CHATBOT FOR PLACEMENT CELL

END TERM REPORT

By

Rivesh Raja, Deepak Kumar, Riya Roy, Swati

(Section: K18JF)

(Roll Number(s):18,23,06,48)

Transforming Education Transforming India

Department of Intelligent Systems
School of Computer Science Engineering
Lovely Professional University, Jalandhar
APRIL-2020

Student Declaration

This is to declare that this report has been written by me/us. No part of the report is copied from sources. All information included from other sources have been duly acknowledged. I/We aver that if any of the part of the report is found to be copied, I/we are shall take full responsibility for it.

Signature of Student

RIVESH RAJA
Roll number: 18

Signature of Student

DEEPAK KUMAR
Roll number: 23

Signature of Student

RIYA ROY
Roll number: 06

Signature of Student

SWATI
Roll number: 48

TABLE OF CONTENTS

BONAFIDE CERTIFICATE

Certified that this project report “ CHATBOT FOR PLACEMENT CELL” is the bonafide work of “ Rivesh Raja, Deepak Kumar, Riya Roy, Swati” who carried out the project work under my supervision.

Signature of the Supervisor

Name of supervisor

Academic Designation

ID of Supervisor

Department of Supervisor

1. INTRODUCTION

Chatbot can be defined as AI based computer program that simulates human conversations. They are also known as digital assistants that understand human capabilities. Bots interpret and process the user requests and give prompt relevant answers. Bots can through voice as well as text and can be deployed across websites, applications and messaging channels such as Facebook Messenger, Twitter or WhatsApp.

Chatbots work by analysing and identifying the intent of the user's request to extract relevant entities, which is the most important task of a chatbot. Once the analysis is done appropriate response is delivered to the user. The chatbots work by adopting three classification methods.

2. CLASSIFICATIONS

Pattern matching

Bots utilize pattern matches to group the text and it produces an appropriate response from the clients. Artificial Intelligence Markup Language (AIML) is a standard structured model of these patterns. A bot is able to get the right answer in the related pattern. The bots react to anything relating it to the correlate patterns.

Natural language understanding (NLU)

NLU is the ability of the chatbot to understand a human. It is the process of converting text into structured data for a machine to understand. NLU follows three specific concepts. They are: entities, context, and expectations.

3. GOALS AND OBJECTIVE

Chatbots boost operational efficiency and bring cost savings to businesses while offering convenience and added services for customers. They allow companies to easily resolve many types of customer queries and issues while reducing the need for human interaction.

According to Forbes, 80% of marketers plan to start using a chatbot in some way or another by 2020. This is a significant reason why brands are investing in improving the customer experience.

Let's find out the importance of adopting the chatbot strategy in your business and how chatbot benefits to win more customers or retain the existing ones.

Reduce customer waiting time – According to Chatbot Report, 21% of consumers see chatbots as the easiest way to contact a business. Chatbots are a smarter way to ensure that customers receive the immediate response that they are looking for without making them wait in a queue.

24×7 availability – 68% of customers switch to a competitor if they don't think you care about them. Bots are always available to engage customers with immediate answers to the common questions asked by them. The top potential benefit of using chatbots is 24-hour customer service.

Better customer engagement – Conversational bots can engage customers round the clock by starting proactive conservation and offering personalized recommendations that boost customer experience.

Easy scalability with bots – Bots can be easily scalable during the peak business hours or and manage 'n' number of customer conversations without additional customer service costs.

Save customer service costs – Juniper Research estimates that chatbots will help businesses save more than $8 billion per year by 2022. Chatbots help businesses to save customer service costs of hiring more support agents that require additional costs such as salary, training and infrastructure costs.

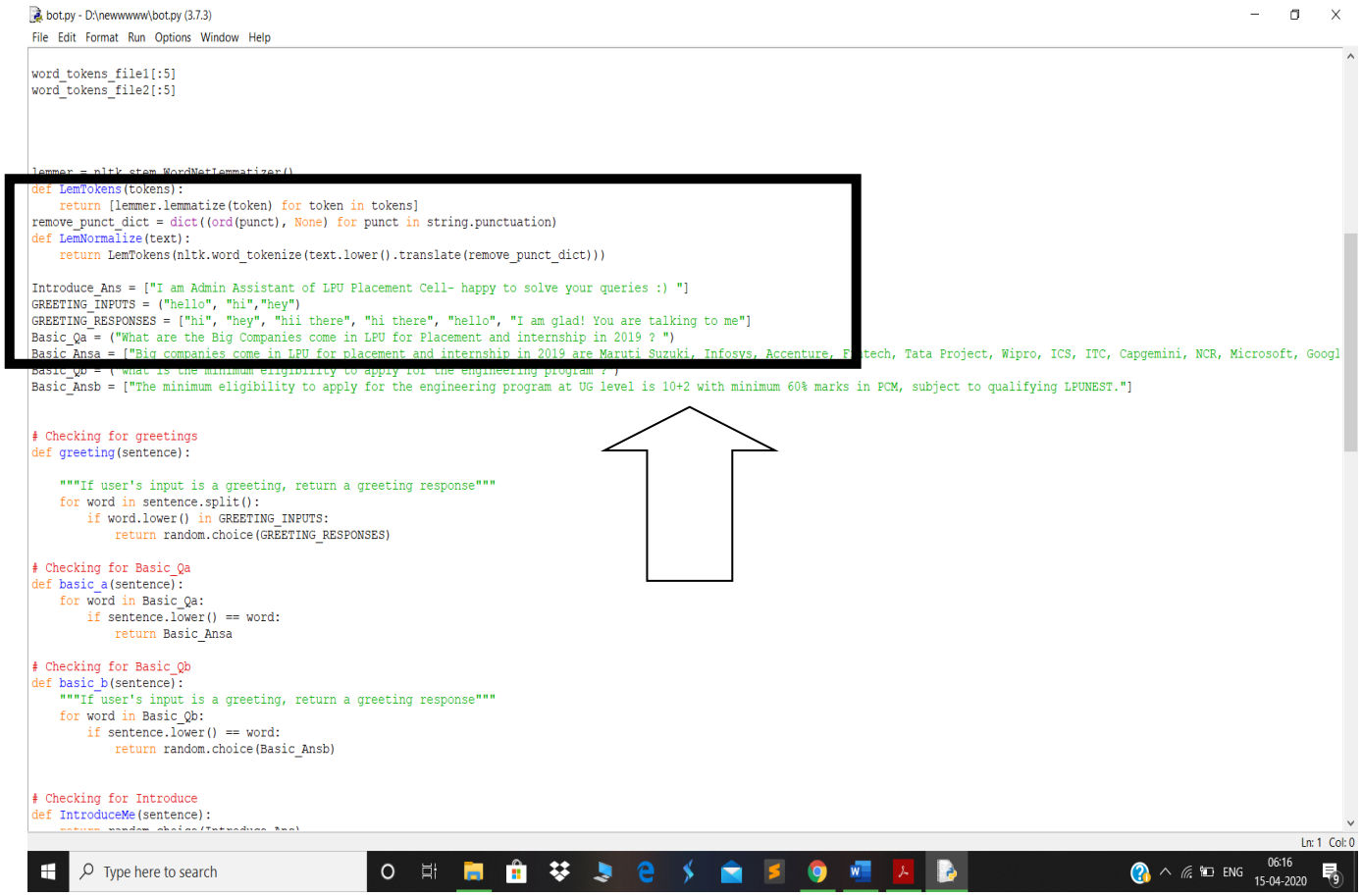
Automate lead qualification & sales – You can automate your sales funnel with chatbots to prequalify leads and directing them to the right team for further nurturing. Being able to engage customers instantly increases the number of leads and conversion rates.

Reduce customer churn rate– Engaging your visitors is arguably the single most sure-fire way of reducing bounce rates and subsequently increasing conversions. With chatbots, you can boost your engagement strategy even further and actually keep visitors hooked.

4. DESCRIPTION OF PROJECT

4.1. TRAINING THE DATA or DATA SET

The dataset is a JSON file like that contains the patterns we need to find and the responses we want to return to the user. In this program we tell train the data about the what to response again n again by using [intents.json] file it will remember the pattern used further to understand and find the pattern to response further.



4.2. PREREQUISITES

The project requires you to have the good knowledge of Python, Natural Language Processing (NLTK), Tkinter, JSON String, Random, NUMPY, Tokenizer, Lemmatize, etc. We will use some helping modules which you can download using the -PIP command in python.

A) NLTK

NLTK stands for Natural Language Toolkit. This toolkit is one of the most powerful NLP libraries which contains packages to make machines understand human language and reply to it with an appropriate response. Tokenization, Stemming, Lemmatization, Punctuation, Character count, word count are some of these packages in this library.

Code used is

```
import nltk

from nltk.stem.lancaster import LancasterStemmer

stemmer = LancasterStemmer()
```

B) NUMPY:

Numpy is a library for the Python programming language, adding

support for large, multi-dimensional arrays and matrices, along with

a large collection of high-level mathematical functions to operate on

these arrays.

Code used is

```
import numpy as np
```

C) TENSORFLOW

TensorFlow is a rich system for managing all aspects of a machine

learning system; however, this class focuses on using a particular

TensorFlow API to develop and train machine learning models.

TensorFlow APIs are arranged hierarchically, with the high-level APIs

built on the low-level APIs. Machine learning researchers use the lowlevel

APIs to create and explore new machine learning algorithms.

Code used is

import tensorflow as tf

D) Random:

In Python, a random module implements pseudo-random number

generators for various distributions including integer, float (real).This

function of the module is used in predicting the output

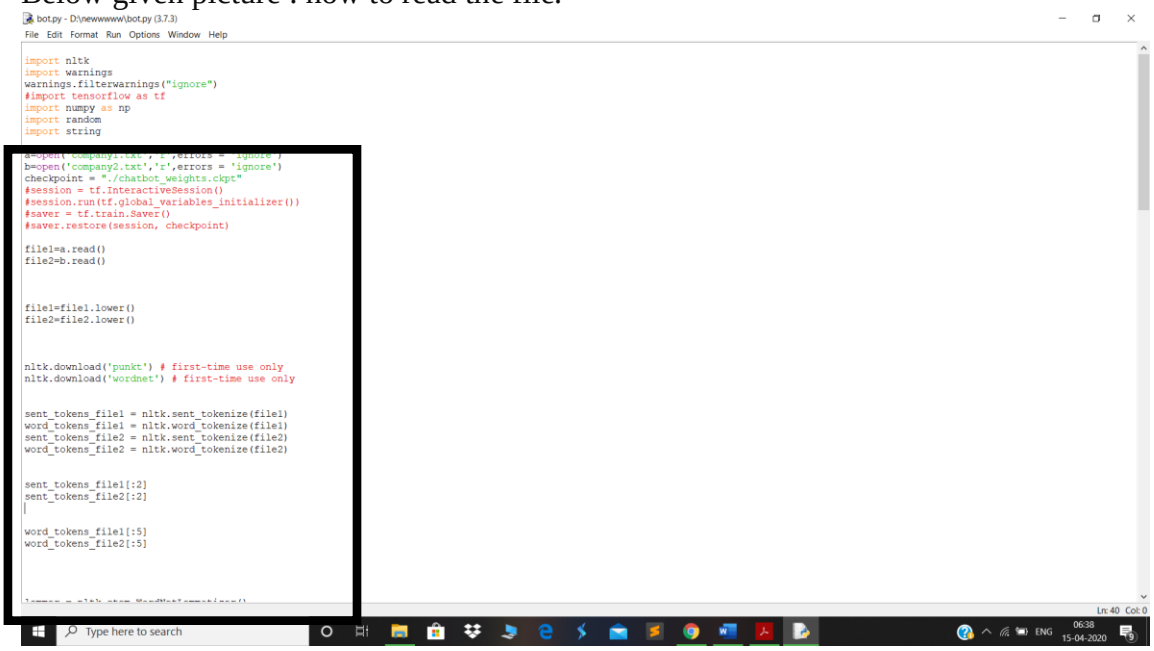
Code used is

import random

4.3. IMPORT AND LOAD THE DATA

First, make a file name as BOT.py. We import the necessary packages for our chatbot and initialize the variables we will use in our Python project. As I have made two files of name COMPANY 1.txt and COMPANY 2.txt . First we read both the files and load the file to break there statements in sentence and word.

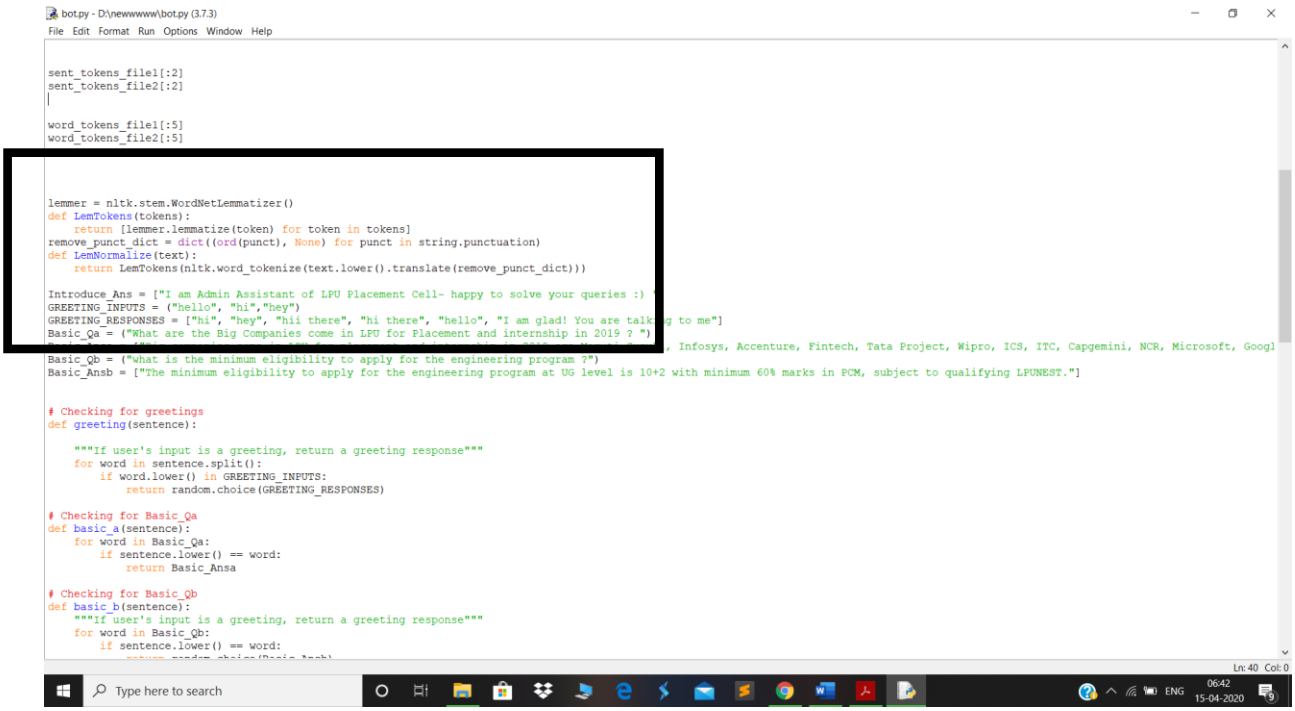
Below given picture : how to read the file.



4.4. PREPROCESS DATA

When working with text data, we need to perform various preprocessing on the data before we make a machine learning or a deep learning model. Tokenizing is the most basic and first thing you can do on text data. Tokenizing is the process of breaking the whole text into small parts like words.

Here we iterate through the patterns and tokenize the sentence using nltk.word_tokenize() function and append each word in the words list. We also create a list of classes for our tags.



Now we will lemmatize each word and remove duplicate words from the list. Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file to store the Python objects which we will use while predicting.

bot.py - D:\newwwww\bot.py (3.7.3)

```
    for word in Basic_Qb:
        if sentence.lower() == word:
            return random.choice(Basic_Ansb)


# Checking for Introduce
def IntroduceMe(sentence):
    return random.choice(Introduce_Ans)


from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity


# Generating response
def response_a(user_response):
    robo_response=''
    sent_tokens_file1.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens_file1)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I don't have answer for that. Is there something else I can help"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens_file1[idx]
        return robo_response


def response_b(user_response):
    robo_response=''
    sent_tokens_file2.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens_file2)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I don't have answer for that. Is there something else I can help"
        return robo_response
    else:
```

4.5. PREDICT THE RESPONSE (Graphical User Interface)

Now to predict the sentences and get a response from the user to let us create a new file ‘bot.py’.

We will load the trained model and then use a graphical user interface that will predict the response from the bot. The model will only tell us the class it belongs to, so we will implement some functions which will identify the class and then retrieve us a random response from the list of responses.

gui.py - D:\newwwww\gui.py (3.7.3)

```
from tkinter import *
import time
import tkinter.messagebox
from bot import chat
import pyttsx3
import threading

saved_username = ["You"]
#ans=["ADMIN"]
window_size="400x400"

class ChatInterface(Frame):

    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master

        # sets default bg for top level windows
        self.tl_bg = "#EEEEEE"
        self.tl_bg2 = "#EEEEEE"
        self.tl_fg = "#000000"
        self.font = "Verdana 10"

        menu = Menu(self.master)
        self.master.config(menu=menu, bd=5)
# Menu bar

    # File
        file = Menu(menu, tearoff=0)
        menu.add_cascade(label="File", menu=file)
      # file.add_command(label="Save Chat Log", command=self.save_chat)
        file.add_command(label="Clear Chat", command=self.clear_chat)
     #  file.add_separator()
        file.add_command(label="Exit",command=self.chatexit)

    # Options
        options = Menu(menu, tearoff=0)
        menu.add_cascade(label="Options", menu=options)

        # username



        # font
        font = Menu(options, tearoff=0)
        options.add_cascade(label="Font", menu=font)
```

Using GUI application we created an interface so that the person can come and interact with us in this GUI we have made a window box with various option is the tab to exit the application window to clear the chats. About the developers. You can also change the colour of the text .

We have use button keyword to submit the response as tkinter message box, a scrolling window etc.

To predict the class, we will need to provide input in the same way as we did while training. So we will create some functions that will perform text pre-processing and then predict the class.

Now we will code a graphical user interface. For this, we use the Tkinter library which already comes in python. We will take the input message from the user and then use the helper functions we have created to get the response from the bot and display it on the GUI. Here is the full source code for the GUI.

Code:

```
from tkinter import *
import time
import tkinter.messagebox
from bot import chat
import pyttsx3
import threading
saved_username = ["You"]
#ans=["ADMIN"]
window_size="400x400"
class ChatInterface(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master
```

```
  # sets default bg for top level windows
        self.tl_bg = "#EEEEEE"
        self.tl_bg2 = "#EEEEEE"
        self.tl_fg = "#000000"
        self.font = "Verdana 10"
        menu = Menu(self.master)
        self.master.config(menu=menu, bd=5)
# Menu bar
    # File
        file = Menu(menu, tearoff=0)
        menu.add_cascade(label="File", menu=file)
       # file.add_command(label="Save Chat Log", command=self.save_chat)
        file.add_command(label="Clear Chat", command=self.clear_chat)
      #  file.add_separator()
        file.add_command(label="Exit",command=self.chatexit)

  # Options
        options = Menu(menu, tearoff=0)
        menu.add_cascade(label="Options", menu=options)
        # username
        # font
        font = Menu(options, tearoff=0)
        options.add_cascade(label="Font", menu=font)
        font.add_command(label="Default",command=self.font_change_default)
        font.add_command(label="Times",command=self.font_change_times)
        font.add_command(label="System",command=self.font_change_system)
        font.add_command(label="Helvetica",command=self.font_change_helvetica)
        font.add_command(label="Fixedsys",command=self.font_change_fixedsys)
```

```
        # color theme
        color_theme = Menu(options, tearoff=0)
        options.add_cascade(label="Color Theme", menu=color_theme)
        color_theme.add_command(label="Default",command=self.color_theme_default)
       # color_theme.add_command(label="Night",command=self.)
        color_theme.add_command(label="Grey",command=self.color_theme_grey)
        color_theme.add_command(label="Blue",command=self.color_theme_dark_blue)

        color_theme.add_command(label="Torque",command=self.color_theme_turquoise)
        color_theme.add_command(label="Hacker",command=self.color_theme_hacker)
       # color_theme.add_command(label='Mkbhd',command=self.MKBHD)
        help_option = Menu(menu, tearoff=0)
        menu.add_cascade(label="Help", menu=help_option)
        #help_option.add_command(label="Features", command=self.features_msg)
        help_option.add_command(label="About PyBot", command=self.msg)
        help_option.add_command(label="Develpoers", command=self.about)

        self.text_frame = Frame(self.master, bd=6)
        self.text_frame.pack(expand=True, fill=BOTH)

        # scrollbar for text box
        self.text_box_scrollbar = Scrollbar(self.text_frame, bd=0)
        self.text_box_scrollbar.pack(fill=Y, side=RIGHT)

        # contains messages
        self.text_box = Text(self.text_frame, yscrollcommand=self.text_box_scrollbar.set,
state=DISABLED,
                             bd=1, padx=6, pady=6, spacing3=8, wrap=WORD, bg=None, font="Verdana
10", relief=GROOVE,width=10, height=1)
```

```
        self.text_box.pack(expand=True, fill=BOTH)
        self.text_box_scrollbar.config(command=self.text_box.yview)

        # frame containing user entry field
        self.entry_frame = Frame(self.master, bd=1)
        self.entry_frame.pack(side=LEFT, fill=BOTH, expand=True)

        # entry field
        self.entry_field = Entry(self.entry_frame, bd=1, justify=LEFT)
        self.entry_field.pack(fill=X, padx=6, pady=6, ipady=3)
        # self.users_message = self.entry_field.get()

        # frame containing send button and emoji button
        self.send_button_frame = Frame(self.master, bd=0)
        self.send_button_frame.pack(fill=BOTH)

        # send button
        self.send_button = Button(self.send_button_frame, text="Send", width=5, relief=GROOVE,
bg='white',
                                  bd=1, command=lambda: self.send_message_insert(None),
activebackground="#FFFFFF",
                                  activeforeground="#000000")
        self.send_button.pack(side=LEFT, ipady=8)
        self.master.bind("<Return>", self.send_message_insert)

        self.last_sent_label(date="No messages sent.")
        #t2 = threading.Thread(target=self.send_message_insert(, name='t1')
        #t2.start()
```

```
    def playResponce(self,responce):
        x=pyttsx3.init()
        #print(responce)
        li = []
        if len(responce) > 100:
            if responce.find('--') == -1:
                b = responce.split('--')
                #print(b)

        x.setProperty('rate',120)
        x.setProperty('volume',100)
        x.say(responce)
        x.runAndWait()
        #print("Played Successfully.....")
 def last_sent_label(self, date):

        try:
            self.sent_label.destroy()
        except AttributeError:
            pass

        self.sent_label = Label(self.entry_frame, font="Verdana 7", text=date, bg=self.tl_bg2,
fg=self.tl_fg)
        self.sent_label.pack(side=LEFT, fill=X, padx=3)

    def clear_chat(self):
        self.text_box.config(state=NORMAL)
        self.last_sent_label(date="No messages sent.")
```

```
        self.text_box.delete(1.0, END)
        self.text_box.delete(1.0, END)
        self.text_box.config(state=DISABLED)

    def chatexit(self):
        exit()

    def msg(self):
        tkinter.messagebox.showinfo("CHATBOT v1.0",'CHATBOT is a chatbot for answering python queries\nIt is based on retrival-based NLP using pythons NLTK tool-kit module\nGUI is based on Tkinter\nIt can answer questions regarding python language for new learners')

    def about(self):
        tkinter.messagebox.showinfo("CHATBOT Developers","1.Rivesh Raja\n2.Deepak Kumar\n3.Riya Roy\n4.Swati")

    def send_message_insert(self, message):
        user_input = self.entry_field.get()
        pr1 = "You : " + user_input + "\n"
        self.text_box.configure(state=NORMAL)
        self.text_box.insert(END, pr1)
        self.text_box.configure(state=DISABLED)
        self.text_box.see(END)
        #t1 = threading.Thread(target=self.playResponce, args=(user_input,))
        #t1.start()
        #time.sleep(1)
        ob=chat(user_input)
        pr="ADMIN : " + ob + "\n"
        self.text_box.configure(state=NORMAL)
```

```
        self.text_box.insert(END, pr)
        self.text_box.configure(state=DISABLED)
        self.text_box.see(END)
        self.last_sent_label(str(time.strftime( "Last message sent: " + '%B %d, %Y' + ' at ' +
'%I:%M %p')))
        self.entry_field.delete(0,END)
        time.sleep(0)
        t2 = threading.Thread(target=self.playResponce, args=(ob,))
        t2.start()
        #return ob

    def font_change_default(self):
        self.text_box.config(font="Verdana 10")
        self.entry_field.config(font="Verdana 10")
        self.font = "Verdana 10"

    def font_change_times(self):
        self.text_box.config(font="Times")
        self.entry_field.config(font="Times")
        self.font = "Times"

    def font_change_system(self):
        self.text_box.config(font="System")
        self.entry_field.config(font="System")
        self.font = "System"

    def font_change_helvetica(self):
        self.text_box.config(font="helvetica 10")
        self.entry_field.config(font="helvetica 10")
```

```
        self.font = "helvetica 10"

    def font_change_fixedsys(self):
        self.text_box.config(font="fixedsys")
        self.entry_field.config(font="fixedsys")
        self.font = "fixedsys"

    def color_theme_default(self):
        self.master.config(bg="#EEEEEE")
        self.text_frame.config(bg="#EEEEEE")
        self.entry_frame.config(bg="#EEEEEE")
        self.text_box.config(bg="#FFFFFF", fg="#000000")
        self.entry_field.config(bg="#FFFFFF", fg="#000000", insertbackground="#000000")
        self.send_button_frame.config(bg="#EEEEEE")
        self.send_button.config(bg="#FFFFFF", fg="#000000", activebackground="#FFFFFF",
activeforeground="#000000")
        #self.emoji_button.config(bg="#FFFFFF", fg="#000000", activebackground="#FFFFFF",
activeforeground="#000000")
        self.sent_label.config(bg="#EEEEEE", fg="#000000")

        self.tl_bg = "#FFFFFF"
        self.tl_bg2 = "#EEEEEE"
        self.tl_fg = "#000000"

    # Dark
    def color_theme_dark(self):
        self.master.config(bg="#2a2b2d")
        self.text_frame.config(bg="#2a2b2d")
        self.text_box.config(bg="#212121", fg="#FFFFFF")
```

```
        self.entry_frame.config(bg="#2a2b2d")
        self.entry_field.config(bg="#212121", fg="#FFFFFF", insertbackground="#FFFFFF")
        self.send_button_frame.config(bg="#2a2b2d")
        self.send_button.config(bg="#212121", fg="#FFFFFF", activebackground="#212121",
activeforeground="#FFFFFF")
       # self.emoji_button.config(bg="#212121", fg="#FFFFFF", activebackground="#212121",
activeforeground="#FFFFFF")
        self.sent_label.config(bg="#2a2b2d", fg="#FFFFFF")


        self.tl_bg = "#212121"
        self.tl_bg2 = "#2a2b2d"
        self.tl_fg = "#FFFFFF"


    # Grey
    def color_theme_grey(self):
        self.master.config(bg="#444444")
        self.text_frame.config(bg="#444444")
        self.text_box.config(bg="#4f4f4f", fg="#ffffff")
        self.entry_frame.config(bg="#444444")
        self.entry_field.config(bg="#4f4f4f", fg="#ffffff", insertbackground="#ffffff")
        self.send_button_frame.config(bg="#444444")
        self.send_button.config(bg="#4f4f4f", fg="#ffffff", activebackground="#4f4f4f",
activeforeground="#ffffff")
        #self.emoji_button.config(bg="#4f4f4f", fg="#ffffff", activebackground="#4f4f4f",
activeforeground="#ffffff")
        self.sent_label.config(bg="#444444", fg="#ffffff")
        self.tl_bg = "#4f4f4f"
        self.tl_bg2 = "#444444"
        self.tl_fg = "#ffffff"
```

```
    def color_theme_turquoise(self):
        self.master.config(bg="#003333")
        self.text_frame.config(bg="#003333")
        self.text_box.config(bg="#669999", fg="#FFFFFF")
        self.entry_frame.config(bg="#003333")
        self.entry_field.config(bg="#669999", fg="#FFFFFF", insertbackground="#FFFFFF")
        self.send_button_frame.config(bg="#003333")
        self.send_button.config(bg="#669999", fg="#FFFFFF", activebackground="#669999",
activeforeground="#FFFFFF")
        #self.emoji_button.config(bg="#669999", fg="#FFFFFF", activebackground="#669999",
activeforeground="#FFFFFF")
        self.sent_label.config(bg="#003333", fg="#FFFFFF")


        self.tl_bg = "#669999"
        self.tl_bg2 = "#003333"
        self.tl_fg = "#FFFFFF"


    # Blue
    def color_theme_dark_blue(self):
        self.master.config(bg="#263b54")
        self.text_frame.config(bg="#263b54")
        self.text_box.config(bg="#1c2e44", fg="#FFFFFF")
        self.entry_frame.config(bg="#263b54")
        self.entry_field.config(bg="#1c2e44", fg="#FFFFFF", insertbackground="#FFFFFF")
        self.send_button_frame.config(bg="#263b54")
        self.send_button.config(bg="#1c2e44", fg="#FFFFFF", activebackground="#1c2e44",
activeforeground="#FFFFFF")
        #self.emoji_button.config(bg="#1c2e44", fg="#FFFFFF", activebackground="#1c2e44",
activeforeground="#FFFFFF")
```

```
        self.sent_label.config(bg="#263b54", fg="#FFFFFF")


        self.tl_bg = "#1c2e44"
        self.tl_bg2 = "#263b54"
        self.tl_fg = "#FFFFFF"
# Torque
    def color_theme_turquoise(self):
        self.master.config(bg="#003333")
        self.text_frame.config(bg="#003333")
        self.text_box.config(bg="#669999", fg="#FFFFFF")
        self.entry_frame.config(bg="#003333")
        self.entry_field.config(bg="#669999", fg="#FFFFFF", insertbackground="#FFFFFF")
        self.send_button_frame.config(bg="#003333")
        self.send_button.config(bg="#669999", fg="#FFFFFF", activebackground="#669999",
activeforeground="#FFFFFF")
        #self.emoji_button.config(bg="#669999", fg="#FFFFFF", activebackground="#669999",
activeforeground="#FFFFFF")
        self.sent_label.config(bg="#003333", fg="#FFFFFF")


        self.tl_bg = "#669999"
        self.tl_bg2 = "#003333"
        self.tl_fg = "#FFFFFF"


    # Hacker
    def color_theme_hacker(self):
        self.master.config(bg="#0F0F0F")
        self.text_frame.config(bg="#0F0F0F")
        self.entry_frame.config(bg="#0F0F0F")
        self.text_box.config(bg="#0F0F0F", fg="#33FF33")
```

```
        self.entry_field.config(bg="#0F0F0F", fg="#33FF33", insertbackground="#33FF33")

        self.send_button_frame.config(bg="#0F0F0F")

        self.send_button.config(bg="#0F0F0F", fg="#FFFFFF", activebackground="#0F0F0F",
activeforeground="#FFFFFF")

        #self.emoji_button.config(bg="#0F0F0F", fg="#FFFFFF", activebackground="#0F0F0F",
activeforeground="#FFFFFF")

        self.sent_label.config(bg="#0F0F0F", fg="#33FF33")


        self.tl_bg = "#0F0F0F"

        self.tl_bg2 = "#0F0F0F"

        self.tl_fg = "#33FF33"


      # Default font and color theme

    def default_format(self):

        self.font_change_default()

        self.color_theme_default()

root=Tk()

a = ChatInterface(root)

root.geometry(window_size)

root.title("LPU PLACEMENT CELL QUERIES CHATBOT")

root.iconbitmap('i.ico')

root.mainloop()
```

5. DESCRIPTION OF WORK DIVISION IN TERMS OF ROLES AMONG STUDENTS.

1) Name: Rivesh Raja

Work : write the code to collect information and make the chatbot implementation and together with Riya Roy, Deepak Kumar, and Swati worked on GUI.

2) Name: Deepak Kumar

Work : Helped in writing the chatbot codes implementation and GUI.

3) Name: Riya Roy

Work : write code for GUI and implementation of tkinter

4) name : Swati

Work : helped in GUI and in finding the data information about the Placement cell.

Together we make the Report file.

Reference -www.geeksforgeeks.com

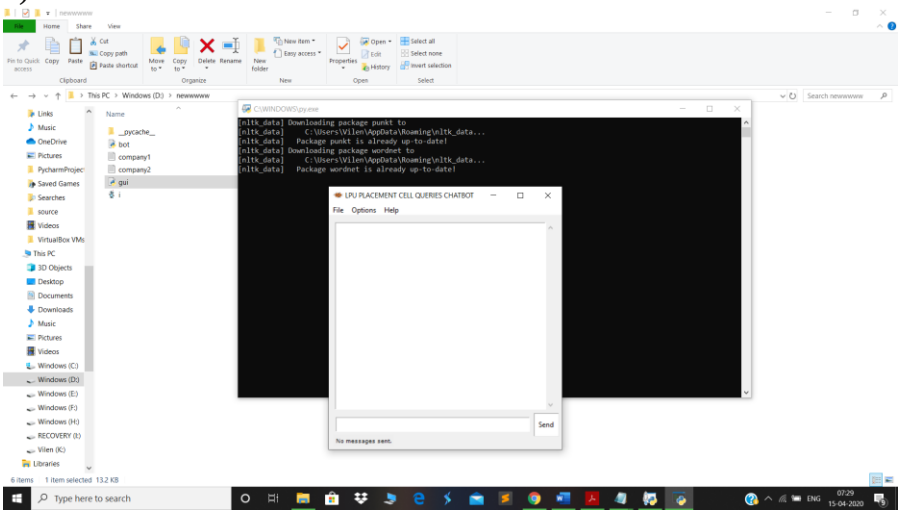
Reference 2- https://www.revechat.com/blog/what-is-a-chatbot/

RUN THE CHATBOT

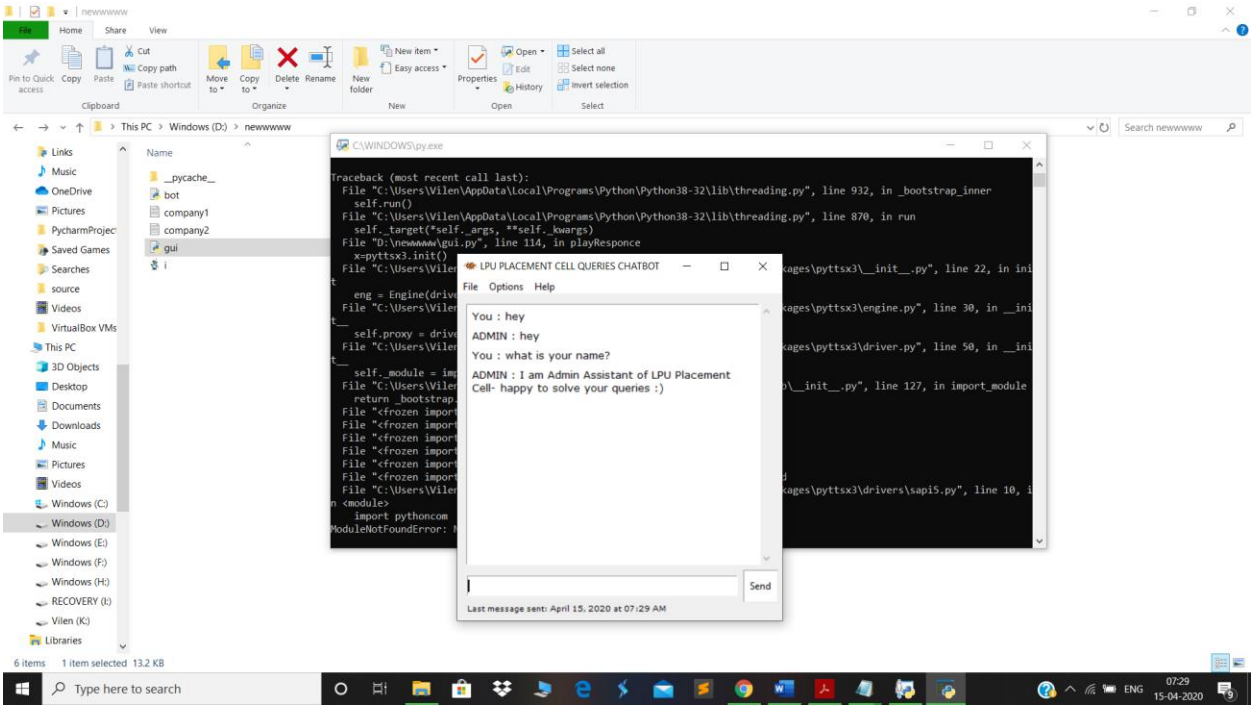
To run the chatbot, we have two main files; bot.py and gui.py.

6. SCREENSHOTS

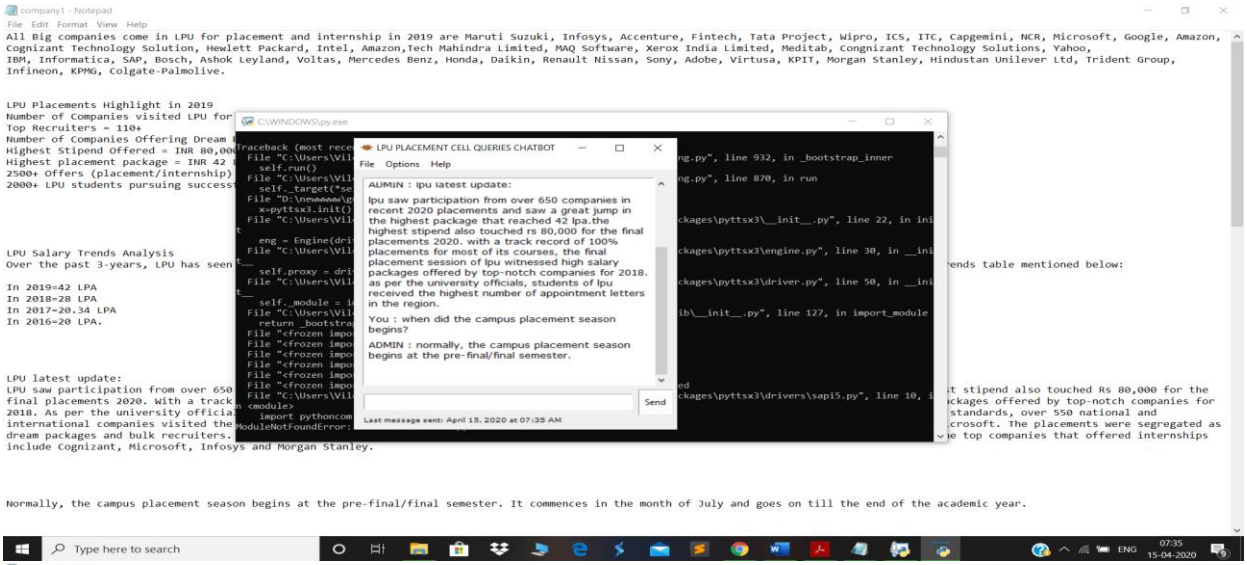
1) initial window

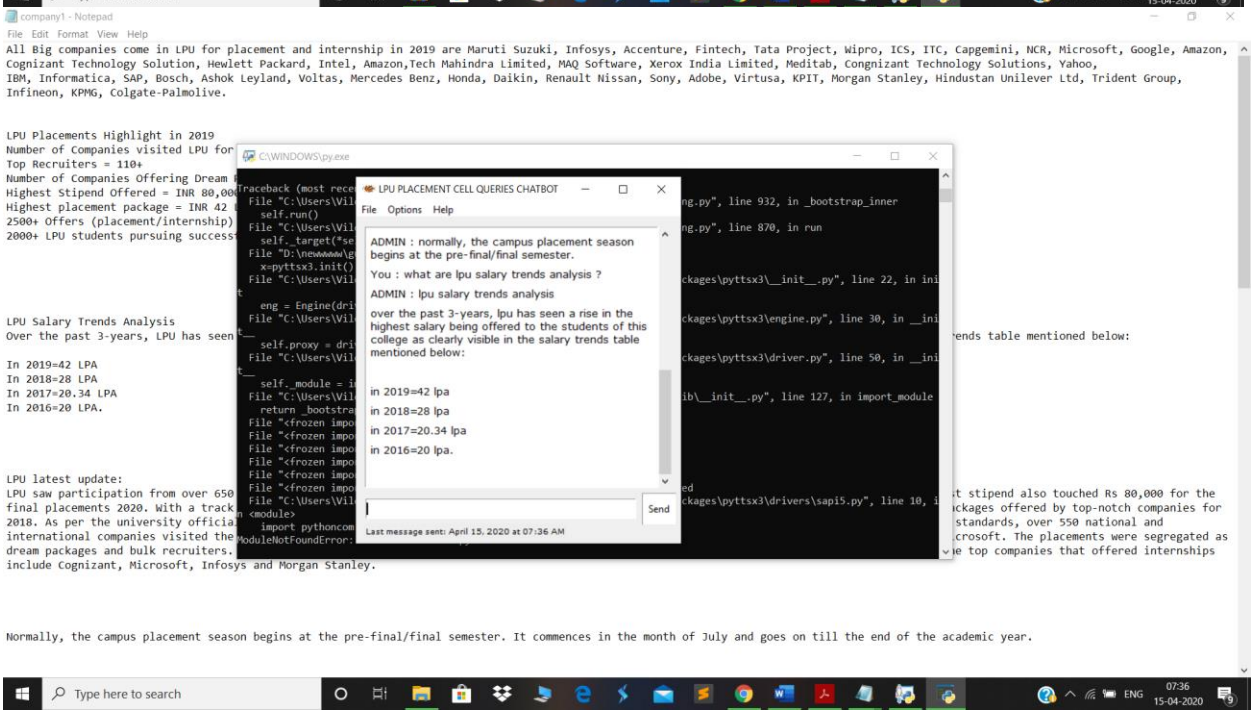


2)writing hello



3)Asking some random Questions





7. SWOT Analysis

Strength

- Code is precise and short
- Code is easy to understand
- Complexity of code is lesser than other methods used.
- Good interface

Weakness

- Can’t recognize little similar question precisely

Opportunities

- Can use this software at Placement Query
- Can be used as to reply general question without human interection.
- Augmented reality.

Threats

- This can be get confused
- This AI simply works on database, cannot learn (for example it can’t)
- Not easy to manage the database.

So, basically this particular type of chatbot can be used at smaller level where only human entry is needed and there’s small data. But it can exchange the person from computer bot to answer random question and help to do multitasking and save efforts of person to reply all the logic static message to share information to user.

8. SUMMARY

In this Python data science project, we understood about chatbots and implemented a deep learning version of a chatbot in Python which is accurate. You can customize the data according to business requirements and train the chatbot with great accuracy. Chatbots are used everywhere and all businesses is looking forward to implementing bot in their workflow.