

2.spaCy:
Spacey is one of the powerful libraries that is used in Natural Language Processing. It is used in python as well as in python. SpaCy allows us to connect to statistical models like PyTorch, or TensorFlow. It allows us to do word vectors, dependency parsing, language models, and entity detection. spaCy is used to make chatbots, autocompletion, and autocorrection work.

Polyglot
Polyglot is an open-source library that is used for NLP. It works on NumPy which works on an array of words and therefore is fast at processing. It consists of features like sentiment analysis, named entity recognition, and part of speech tagging. Polyglot is similar to SpaCy and works on languages that are not supported by SpaCy.

Gensim
Gensim is a library used for NLP and topic modeling. It does not targe in-memory processing which makes it different from other libraries. We use Gensim to perform topic identification, corpora, and build word vectors or documents. Gensim consists of some libraries: fastText, word2vec, Latent Semantic Analysis, Latent Dirichlet Allocation. Without loading large data in memory it provides the output.

Pattern
It is easy to modify or manipulate numeric data but not text data, pattern is one of the packages used in python for text handling. This package is also used in machine learning and neural networking. Parse method is used for the operations like tokenization or POS. A module called en

must be imported from the pattern library to use the parse module. Using parsing we get the input string separated as noun, verb, object, subject, etc.

TextBlob

Many in-built libraries are coming up for the enhancement of natural language processing and TextBlob is one among them. It's usually used for sentiment analysis using the input data in review or Twitter. TextBlob works faster than the NLTK model in text processing and is easy to use.

Natural Language Processing (NLP) Examples

Email filters

Email filters are one of the most basic and initial applications of NLP online. The system recognizes if emails belong in one of three categories (primary, social, or promotions) based on their contents. For all Gmail users, this keeps your inbox to a manageable size with important, relevant emails you wish to review and respond to quickly.

Smart assistants

Smart assistants like Apple's Siri and Amazon's Alexa recognize patterns in speech thanks to voice recognition, then infer meaning and provide a useful response.We now expect assistants like Alexa and Siri to understand contextual clues as they improve our lives and make certain activities easier like ordering items, and even appreciate when they respond humorously or answer questions about themselves.

Predictive text

Things like autocorrect, autocomplete, and predictive text are so commonplace on our smartphones that we take them for granted. Autocomplete and predictive text are similar to search engines in that they predict things to say based on what you type, finishing the word or suggesting a relevant one.

2.2 Proposed System

The model we have used is NLTK which also works with a neural network which is a concept in data science that works to emulate the workings of a human brain
Additionally, this model is rule-based and directed to serve the purpose of a commercial website.

2.3 Objectives of the Proposed System

The proposed system is made to understand the underlying working of the chatbot and also automize the 'order taking' and "FAQ" functionality.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Hardware Requirements

The following are needed to efficiently use the application.

Processor	-	Intel Core i3 and above
Speed	-	2.21 GHz
RAM	-	4 GB (min)
Hard Disk	-	50 GB

3.2 Software Requirements

Software requirements define software resource fundamentals that need to be installed on a workstation to provide optimum working of a software. The following are required for optimal development and usage of the application.

Operating System	Windows 7 and above
Programming Language	Html,CSS , Python 3.7
Compiler	Anaconda(knowledge on creating virtual environments on anaconda), Spyder
Implementation is done on	Jupyter Notebook
Additional libraries used	Tensorflow(deep- learning open-sourced lib) NLTK (natural language toolkit) Keras model(used for creating easy python neural networks)

Additional files to be created-

intents.json file-

- aimed at helping the machine create patterns and combinations and try to detect them from the user's input.
- Contains patterns (expected input combinations) and responses (responses from the bot). Refer Fig 5.2: intents.json

Orders.txt-

- Hold the user inputted choice according to identification based on an index.

Custom HTML/CSS package

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>best cafe in India |mcafe.com</title>
    <link rel="stylesheet" href="css/style.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Baloo+2&display=swap" rel="stylesheet">
</head>
<body>
    <nav id="navbar">
        <div id="logo">
            <img src="logo.jpg" alt="M CAFE">
        </div>
        <ul>
            <li class="items"><a>Home</a></li>
            <li class="items"><a>Services</a></li>
            <li class="items"><a>About Us</a></li>
            <li class="items"><a>Contact Us</a></li>

        </ul>
    </nav>
    <section id="home">
        <h1 class="h-primary">WELCOME TO M CAFE</h1>
        <p>COFFEE THAT TASTES BETTER!!</p>
        <p>NEVER LIKE BEFORE ENJOY! START YOUR DAY WITH US!</p>
        <pre> <button class="btn"> <a href="/my-link/">chat with us</a></button></button></pre>
```

```css
/* css reset */
*{
    margin: 0;
    padding: 0;
}
/* css variabl  */
:root{
    --navbar-height:59px;
}
/* navigation bar */
#navbar{
    display:flex;
    align-items: center;

}
#navbar::before{
    content: "";
    background-color: black;
    position: absolute;
    height: 100%;
    width: 100%;
    z-index: -1;
    opacity:0.6;
}
/* nav bar: logo and image  */
#logo{
        margin: 5px 30px;
    ;
```

Fig 3.1.:HTML/CSS code

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture

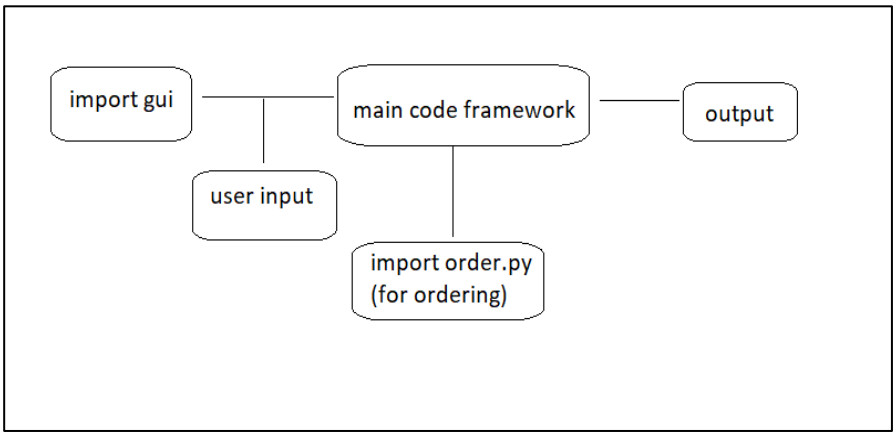

Fig 4.1.a: Chatbot A architecture

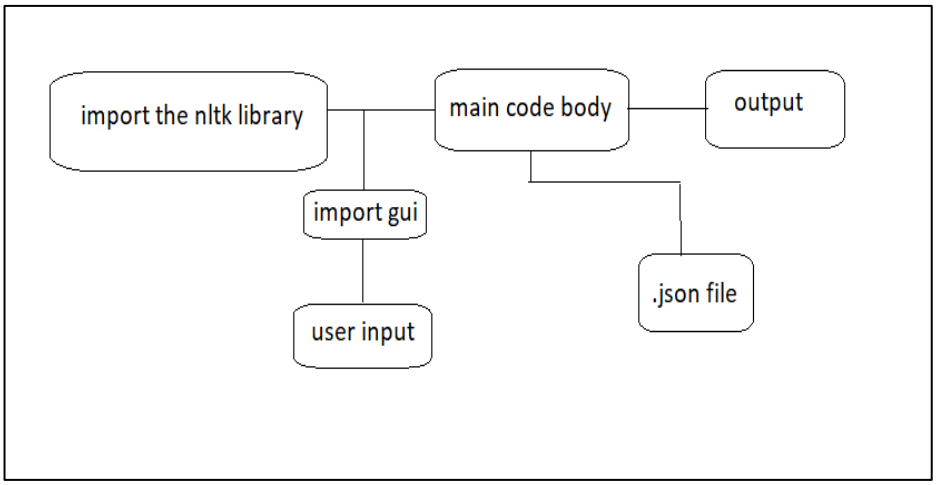

Fig 4.1.b.: Chatbot B architecture

4.2 Flowchart of Proposed System

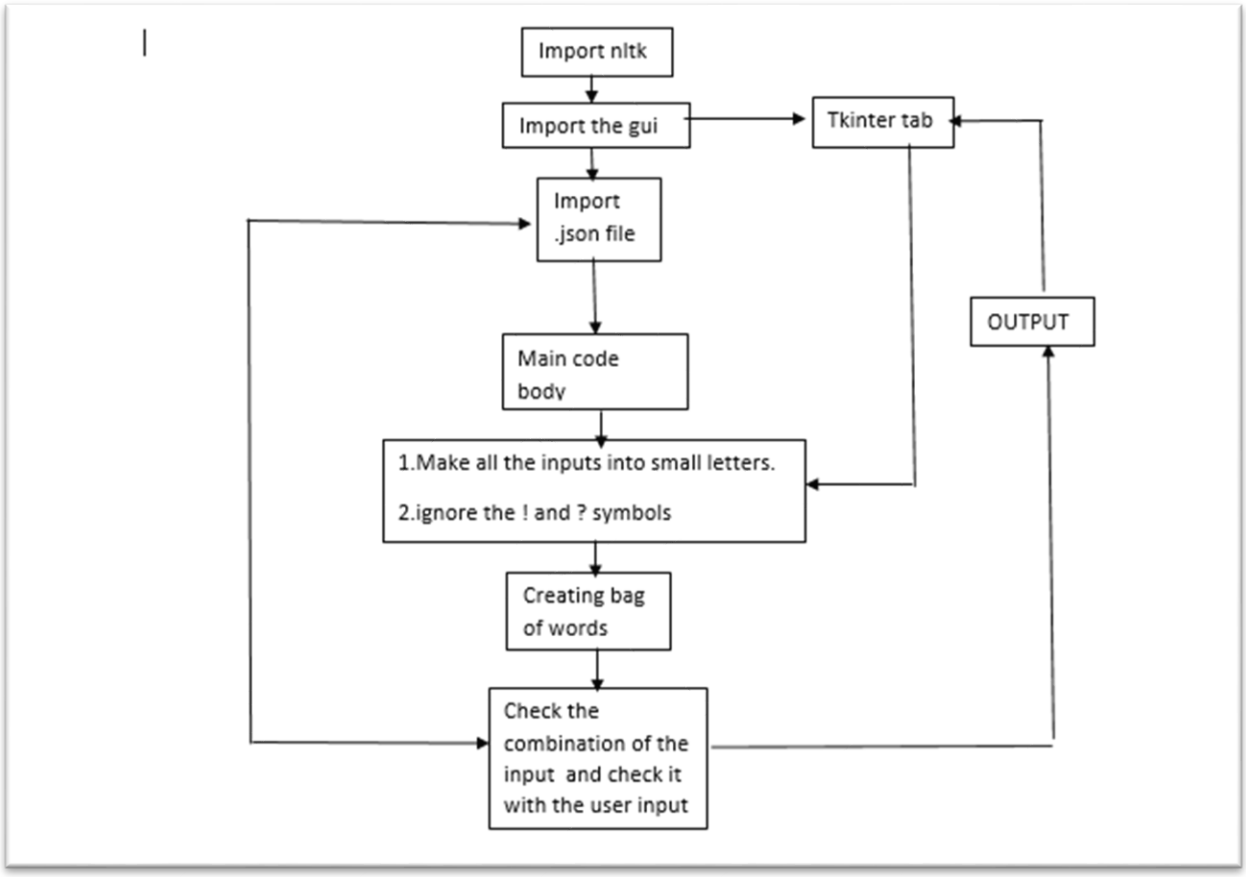

Fig 4.2: Flowchart of the proposed system

ALGORITHM

Step 1: Import all the required libraries (nltk, pickle, TensorFlow, NumPy, random, JSON)

Step 2: Initialising the variables as an empty list

Step 3: Corpus is imported and read from JSON file

Step 4: Using iteration the input sentence is tokenized and appended to the word list

Step 5: In the same iterative function tag of the words are appended to the class list

Step 6: Lemmatization of each token is done and also converted into lower case words and duplicate words are removed from the list of words

Step 7: Words are then sorted and stored in words

Step 8: Classes is also sorted

Step 9: Words and classes lists are dumped in a pickle file

Step 10: Empty list is created for training the data

Step 11: Bag of words list is initialized

Step 12: Lemmatization of each word is done and a base word is created to represent related words

Step 13: For loop is used to find if word match is found in current pattern and is appended to a bag of words with array 1 if found

Step 14: The training data is shuffled using random function and converted into np.array

Step 15: Create train and text list on pattern and intents

Step 16: Neural network model is created by determining parameters and deep learning model

Step 17: The model is compiled, fitted, and saved

Step 18: Another file is created and all the required models are imported

Step 19: Intents, words, and classes are loaded from JSON and pickle files

Step 20: Tokenisation is done and related random response is found from the list of responses

Step 21: Trained model is loaded

Step 22: Tkinter's model is imported for the user UI

Step 23: Input is taken from the user and the response is displayed on the UI of the bot

CHAPTER 5

IMPLEMENTATION

5.1 Code Implementation

Core keywords used in the code-

- Nltk
- .json
- Pickle
- Numpy
- Keras model in TensorFlow
- Random

1. NLTK- it is a natural language processing toolkit that is used in human language to computer language conversion. Within NLTK we use the following-

- Tokenization-normal input strings were converted to a list of tokens that is words needed to be obtained.
- Upper and Lowercases conversion so that the computer can recognize that the same words in different cases have the same meaning.
- Nltk.stem- "Stemming" is the process of finding the core keyword out of any word in its adjective, noun, or verb form

E.g- Ordering and Ordered → order
Beauty, Beautiful, Beautification→ Beauty

- Lemmanisation- above mentioned stemming may lead to the formation of words that aren't present in the English dictionary(i.e creation of non-existent words). Hence multiple words are grouped in a collection called a lemma.

E.g-"Better" and "Good" are present in the same lemma hence have the same meaning.

- "Bag of Words"- the "bag of words" is a method of representation that shows the occurrence of the inputted words in the corpus document (corpus is the additional data set that we have created according to the needs of the e-merchant)

E.g: If the input is "when can I get the order?"

bag of the word - {'when',' not',' can',' I',' delivery',' get',' the',' order'}

machine language -{ 1 , 0 , 1 , 1 , 0 , 1 , 1 , 1 }

here, in the bag, the word occurrence matters not the position of the word.

```
26 documents
7 classes ['delivery', 'goodbye', 'greeting', 'items', 'menu', 'payments', 'thanks']
54 unique lemmatized words ["'s", 'a', 'accept', 'anyone', 'are', 'bye', 'can', 'card', 'cash', 'credit', 'day', 'delivery',
'do', 'doe', 'for', 'get', 'good', 'goodbye', 'have', 'hello', 'helpful', 'hey', 'hi', 'how', 'i', 'is', 'item', 'kind', 'lat
er', 'long', 'lot', 'mastercard', 'my', 'of', 'only', 'pay', 'paypal', 'see', 'sell', 'shipping', 'special', 'take', 'thank',
'thanks', 'that', 'the', 'there', 'today', 'what', 'whats', 'when', 'which', 'with', 'you']
Training data created
Epoch 1/200

C:\Users\ADMIN\AppData\Local\Temp/ipykernel_13460/2566939595.py:73: VisibleDeprecationWarning: Creating an ndarray from ragge
d nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated.
If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
  training = np.array(training)

6/6 [==============================] - 0s 5ms/step - loss: 1.9801 - accuracy: 0.0769
Epoch 2/200
6/6 [==============================] - 0s 4ms/step - loss: 1.9504 - accuracy: 0.2692
Epoch 3/200
6/6 [==============================] - 0s 3ms/step - loss: 1.9127 - accuracy: 0.2692
Epoch 4/200
```

Fig 5.1: Creation and display of the bag of words

2.intents.json file- the corpus/data set created that is customized by the e- merchant

```
{
  "tag": "items",
  "patterns": [
    "Which items do you have?",
    "What kinds of items are there?",
    "What do you sell?"
  ],
  "responses": [
    "We sell coffee,tea and other confectionary",
    "We have coffee,tea and some delicious snacks!"
  ]
},
{
  "tag": "payments",
  "patterns": [
    "Do you take credit cards?",
    "Do you accept Mastercard?",
    "Can I pay with Paypal?",
    "Are you cash only?"
  ],
  "responses": [
    "We accept VISA, Mastercard and Paypal",
    "We accept most major credit cards, and Paypal"
  ]
},
```

```
  "patterns": [
    "How long does delivery take?",
    "How long does shipping take?",
    "When do I get my delivery?"
  ],
  "responses": [
    "Delivery takes 2-4 hrs dependng on the venue",
    "Shipping takes 2-4 hrs depending on the venue"
  ]
},
{
  "tag": "menu",
  "patterns": [
    "what is  the special for today?",
    "what can i get today?",
    "whats the special for today?"
  ],
  "responses": [
    "You could try our CHERRY AMARETTO DONUT ",
    "You could try our DOLCE DU LECHE",
    "You could try our CREAM CHEESE DONUT",
    "You could try our LEMONCELLO DONUT ",
    "You could try our HAZELNUT NUTELLA DONUT ",
    "You could try our MATCHA DONUT ",
    "You could try our CHOCOLATE GANACHE DONUT  ",
    "You could try our OREO DONUT ",
    "You could try our IRISH CREAM DONUT ",
```

Fig 5.2: intents.json

Here, there are patterns and corresponding responses that the machine must produce. The above-mentioned “bag of words” and the corresponding Fig 5.1 are created with this data.

3.NuMpy-this is used for working with mathematical operations on arrays. This helps work with enormous libraries of data with high-level mathematical functions.

4. Pickle- Pickling is the process of serializing or deserializing python object structures. It helps when working with databases and also helps maintain multiple sessions.

5. Keras model and TensorFlow- Tensorflow is a platform for machine learning and helps create neural networks. Neural networks as the word suggests are a subset of artificial intelligence and machine learning that emulates neural working in humans (done to a lesser extent).

We created multiple layers or “nodes” that are activated by the input have a cascading effect in activating nodes in subsequent layers. Hence we can obtain a rational output.

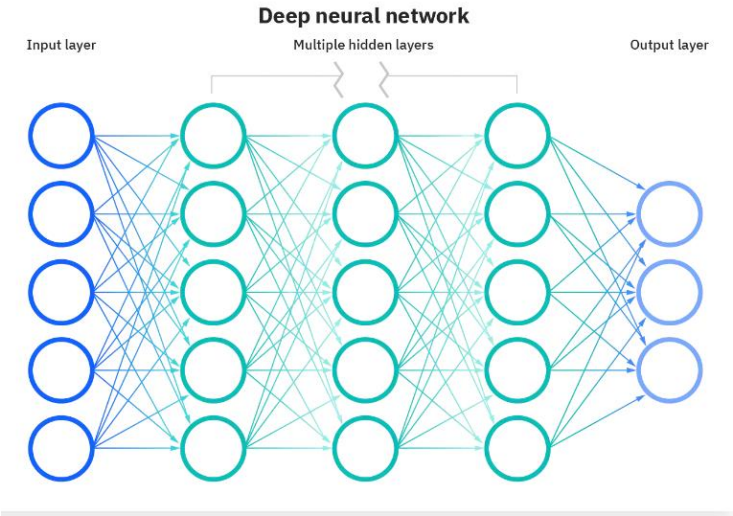

Fig 5.3:Representation of neural network layers

We create a 3 layer neural model with the first layer having 128 neurons, the second having 64, and (output) the third layer having as many as the number of intents in the ‘intents.json’ file.

```
# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer contains number of neurons
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save('chatbot_model.h5', hist)
```

Fig 5.4:Model created in code

PUNKT-

- Punkt is used to remove noise i.e anything inputted that isn't a number or a string.
- It is also used to remove "stop words" or parts of the input that don't have any effect on the meaning of the entire sentence

CODE – CHATBOT BASED ON NLTK LIBRARY- ENQUIRY CHATBOT

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import SGD
import random

words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('intents1.json').read()
intents = json.loads(data_file)


for intent in intents['intents']:
    for pattern in intent['patterns']:

        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# lemmaztize and lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)


pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))

# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
```

```
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # lemmatize each word - create base word, in attempt to represent related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    # create our bag of words array with 1, if word match found in current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag (for each pattern)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])
# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")
```

```
# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save('chatbot_model.h5', hist)

print("model created")
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np

from keras.models import load_model
model = load_model('chatbot_model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))


def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words

# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))

def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
```

```
def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result

def chatbot_response(msg):
    ints = predict_class(msg, model)
    res = getResponse(ints, intents)
    return res


#Creating GUI with tkinter
import tkinter
from tkinter import *


def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))

        res = chatbot_response(msg)
        ChatLog.insert(END, "Bot: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)


base = Tk()
base.title("Hello")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)

#Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="8", width="50", font="Arial",)

ChatLog.config(state=DISABLED)

#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set

#Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="12", height=5,
                    bd=0, bg="#32de97", activebackground="#3c9d9b",fg='#ffffff',
```

CODE-MENU CHATBOT

As evident, in this framework, the machine depended on simple if-else conditions to make corresponding output. Hence it has the demerits such as the word “MENU” and “Menu” and “menu” are identified as different words.

```
from tkinter import *
from orders import *
from PIL import ImageTk, Image
root = Tk()
def send():
    send = "You:"+ e.get()
    text.insert(END,"\n" + send)
    if(e.get()=='hi'):
        text.insert(END, "\n" + """Bot: hello how can i help you today
                                    ~~~~~~MENU~~~~~~
                                type the required keyword based on need
                                a.Menu
                                b.Place order
                                c.Enquiries and questions""")
    elif(e.get()=='hello'):

        text.insert(END, "\n" + """Bot: hi,how can i help you?
                                    ~~~~~~MENU~~~~~~
                                type the required keyword based on need
                                1.Menu
                                2.Place order
                                3.Enquiries and questions""")
    elif (e.get() == 'menu' or 'Menu' or'MENU'):
        text.insert(END, "\n" + """
1.CHERRY AMARETTO DONUT
Bright pink with a cherry amaretto glaze. This sweet treat is sure to please
with a dollop of butter cream on top.
```

```
2. DOLCE DU LECHE
Sweet, milky caramel covered in toasted almonds drizzled with a classic glaze.

3.CREAM CHEESE DONUT
Filled with cream cheese and covered with red velvet cake crumbs, this donut
highlights the best part of a red velvet cake, the cream cheese frosting.


4.LEMONCELLO DONUT
Filled with a sweet and tangy Lemoncello buttercream, this donut definitely
has a sweet surprise inside.


5.HAZELNUT NUTELLA DONUT
Topped with a thin layer of Nutella and covered with crispy cereal pieces,
this donut introduces a new way to eat your favorite chocolate spread.


6.MATCHA DONUT
This Japanese inspired donut has a matcha green tea based glaze on top of
a classic donut ring.


7.CHOCOLATE GANACHE DONUT
A chocolate glazed donut taken to the next level. Our version is glazed
with Chocolate Ganache for a rich, velvety taste.


8.OREO DONUT
```

```
8.OREO DONUT
Soft and Crunchy, this donut is the perfect balance between soft fried
 dough and crunchy Oreos.


9.IRISH CREAM DONUT
Filled with a Baileys Irish Cream Infused Donut, this adult version of
 the classic cream filled donut is the perfect sweet treat.


10.PEANUT DONUT
Peanut butter and chocolate come together to form this sweet and crunchy
 donut.


11.MAPLE BACON DONUT
A perfect blend of sweet maple syrup and and savory bacon. This flavor
combination is sure to satisfy all your taste buds.


12.CHOCOLATE HOKKAIDO CUPCAKE
This  soft and pillowly Japanese cupcake is filled with a dallop of
Chocolate buttercream.


13.VANILLA HOKKAIDO CUPCAKE
This  soft and pillowly Japanese cupcake is filled with a dollop of vanilla
buttercream.
```

```
""")
        a=e.get()
        a=None
        text.insert(END, "\n" + "")
        choice=e.get()
        print(choice)
        e.delete(0,END)
        text.insert(END, "\n" + "enter choice")
        quantity=e.get()
        print(quantity)
        menu(choice,quantity)
    if(e.get()=='questions'):
        %run "C:\Users\ADMIN\Desktop\mini project3rd sem\mini\chatbot.ipynb"

text = Text(root,bg='light blue')
text.grid(row=0,column=0,columnspan=2)
e = Entry(root,width=80)
send = Button(root,text='Send',bg='blue',width=20,command=send).grid(row=1,column=1)
e.grid(row=1,column=0)
root.title('Say Hi!')
root.geometry("650x500")
root.mainloop()
```

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 Outcome of Proposed System

Add new products to the store database:

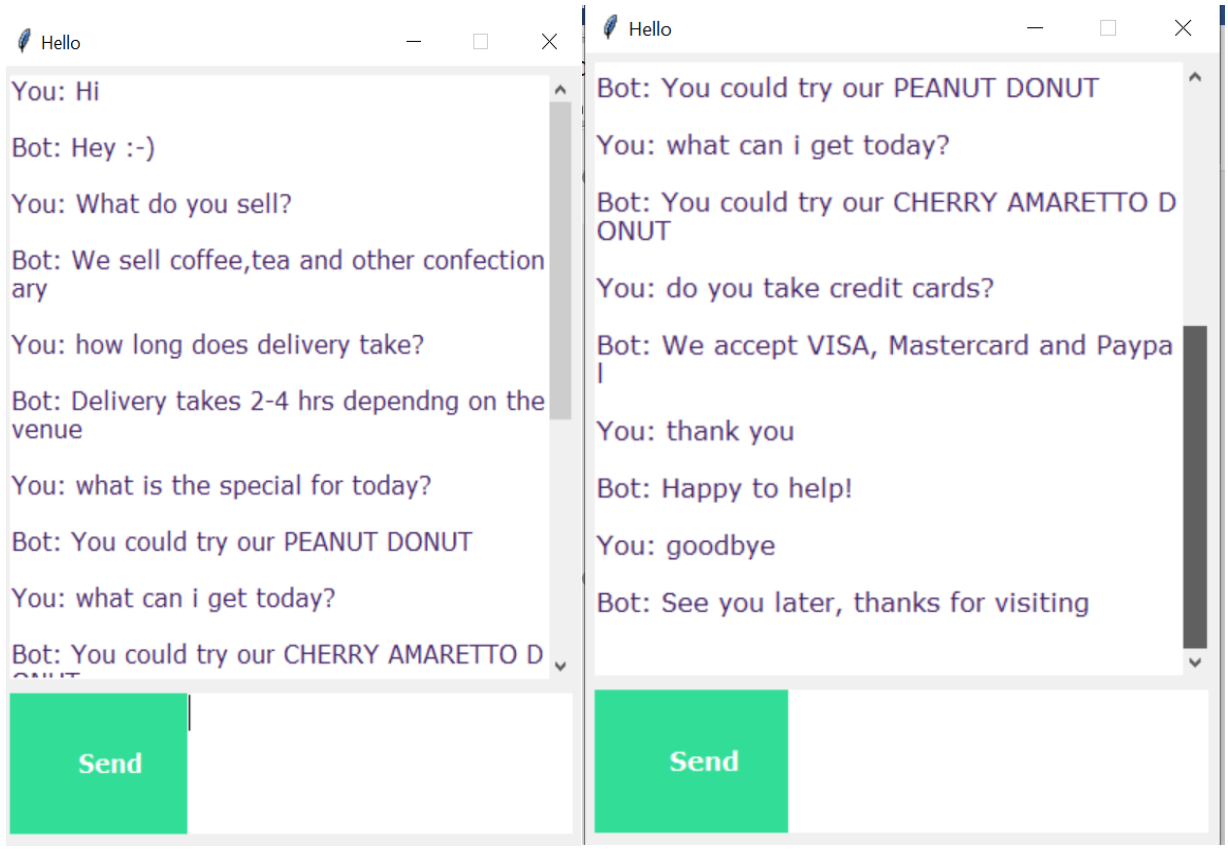

Fig 6.1: Enquiry chatbot

We can ask questions regarding the delivery, payment options and even ask for suggestions on the item to choose from.

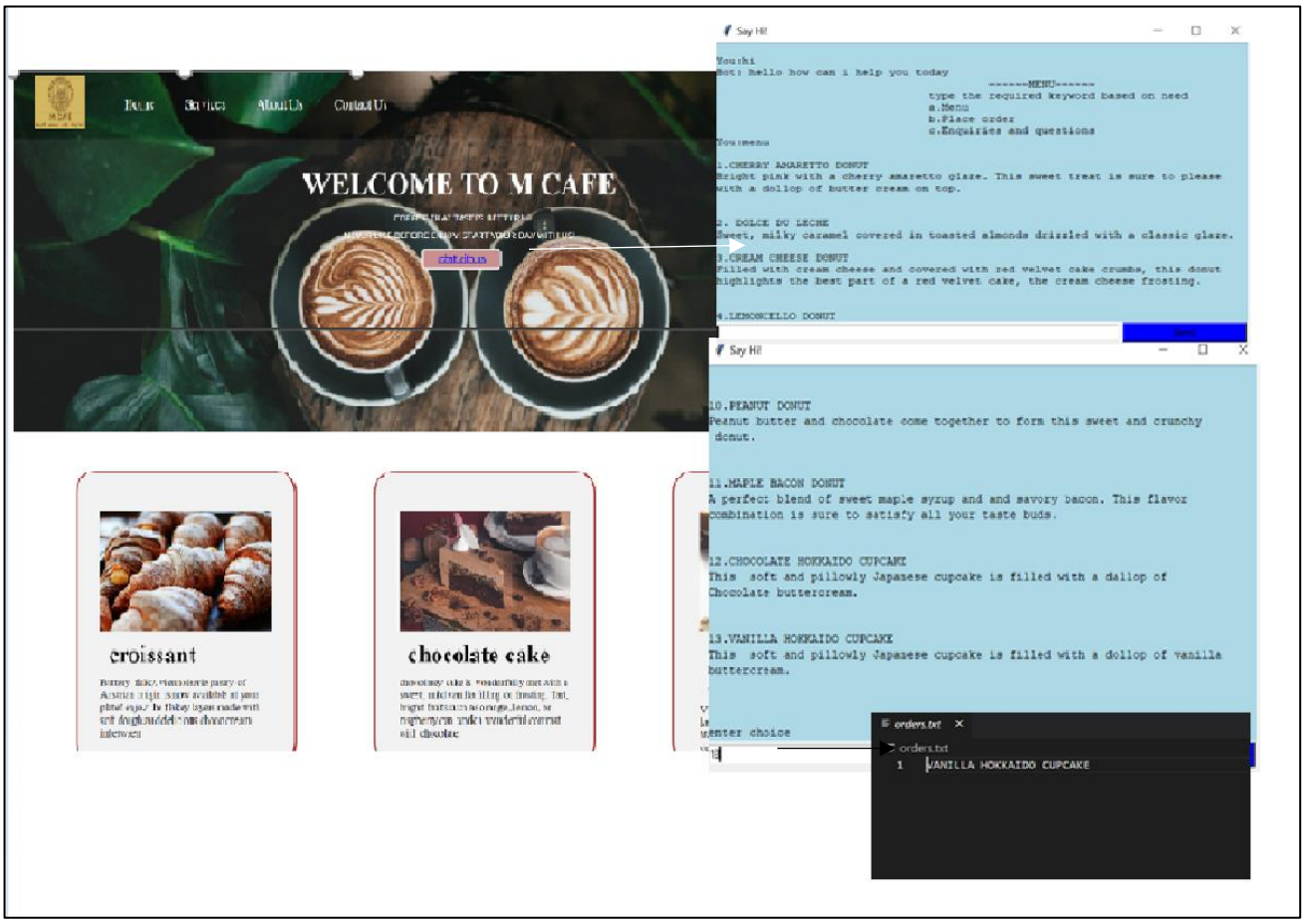

Fig 6.2: Implementation of menu chatbot

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion

This project aims at making the process of customer service as well as obtaining orders automated.

7.2 Future Enhancement

The model used takes time to run and hence having a better software architecture for the same would make the process more efficient. A very detailed .json file can extend the usage of the enquiry chatbot to be more than just the displayed purpose.