A REPORT

ON

WhatsApp Business Chatbot

By

Bollavaram Golla Riteesh Ram Chander-1800206C203

Prepared in the partial fulfillment of the

Practice School II Course

AT

RannLab Technologies Pvt. Ltd.

805, 8th Floor, Om Tower, (Above Domino's Alpha Commercial Belt, Alpha 1, Block E,Greater Noida, Uttar Pradesh 201310

A Practice School II Station of

BML MUNJAL UNIVERSITY

Table of contents

BML MUNJAL UNIVERSITY
FROM HERE TO THE WORLD

Certificate of authenticity

CERTIFICATE

This is to certify that Practice School Project of **Bollavaram Golla Riteesh Ram Chander** titled **WhatsApp Business Chatbot** is an original work and that this work has not been submitted anywhere in any form. Indebtedness to other works/publications has been duly acknowledged at relevant places. The project work was carried during **May 25, 2020** to **July 20, 2020** in Rannlab Technologies Pvt. Ltd.

Signature of PS-II faculty	**Signature of industry mentor/Supervisor**
Name: Dr. Atul Mishra	**Name: Mr. Umesh Kushwaha**
Designation: Professor, BML Munjal University	**Designation: Co-founder and CEO of RannLab**
(Seal of the organization with Date)	[illegible] Rann Lab Technologies Pvt. Ltd ***(Seal of the organization with Date)***

BML MUNJAL UNIVERSITY
FROM HERE TO THE WORLD

Acknowledgement

This project has been inspired to create an automated WhatsApp messages for business purpose. To accomplish this there is a requirement to create a chatbot. Basically this chatbot is a software which performs automated tasks like answering the queries, giving the information about something and used in normal conversations. We can say this chatbot as an Application bot which behaves like a scripted bot as well as an intelligent bot. So by using a chatbot in WhatsApp for business purpose the customers can easily chat with the bot and even the bot can easily answer their queries and gives some required information. So finally this project was assigned to me a part of my evaluation for the PS-2 Internship. All the resources and references used as a part of this project were procured online and were used by adhering to strict plagiarism policy as mentioned. As a token of gratitude, I would like to give my humble vote of thanks to all the men and women behind this massive academic endeavor. My thanks go to the:

(i) Vice-Chancellor: Dr. Manoj Kumar Arora
(ii) Dean SoET: Dr.Maneek Kumar
(iii) Co-ordinator in-charge: Dr. Maheshwar Dwivedy
(iv) Professional Expert in charge of the project: Mr. Pranshu Shrivastava
(v) Faculty of the University: Dr. Atul Mishra

B.G.Riteesh Ram

Signature of student

Objective:

In this project I am going to create a WhatsApp Business chatbot or simply a WhatsApp chatbot for Business which will send automated messages from the company side to their customers/users. In this project I took an example of a product selling company which will sell Cadbury Dairy Milk Silk chocolates to their customers. This bot will solve the customer questions/issues/queries, chat with the customers normally and sell the products. This bot has some scripted data like there will be a .json file which stores the user objectives, set of questions that may be asked by the user and the set of replies that the bot should give. Hence at the time of conversation the chatbot follows predetermined paths. But also this bot will be intelligent too by using the concepts of Deep Neural Networks and Natural Language Processing. I also created a Database which will store user and bot messages including the user mobile/phone number because multiple users can chat with the bot. Here I used SQLite3 for database creation, table creation and also inserting data into tables. Basically in this project I'm using Twilio API which will act as an interface between WhatsApp server as well as flask app that I have created for my bot.

Problem Statement:

WhatsApp as we already know is a free and cross platform messaging service currently owned by Facebook. It is very popular all over the world and has over 1.5 billion users. In January 2018 WhatsApp launched WhatsApp Business an application which helps businesses to interact with their customers directly. But soon the need of automation became obvious lets take an example that there is a large enterprise company named as ‘x’ this is a product selling company here the company will have millions of customers, so each and every person will have their own concerns, queries and issues so to chat with each and every person manually is not possible. So to overcome this problem one need to use a bot. Therefore in august 2018 Facebook announced the launch of WhatsApp business API. Here this set of API will let businesses to implement automated messaging or in simpler words bots. So now our problem is the set up a chatbot which will send automated messages to their customers.
In my project I’m not using WhatsApp Business API, because to use Business API I need a permission from the WhatsApp team which will take a lot of time. Instead of Business API I used Twilio API to create a WhatsApp Business chatbot. One can use Twilio in developer environment. Hence by this reason I’m using Twilio API which doesn’t require any permissions to access.

Company Profile: Rannlab Technologies Pvt. Ltd.:

Rannlab Technologies Pvt. Ltd. Is an emerging software company which deals in providing business consulting services, Enhanced Education services and Innovative Health Care services, Society Oriented Software's, information systems with commercial ERP solutions. This company has specialities like Ionic App Development, Linux Administration, Network Security, IT Support, Windows Server Administration, Windows Application Development, Moblie Application Development, Web Application Development and Customised ERP Development.

RannLab has shown a consistent record of dynamic growth since its inception and enjoys the trust and value of its clients. RannLab have a team of IT professionals that experts in each of the unique skills necessary for a strong internet presence. The aim is to provide meet the best services to help clients meet their business needs and to optimize their strengths. With its examined processes this company makes recommendations to maximize a client's profitability. We serve an extensive portfolio of clients, which have soared high within a short span of time without any hassles.

BML MUNJAL
UNIVERSITY
FROM HERE TO THE WORLD

Methodology:

In this section we will be going across the methodologies like:
1. How to create an echo WhatsApp chatbot.
2. How to create a chatbot that runs on command prompt.
3. How to integrate the chatbot that is running on command prompt with WhatsApp.
So before going into the details of above mentioned methodologies, in Problem statement section I mentioned instead of WhatsApp Business API I'm using Twilio API. So firstly let's discuss how to set up WhatsApp chatbot using WhatsApp Business API.
Note: I haven't used WhatsApp Business API in my project the below mentioned things related to WhatsApp Business API is only the research that I have done for how to create a WhatsApp Business chatbot using WhatsApp Business API.

WhatsApp Business API:

If one need to get the access to use WhatsApp Business API he/she should follow some procedures:
I) Verifying one's Business in FBM(Facebook Business Manager).
II) In that Facebook Business Manager one should setup the WhatsApp Business account.
After this one need to give a verification for WhatsApp Business API. That should be approved by the WhatsApp Team to move further.
III) To the business account one need to setup their respective phone number and display name. Even one need to enter some required details. When all these reviews are completed, then to create a WhatsApp Business API client one requires business64-encoded certificate.

Now to send messages one need to setup WhatsApp Business API client. Firstly one should login as an admin. Secondly by using Docker one need to setup an environment for WhatsApp Business API.
What is a Docker?
Docker: It is a platform that delivers software's in the form of containers. By using OS-level virtualization it delivers the containers. One can store flies, software's and libraries in these containers. By using Docker one can install and update WhatsApp Business API client. After the client is running perfectly one can register their respected mobile numbers. After that there are two things to follow:
i) Application settings
ii) Business Profile settings
Application setting are like how the client is working:call back settings, on call-pager, media download behaviour, webhooks, sent status.
Business Profile settings are like address, email, website, industry, about info and profile pic.
To test whether the Business API is working fine or not, one can send test messages. Those messages are like text, messing templates, documents and media. After all this done one is required to make

some advanced setups like Monitoring, Media, Messaging Templates, Two-factor verification, Backup/restore, Webhooks.
After all these done one should think about how to create a chatbot. And integrate that chatbot code with WhatsApp Business API client. Hence this is the approach that one should follow to create a WhatsApp Business chatbot using WhatsApp business API.

Now lets come into the original methodology, at the very beginning of this section I said three things that I did in my project. Firstly creating an echo bot for WhatsApp, Secondly writing the chatbot code using python that will run on command prompt and finally how to integrate the chatbot that is running on command prompt with WhtatsApp account.

How to create an echo WhatsApp chatbot:

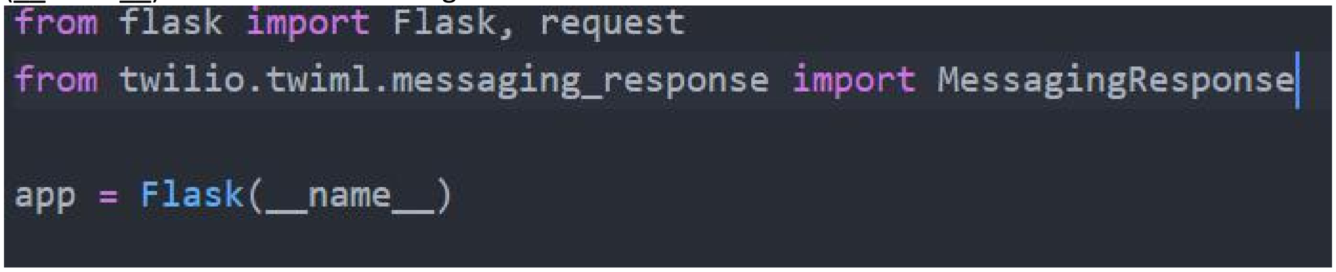
Now lets discuss how to create an echo WhatsApp chatbot. To do that one need to follow the steps like:
I) Setting up Twilio sandbox
II) Creating the flask application
III) Getting Public URL using Ngrok and configuring sandbox
IV) Creating github repository
V) Integrating the flask app that is present in github to Heroku cloud platform.

1. Setting up Twilio sandbox:
Firstly one need to create a Twilio sandbox, ok now lets see what is the definition and use of Twilio in my project.
Twilio: It is a cloud communication platform which helps developers to make and receive phone calls, performs communication functions using some web API's and even we use it to send and receive text messages.
Let me explain a bit about why one exactly need Twilio API. Basically there will be a particular sandbox that provided to one's Twilio account. Now he/she need to enter that sandbox name through their WhatsApp account so, by doing this they can access their Twilio API through WhatsApp account. Now in Twilio console particularly in sandbox section there one need to mention the URL of the flask application. Like first he/she deploy the flask app on some cloud platform after deployment a URL will be given to that particular app. After that they need to paste that URL in sandbox section so when a user sends a message then that message will be sent to Twilio now Twilio will call the URL that is pasted. Thus by this way the flask app will be called and it will generate a response and sends it to Twilio and finally goes to WhatsApp.
Now lets see how to set up Twilio sandbox:
I) Make account on Twilio
II)Create new project
 i) Select products

ii) Choose programmable SMS
iii) Click continue
iv) Give a name to one's project
V) Click continue

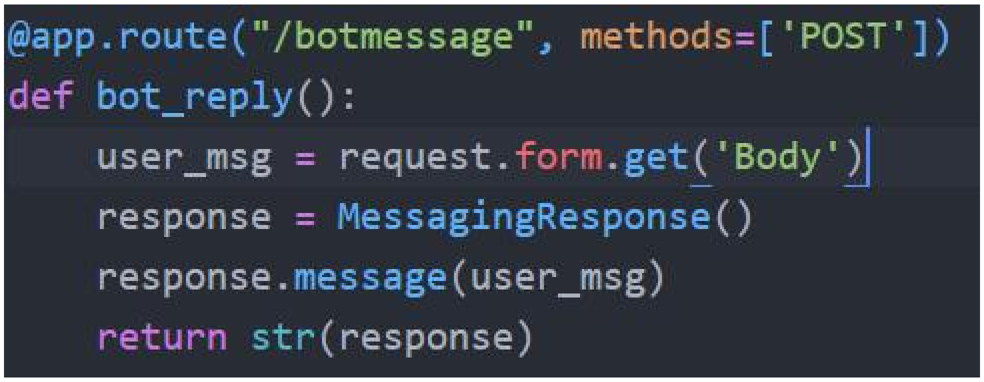
III) Open programmable SMS on dashboard on project console and select WhatsApp beta
IV) Link ones WhatsApp phone number to their sandbox
Now we are done with setting up Twilio sandbox.

2. Creating the flask application:
Lets see why one need a flask application. Basically this echo bot will take a user input from WhatsApp account, fetch the message and generate the echo response. So to do that one is required to create a flask application. Firstly in command prompt one should type pip install flask, pip install twilio to install flask and twilio. After that one can start working on flask application code. Firstly import Flask class and request from flask, import MessagingResponse function from twilio. Here one need Flask class to create an object, that object is the flask application. The current module (__name__) will be taken as an argument for Flask constructor.



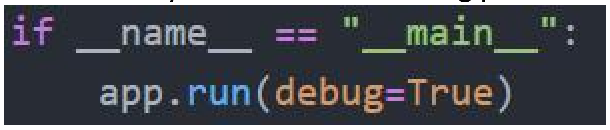
```
from flask import Flask, request
from twilio.twiml.messaging_response import MessagingResponse

app = Flask(__name__)
```

Now one need to define a function lets name it as bot_reply() to fetch the user message and to generate the echo response. For that firstly call route() which is present in the Flask class. This route() is a decorator that tells to the flask application about which URL that one should mention to call specific function. In my case I need to call /botmessage URL to call bot_reply() function and also one should mention the method in my case I mentioned POST HTTP method. Basically we have two HTTP methods POST, GET these HTTP methods will help to send and receive information or data to and from web server. I used POST which will send information or data to web server. Now in bot_reply() function to fetch the user message one should use request() from flask. Like the request.form.get('body') will get the user message. To generate a response one need to call MessagingResponse() and store it in response variable. By using response variable they can generate response by calling message(). In case of echo bot one should specify response.message('user_msg'). At last they should return the string form of response variable.



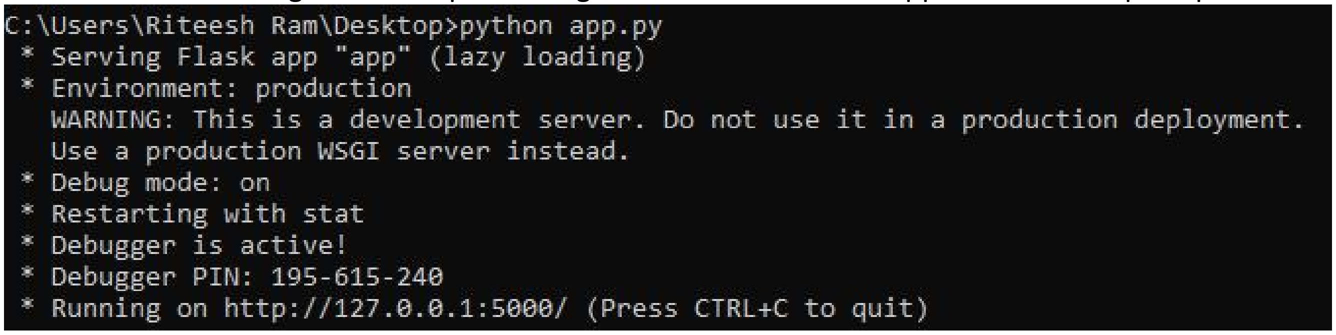
```
@app.route("/botmessage", methods=['POST'])
def bot_reply():
    user_msg = request.form.get('Body')
    response = MessagingResponse()
    response.message(user_msg)
    return str(response)
```

Finally one should run the flask application for that they can use run() method from Flask class. In run method the actual parameters are host, port, debug and options but in this case we only need and should use debug parameter. Why debug parameter? Because in case of developing flask application the app should be restarted manually again with even a small change in code. So to do that automatically one should use debug parameter and set it as True.



```
if __name__ == "__main__":
    app.run(debug=True)
```

The below shown image is the output that I got when I runned flask app on command prompt



```
C:\Users\Riteesh Ram\Desktop>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 195-615-240
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

By this way the flask app is created now it’s time to generate public URL using ngrok.

3. Getting Public URL using Ngrok and configuring sandbox:

Why I used ngrok because when one will run the flask application on command prompt it will generate local host URL which only the creater can use to make that URL as public or to get public URL one should use the service of ngrok. Basically ngrok will provide a session time for 8hrs the URL generated by ngrok will work for that particular session time only.

Ngrok: It is a cross-platform application that will make developers to expose local development server to internet. For ngrok we don’t need any domain name or public IP.

Steps to work with ngrok:

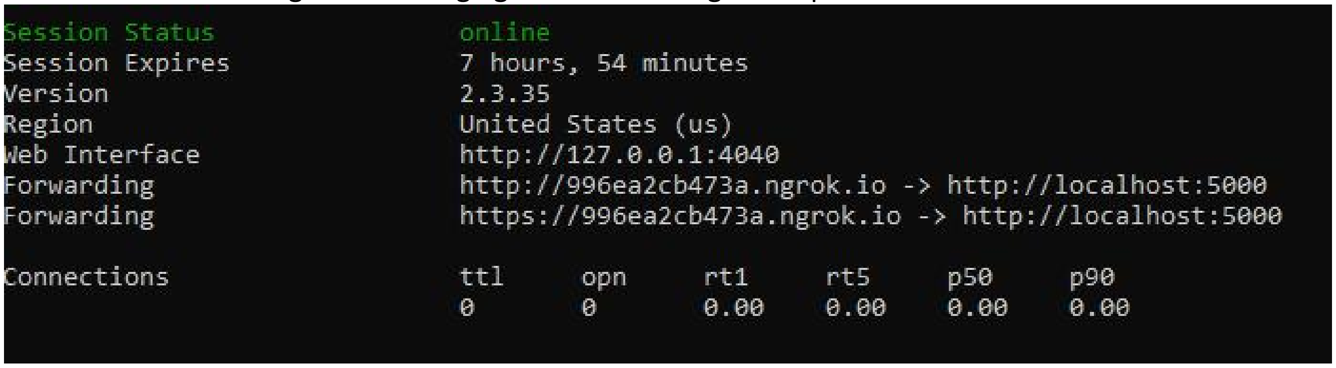
I) Run the flask application in command prompt

II) Generate public URL for Webhook using ngork.io

i) Download ngrok and unzip it

ii) Run ngork in command prompt (using ngrok http 5000 command) copy the HTTPS forwarding URL

III) Paste it as the Webhook URL for incoming messages in your sandbox configuration.
IV) Finally save it
By doing this one can chat with the echo bot through their respected WhatsApp accounts. But remember this URL will work for only 8hrs after 8hrs one should generate another URL by typing: Ngrok http 5000 in command prompt. So to overcome this issue one should deploy flask app on cloud platform in my project I used Heroku cloud platform.
The below shown image is the thing I got when I run ngork http 5000 in cmd.

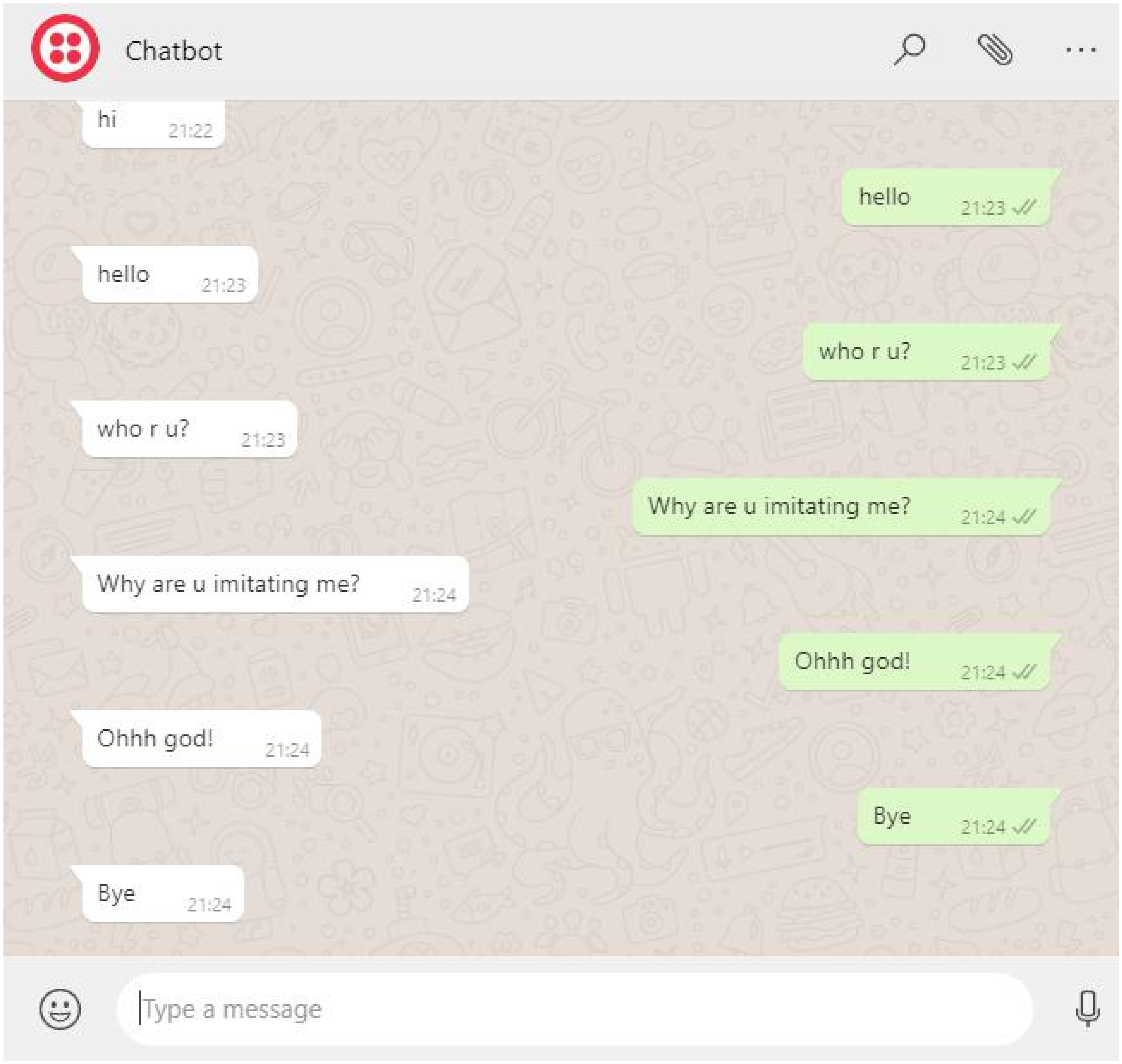


```
Session Status                online
Session Expires               7 hours, 54 minutes
Version                       2.3.35
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://996ea2cb473a.ngrok.io -> http://localhost:5000
Forwarding                    https://996ea2cb473a.ngrok.io -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

4. Creating github repository:
One need to create a github repository with the flask app, requirements.txt, runtime.txt and procfile. Let me discuss the us of all these files.
Flask app: In creating flask app section I discussed the use of flask app in my project so no need to discuss any thing about it, just create a github repository and upload it on that repository.
Requirements.txt: This file will contain all the required libraries, packages that are useful to run the flask app.
Command: pip freeze > requirements.txt
Runtime.txt: This file will contain the required version of python like one need to specify the particular version of python that he/she want to run.
Command: python-3.6.8
Procfile: In this procfile one should specify that the flask app is a web gunicorn. For WSGI applications gunicorn is a pure python HTTP server. It will let us to run the python applications concurrently by running multiple python processes in a single dyno.
Command: web gunicorn app:app
Note: In case of requirements.txt one should create a separate python virtual environment and install

flask , twilio then only it will take the requirements of the packages or libraries that he/she need or else it will take all the packages, libraries present in their python environment. In my case I haven't created any virtual environment.
Now we are done with github repository, it's time to connect Heroku app with github repository and deploy flask app on Heroku.

5. Integrating the flask app that is present in github to Heroku cloud platform:
The below mentioned procedure is the way how to proceed with Heroku
I) Open Heroku and create a new account in Heroku.
II) Now login to your Heroku account.
III) Click on create a new app and mention the name of the application you want.
IV) Now it will redirect to your personal section and here click on the app
V) In your app go to the deploy section, scroll down and click on github
VI) Then after enter your github repository name and press connect
VII) Enable the automatic deploys and select the branch name
VIII) Finally select the Deploy Branch and the deployment will start
Finally one will get a URL, copy that URL and paste it in place of ngrok URL in sandbox. Now save it finally the echo bot creation is done and one can use it on their respective WhatsApp account.
The below shown image is my conversation with the WhatsApp echo bot I have created.



hi 21:22

hello 21:23

hello 21:23

who r u? 21:23

who r u? 21:23

Why are u imitating me? 21:24

Why are u imitating me? 21:24

Ohhh god! 21:24

Ohhh god! 21:24

Bye 21:24

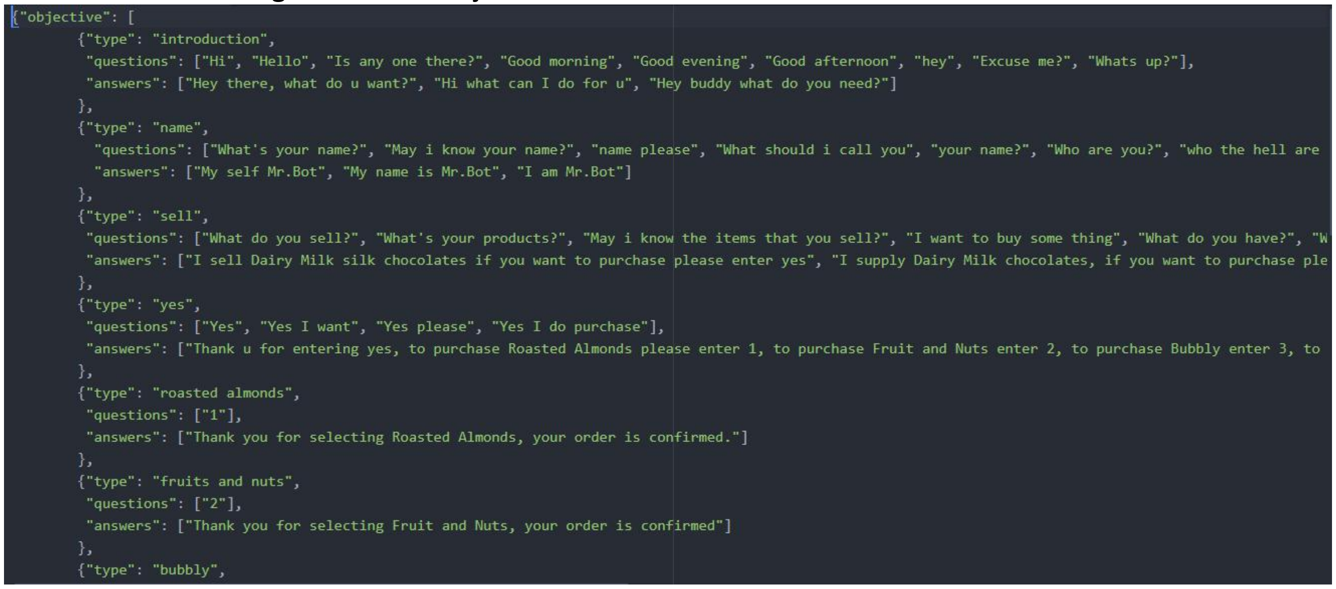
Bye 21:24

Now we are done with creating an echo bot so lets go into the next methodology like how to create a python based chatbot that will run on command prompt.

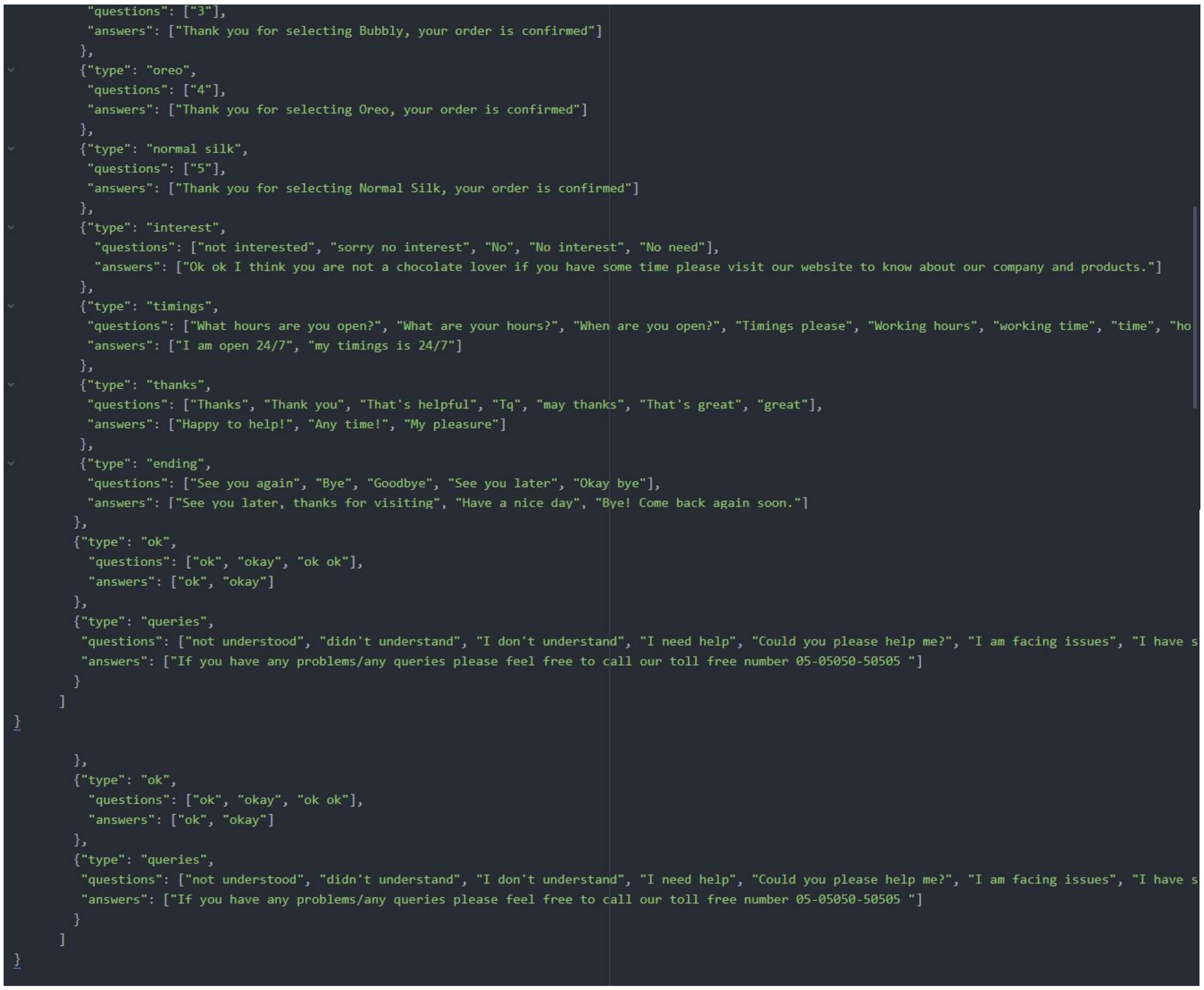
How to create a python chatbot that runs on command prompt:

I wrote a python code for chatbot before going into the python code firstly one need to create a .json file. Basically in this file one can create their intents or objectives, the below shown image is the .json

file that I have created. In this .json file I took an example of a cadbury diary milk silk selling company is there and for that company I should design the bot that will give responses to the specified user intents/objectives. Basically there will be a dictionary with key=objective and there will be a list with some dictionaries in it. And in each dictionary there will be a key=type and the corresponding values like introduction, name, sell, yes, roasted almonds, fruits & nuts, bubbly, oreo, normal silk, interest, timings, thanks, ending, ok, queries. In this way one can specify their own objectives/intents. Here the questions will specify a list of questions asked by the customers/users that belongs to any particular type that I have mentioned above and the answers are a list of answers that the bot should randomly pick one from answers for a corresponding type and reply to the user/customer.
The below show image is the intent.json file that I have created



```
{"objective": [
        {"type": "introduction",
         "questions": ["Hi", "Hello", "Is any one there?", "Good morning", "Good evening", "Good afternoon", "hey", "Excuse me?", "Whats up?"],
         "answers": ["Hey there, what do u want?", "Hi what can I do for u", "Hey buddy what do you need?"]
        },
        {"type": "name",
          "questions": ["What's your name?", "May i know your name?", "name please", "What should i call you", "your name?", "Who are you?", "who the hell are
          "answers": ["My self Mr.Bot", "My name is Mr.Bot", "I am Mr.Bot"]
        },
        {"type": "sell",
         "questions": ["What do you sell?", "What's your products?", "May i know the items that you sell?", "I want to buy some thing", "What do you have?", "W
         "answers": ["I sell Dairy Milk silk chocolates if you want to purchase please enter yes", "I supply Dairy Milk chocolates, if you want to purchase ple
        },
        {"type": "yes",
         "questions": ["Yes", "Yes I want", "Yes please", "Yes I do purchase"],
         "answers": ["Thank u for entering yes, to purchase Roasted Almonds please enter 1, to purchase Fruit and Nuts enter 2, to purchase Bubbly enter 3, to
        },
        {"type": "roasted almonds",
         "questions": ["1"],
         "answers": ["Thank you for selecting Roasted Almonds, your order is confirmed."]
        },
        {"type": "fruits and nuts",
         "questions": ["2"],
         "answers": ["Thank you for selecting Fruit and Nuts, your order is confirmed"]
        },
        {"type": "bubbly",
```



```
          "questions": ["3"],
          "answers": ["Thank you for selecting Bubbly, your order is confirmed"]
        },
        {"type": "oreo",
          "questions": ["4"],
          "answers": ["Thank you for selecting Oreo, your order is confirmed"]
        },
        {"type": "normal silk",
          "questions": ["5"],
          "answers": ["Thank you for selecting Normal Silk, your order is confirmed"]
        },
        {"type": "interest",
           "questions": ["not interested", "sorry no interest", "No", "No interest", "No need"],
           "answers": ["Ok ok I think you are not a chocolate lover if you have some time please visit our website to know about our company and products."]
        },
        {"type": "timings",
          "questions": ["What hours are you open?", "What are your hours?", "When are you open?", "Timings please", "Working hours", "working time", "time", "ho
          "answers": ["I am open 24/7", "my timings is 24/7"]
        },
        {"type": "thanks",
          "questions": ["Thanks", "Thank you", "That's helpful", "Tq", "may thanks", "That's great", "great"],
          "answers": ["Happy to help!", "Any time!", "My pleasure"]
        },
        {"type": "ending",
          "questions": ["See you again", "Bye", "Goodbye", "See you later", "Okay bye"],
          "answers": ["See you later, thanks for visiting", "Have a nice day", "Bye! Come back again soon."]
       },
       {"type": "ok",
          "questions": ["ok", "okay", "ok ok"],
          "answers": ["ok", "okay"]
       },
       {"type": "queries",
         "questions": ["not understood", "didn't understand", "I don't understand", "I need help", "Could you please help me?", "I am facing issues", "I have s
         "answers": ["If you have any problems/any queries please feel free to call our toll free number 05-05050-50505 "]
       }
     ]
}
```

```
       },
       {"type": "ok",
          "questions": ["ok", "okay", "ok ok"],
          "answers": ["ok", "okay"]
       },
       {"type": "queries",
         "questions": ["not understood", "didn't understand", "I don't understand", "I need help", "Could you please help me?", "I am facing issues", "I have s
         "answers": ["If you have any problems/any queries please feel free to call our toll free number 05-05050-50505 "]
       }
     ]
}
```

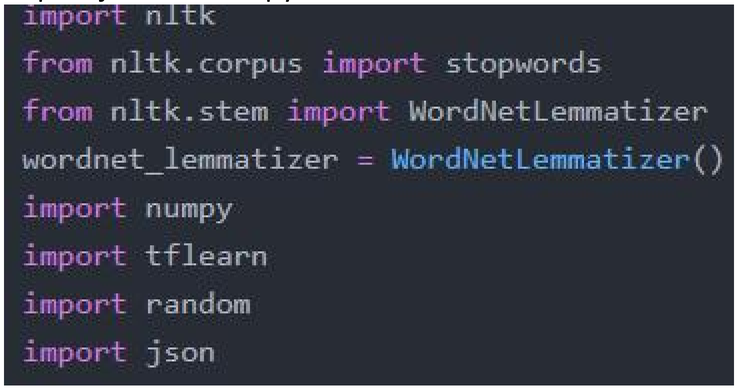
After creating ones intent.json file one can start writing python code to do that firstly they need to import some packages/libraries/frameworks like nltk, numpy, tflearn, tensorflow, random, json etc. Now before going into the code lets see the definition and use of that particular library/framework in the chatbot code.

NLTK: NLTK stands for Natural Language Tool Kit. This library is built to write python programs that will work on human natural language. This NLTK can do semantic reasoning, tagging, stemming, lemmatization, classification, tokenizing and phrasing. In my python code I used nltk because to do word tokenization, lemmatization and to remove stop-words.

I) Word tokenization: In this code one should use word_tokenize() function to split a string or (sentence) into words.

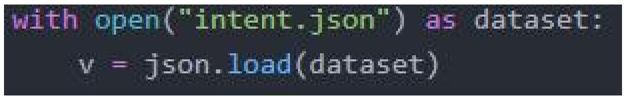
II) Lemmatization: In this code one should use word lemmatization to convert a word into it's root

form and also considers the words that are in similar meaning as a single unit basically it will group them.
III) Stop-words: In English language there are some words which will doesn't add any/much meaning to sentence's so to remove such words from sentences one is required to use stop-words concept.
Numpy: It is a python library which will give fast and efficient operations on arrays. In this code one use numpy because there is a need to convert python lists into arrays here I'm using numpy arrays because I have to give input to my Neural Network so for that I need to use arrays so I used numpy.array() to convert python lists into arrays.
Tflearn: It is a deep learning library which is on the top of tensorflow. Even it is full transparency over tensorflow. It is an independent library and in this code when to build a Deep Neural Network for predictions it is better to use tflearn instead of tensorflow because by using tflearn the code complexity will decreases like no need to initialize any random weights, biases and code part will also decreases we will discuss more about it in further.
Tensorflow: It is a python library created by Google. Tensorflow will be used to create a deep learning models directly or simply by using some wrapper libraries. In this code actually one no need to use tensorflow, instead one can use tflearn but to run tflearn one need tensorflow to be installed in their system.
Random: It is a package which is used to pick up random answers from the answers list present in the intent.json file. Basically user says something that may belong to any particular objective type so the bot should give a response that belongs to that particular objective only. That's why to pick up answer from answers list one can use random.choice(answers) function.
Json: Like I said I have created a .json file so one need to import that file, to do that first they have to import json to their python file.



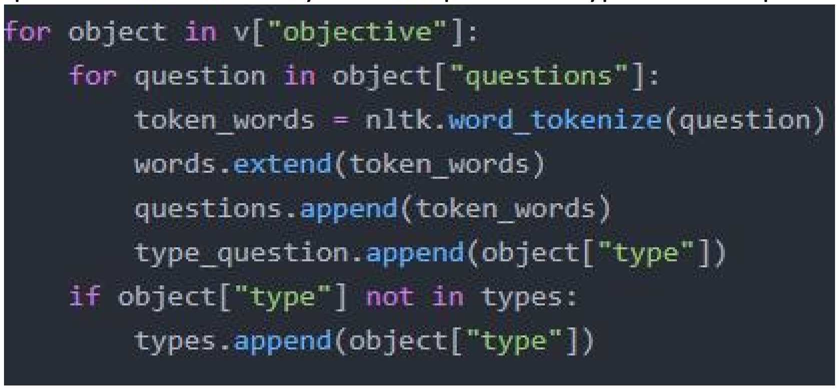
```
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
import numpy
import tflearn
import random
import json
```

Now by using json one need to open their intent file and copy that entire data in a variable named as v



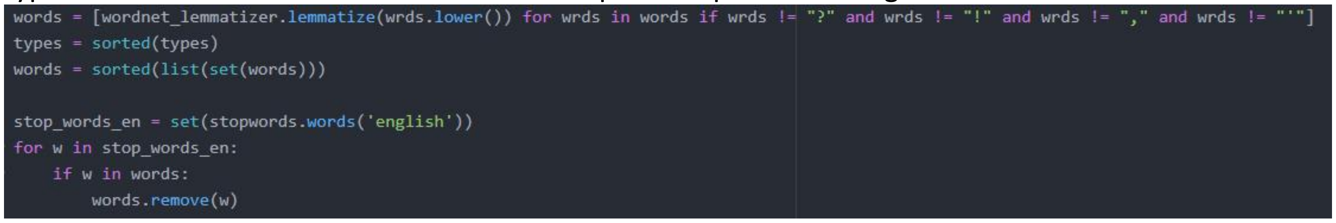
```
with open("intent.json") as dataset:
    v = json.load(dataset)
```

After that one should create four lists they are like words which will store all the words present in all objective questions, types to store all the 15 types that are present in intent.json file like introduction, name, sell, yes, roasted almonds, fruits and nuts, bubbly, oreo, normal silk, interests, timings, thanks, ending, ok and queries. Questions list to store all the questions and finally type_question list to store the corresponding type that the question belongs to. After that one have to do word tokenizing because it will take a question and split it into words and extends it to words list and appends it to questions list and finally note the particular type that the question belongs to in type_question list.



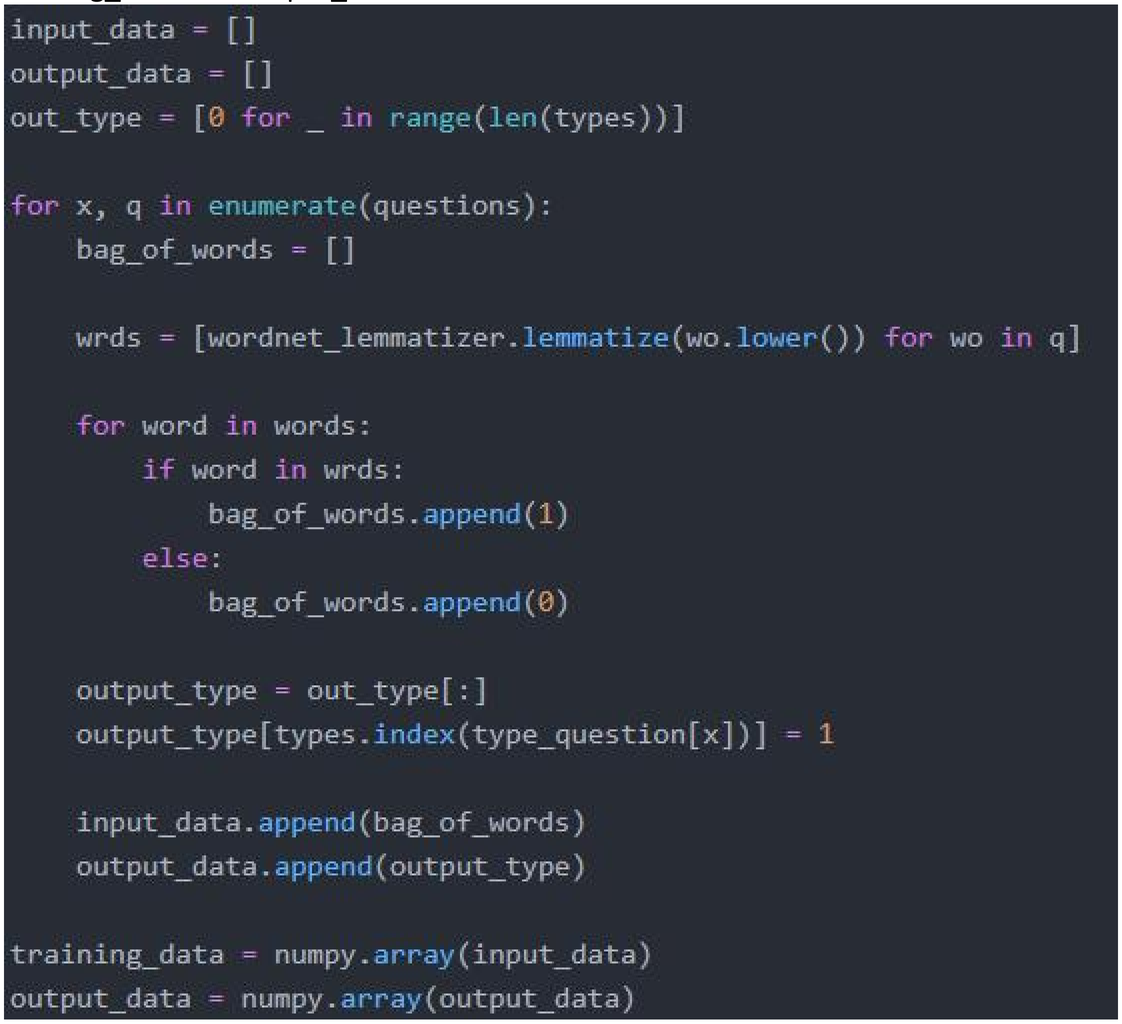
```
for object in v["objective"]:
    for question in object["questions"]:
        token_words = nltk.word_tokenize(question)
        words.extend(token_words)
        questions.append(token_words)
        type_question.append(object["type"])
    if object["type"] not in types:
        types.append(object["type"])
```

Now one should do word lemmatization and exclude question mark, commas, exclamation mark, apostrophe ect. Because they don't need to be considered. After that one have to sort words and types lists. Now It's time to remove all the stop-words present in English from the words list.



```
words = [wordnet_lemmatizer.lemmatize(wrds.lower()) for wrds in words if wrds != "?" and wrds != "!" and wrds != "," and wrds != "'"]
types = sorted(types)
words = sorted(list(set(words)))

stop_words_en = set(stopwords.words('english'))
for w in stop_words_en:
    if w in words:
        words.remove(w)
```

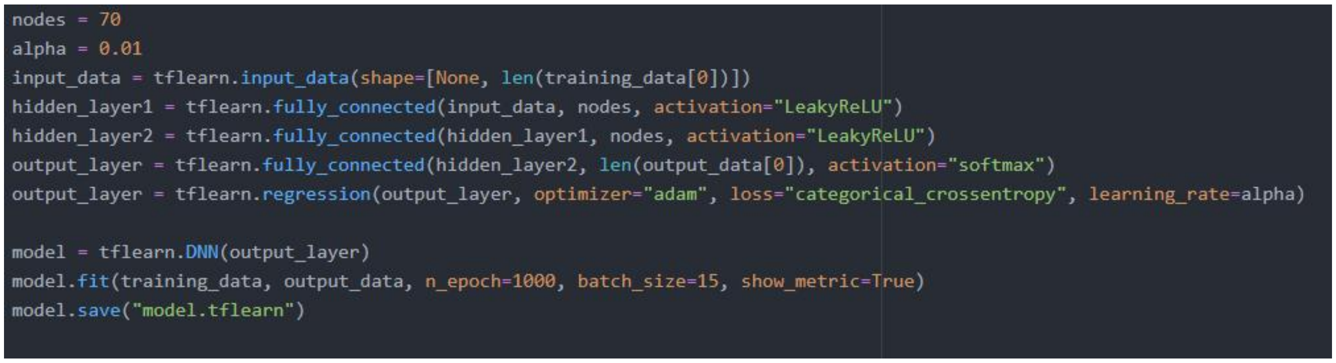
Now one need to use the concept of Bag_of_words basically why they need bag of words in the chatbot code because firstly here we are using Deep Neural Network for this kind of network one can't use words or sentences or letters as input so there is a requirement to convert the available data into numerical form. so what one shell do here means firstly we will create a empty list named as input_data to act as a input nodes to our neural network and output_data list as output of our neural network. Here the input nodes will contain the words it goes with two hidden layers and output nodes will contain all the 15 types. So basically here one have to create a new bag for each and every question then we will check whether a particular word in question sentence is present in the words list then we will append one to the bag_of_words list else append zero to bag_of_words list. Similarly one should do the same thing to the types list. Here we will take some random list which contains all the types and initialize them as zero. After completion of bag appending we will mark the particular question's type as 1 to note the type. Finally one has to append bag list to input_data list and that type list to output_data list and now one need to convert the lists into numpy arrays. Basically like as I said before we can't use lists when we are working with neural networks because numpy data structures take less space as well as faster than lists so one must convert input_data and output_data lists as numpy arrays. Now finally one will get two numpy arrays lets name it as training_data and output_data

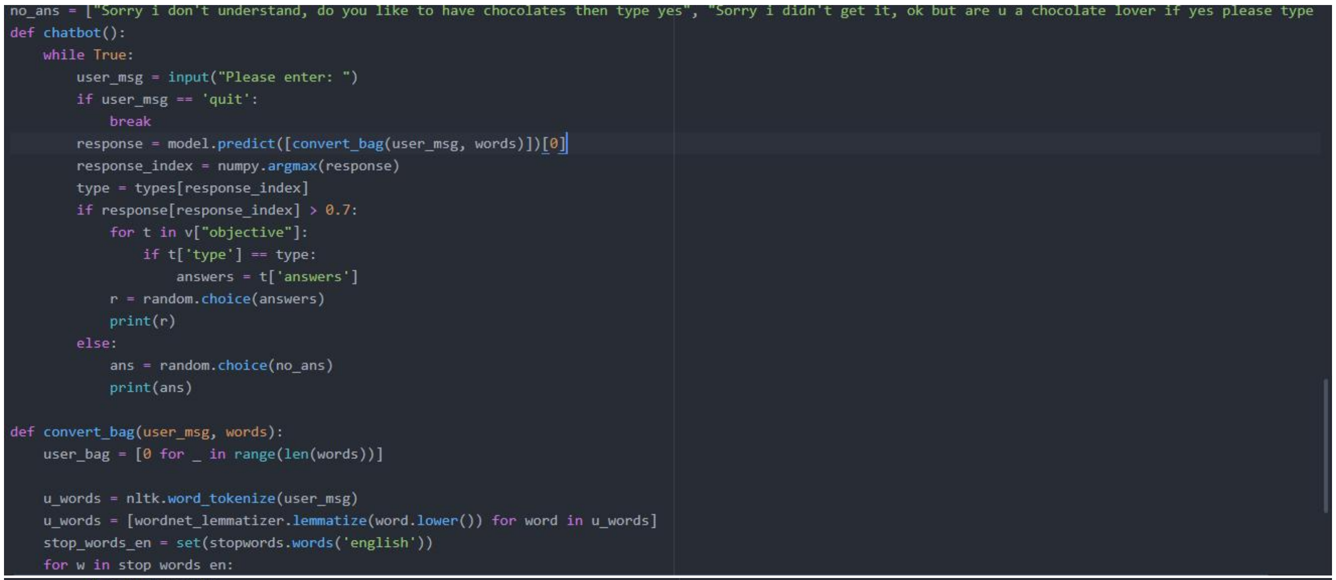


```
input_data = []
output_data = []
out_type = [0 for _ in range(len(types))]

for x, q in enumerate(questions):
    bag_of_words = []

    wrds = [wordnet_lemmatizer.lemmatize(wo.lower()) for wo in q]

    for word in words:
        if word in wrds:
            bag_of_words.append(1)
        else:
            bag_of_words.append(0)

    output_type = out_type[:]
    output_type[types.index(type_question[x])] = 1

    input_data.append(bag_of_words)
    output_data.append(output_type)

training_data = numpy.array(input_data)
output_data = numpy.array(output_data)
```

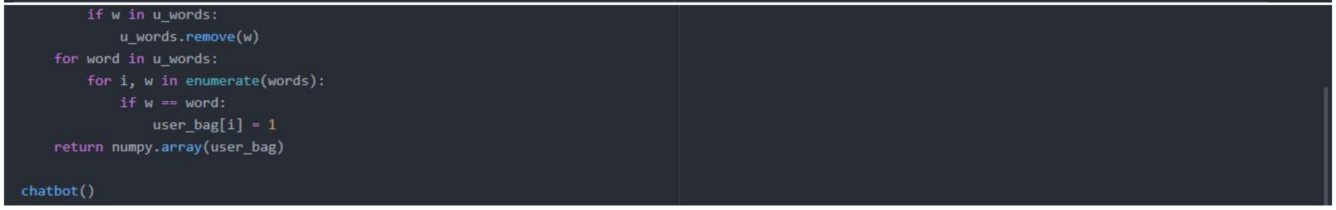
Now its time to create our deep neural network, before going into neural network code that I have wrote firstly lets discuss the basic idea and concept of Deep Neural Networks . Like firstly the difference between neural network and deep neural network is in deep neural network one will have multiple or many hidden layers. Basically in Deep Neural Networks there will be one input layer, hidden layers and output layer. So the input layer nodes will be fully connected to the hidden layer 1 nodes, hidden layer 1 nodes are fully connected to hidden layer 2 nodes … up to hidden layer n nodes are fully connected to output layer. Now one need to randomly initialize weights and biases to their nodes and as we all know those weights and biases are incorrect. Now our target is to make weights and biases as correct as possible so, for that first one have to do forward propagation by getting the weighted sum of inputs and then substituting that value in some activation function that he/she may

use those are like linear, sigmoid, tanh, ReLU, Leaky-ReLU etc. In this way we will do feed-forwarding up to the output layer. Now in output layer one will get some random outputs obviously those are incorrect. After that one should check the output that they got with the correct output from dataset, by this way we can get the loss after getting loss one need to do back propagation to rectify the weights and also to reduce the loss so that our neural networks works accurately. Here to change weights one must use some optimizers like Gradient Descent, Stochastic Gradient Descent, Mini-Batch Gradient Descent, Momentum, Nesterov Accelerated Gradient, Adagard, Adadelta and Adam. But actually Adam optimizer is the best optimizer because it will train the neural network in less time and it is more efficient and in case of Gradient Descent optimizers Mini batch Gradient is best. In this way one can use optimizers and reduce the loss. One more thing while writing neural network code one need to specify the learning rate means the rate at which the model should learn so it is not a fixed value they need to configure it according to no.of hidden layer, no.of nodes in each layer, no.of epochs and the kind of activation and optimizer used. That's the basic concept of Deep Neural Networks
Now lets get into the code that I have wrote for neural networks using tflearn. It's very simple to build a deep neural network using tflearn now lets get into the code firstly one will define their input layer like the number of nodes for input layer is the length of training_data and after that he/she will define a hidden layer 1 with 70 nodes. Why only 70 nodes basically to find how many nodes for a hidden layer one need to follow these three steps
i) Midway between input and output
ii) Less than 2x of input nodes
iii) 2/3*(input nodes)+output nodes
By this way one can get exactly 70 so in my neural network I'm using 70 nodes in each hidden layer. Activation function that one suggested to use in this hidden layer is LeakyReLU because one can't use sigmoid and tanh they are used for binary classification and in case of ReLU there will be a problem of dead nodes so to overcome this LeakyReLU is defined and used. After that one should define hidden layer 2 with 70 nodes and LeakyReLU activation function. Here the hidden layer1 and hidden layer 2 are fully connected. Finally one will define the output layer with 15 nodes and softmax activation function here softmax is the best activation function which is used in case of multiple classes and number of nodes in output layer is the length of output_data. But one is recommended not to use softmax for hidden layers so I used LeakyReLU. After that one should do regression like specifying the optimizer that they are using here we use Adam optimizer, loss function we use that is categorical_crossentropy and finally the learning rate. After all this done one need to specify this is a DNN(Deep Neural Network). Finally one can send the input and output data to their model that they have created as well as need to specify the number of epochs here one epoch is forward+backward here I have defined 1000 epochs. Finally one need to save the model.


```
nodes = 70
alpha = 0.01
input_data = tflearn.input_data(shape=[None, len(training_data[0])])
hidden_layer1 = tflearn.fully_connected(input_data, nodes, activation="LeakyReLU")
hidden_layer2 = tflearn.fully_connected(hidden_layer1, nodes, activation="LeakyReLU")
output_layer = tflearn.fully_connected(hidden_layer2, len(output_data[0]), activation="softmax")
output_layer = tflearn.regression(output_layer, optimizer="adam", loss="categorical_crossentropy", learning_rate=alpha)

model = tflearn.DNN(output_layer)
model.fit(training_data, output_data, n_epoch=1000, batch_size=15, show_metric=True)
model.save("model.tflearn")
```
Now we are at the final part of our chatbot code like here I will define a chatbot function In this function firstly one should take the user input by using input() function. Basically at last one should call the chatbot() function we run it in command prompt in case of chatting if the user want to exit from chat then he/she need to type something like lets say for example 'quit'. Hence if the user types quit then the chatbot() function execution should be stopped. Even in this function we will take the user message as the parameter and after that it will send that message to a convert_bag function which will first tokenize the sentence and store the words in u_words list, lemmatize each and every word present in u_words list and finally removes the stop_words from that list. Now it will convert the user data into bag of words means as numeric form, converts it into numpy array and finally returns it. Now one should call model.predict() by passing returned numpy array as parameter and finally it will predict the output. Now in output layer as I said before there will be 15 types it will show the probability of those 15 types here one should select the highest probable type so we take most probable type from the model. Now check whether that probability is greater then or equal to 0.7. If it is greater or equal to 0.7 one should proceed this like they will get into that most probable type and return some random answer from a answers list that belongs to that particular type. If it is less than 0.7 the bot should print or display that It doesn't understood the question. one should initially define a list of responses that the bot should give when the probability is less than 0.7 and the program will select a random statement from that list and finally print it.

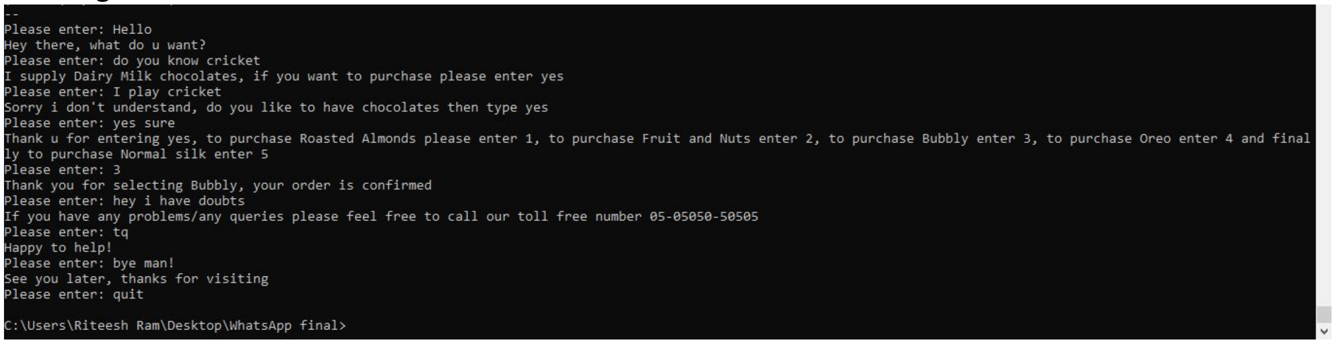


```python
no_ans = ["Sorry i don't understand, do you like to have chocolates then type yes", "Sorry i didn't get it, ok but are u a chocolate lover if yes please type
def chatbot():
    while True:
        user_msg = input("Please enter: ")
        if user_msg == 'quit':
            break
        response = model.predict([convert_bag(user_msg, words)])[0]
        response_index = numpy.argmax(response)
        type = types[response_index]
        if response[response_index] > 0.7:
            for t in v["objective"]:
                if t['type'] == type:
                    answers = t['answers']
            r = random.choice(answers)
            print(r)
        else:
            ans = random.choice(no_ans)
            print(ans)

def convert_bag(user_msg, words):
    user_bag = [0 for _ in range(len(words))]

    u_words = nltk.word_tokenize(user_msg)
    u_words = [wordnet_lemmatizer.lemmatize(word.lower()) for word in u_words]
    stop_words_en = set(stopwords.words('english'))
    for w in stop_words_en:
```



```python
        if w in u_words:
            u_words.remove(w)
    for word in u_words:
        for i, w in enumerate(words):
            if w == word:
                user_bag[i] = 1
    return numpy.array(user_bag)

chatbot()
```

This the chatbot python code. When one followed the above mentioned procedure then her/she can create a chatbot that will perfectly run on command prompt. The below shown conversation is my chatting with the bot that I have created.



```
Please enter: Hello
Hey there, what do u want?
Please enter: do you know cricket
I supply Dairy Milk chocolates, if you want to purchase please enter yes
Please enter: I play cricket
Sorry i don't understand, do you like to have chocolates then type yes
Please enter: yes sure
Thank u for entering yes, to purchase Roasted Almonds please enter 1, to purchase Fruit and Nuts enter 2, to purchase Bubbly enter 3, to purchase Oreo enter 4 and final
ly to purchase Normal silk enter 5
Please enter: 3
Thank you for selecting Bubbly, your order is confirmed
Please enter: hey i have doubts
If you have any problems/any queries please feel free to call our toll free number 05-05050-50505
Please enter: tq
Happy to help!
Please enter: bye man!
See you later, thanks for visiting
Please enter: quit

C:\Users\Riteesh Ram\Desktop\WhatsApp final>
```

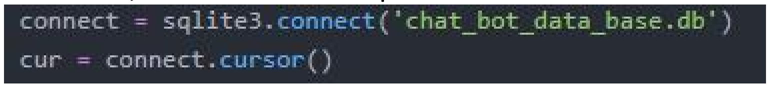
Here when I entered quit the code will exit from chatbot(). Finally the program will stop. Now only one thing is left that is to integrate my chatbot python code with flask application that I have created for echo bot.

How to integrate the chatbot that is running on command prompt with WhatsApp:

Here we are done with creating an echo WhatsApp chatbot and also done with writing a python chatbot code. Now to integrate the chatbot code with WhatsApp one need to do some changes to the flask application and chatbot code. In this section we are going to create a database to store all the user and bot messages. Now firstly lets see what changes that one should do to the flask application. Here one have to import all the required python packages/libraries that will be used in chatbot code even he/she doesn’t need them in flask application code. Similarly need to import SQLite3 as well to create a database using python. Now lets discuss why and what exactly one need to use SQLite3 in this flask app. This SQLite3 is the mini version of SQL database this library is an inbuilt in python 3. The advantages of using SQLite3 is it is lightweight, no need to setup any server as well as user and also no permissions required. Let me explain what exactly I did with sqlite3 first one have to import sqlite3 to their flask app code.

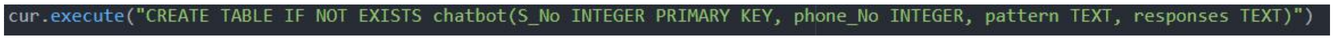
```
import sqlite3
```

After that they need to connect to their database before they start working on it. For that one should use sqlite3.connect() function for that function passed one’s database name as a parameter if the database is not created then this function will automatically create the database and connects to it. After connecting one can define their respective cursor, a cursor is a thing which will be used to point the things and also we need cursor when we want to execute the commands like create a table, insert new rows, delete rows and update some rows.



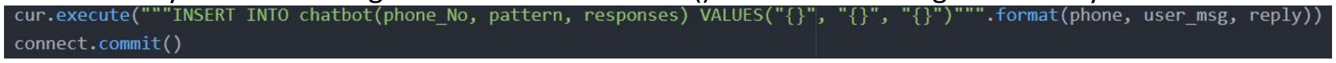
```
connect = sqlite3.connect('chat_bot_data_base.db')
cur = connect.cursor()
```

Now one have to create a table to store the data of the chat by user as well as bot. In that table one should create four columns S_No which is of INTEGER type and it is a primary key of that table, phone_No which is of BLOB or INTEGER type it will store the user mobile number, pattern which is of TXET type it will store all the user messages and finally responses column which is of TEXT type and stores bot responses.



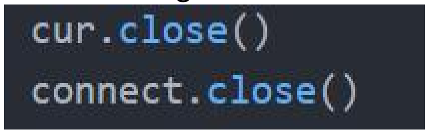
```
cur.execute("CREATE TABLE IF NOT EXISTS chatbot(S_No INTEGER PRIMARY KEY, phone_No INTEGER, pattern TEXT, responses TEXT)")
```

Here the table chatbot will be created if that table is not exists/present in chat_bot_data_base.db. Now one should insert the user mobile/phone number, user message and bot replies in the form of rows to this table. To do that firstly they should know the user phone number, user and bot messages and have to execute insert into table function. One more thing there is no need to manually insert S_No in the table because, S_No is a primary key of integer type so it will be incremented automatically. After inserting one should call commit() to save the changes that they have made.



```
cur.execute("""INSERT INTO chatbot(phone_No, pattern, responses) VALUES("{}", "{}", "{}")""".format(phone, user_msg, reply))
connect.commit()
```

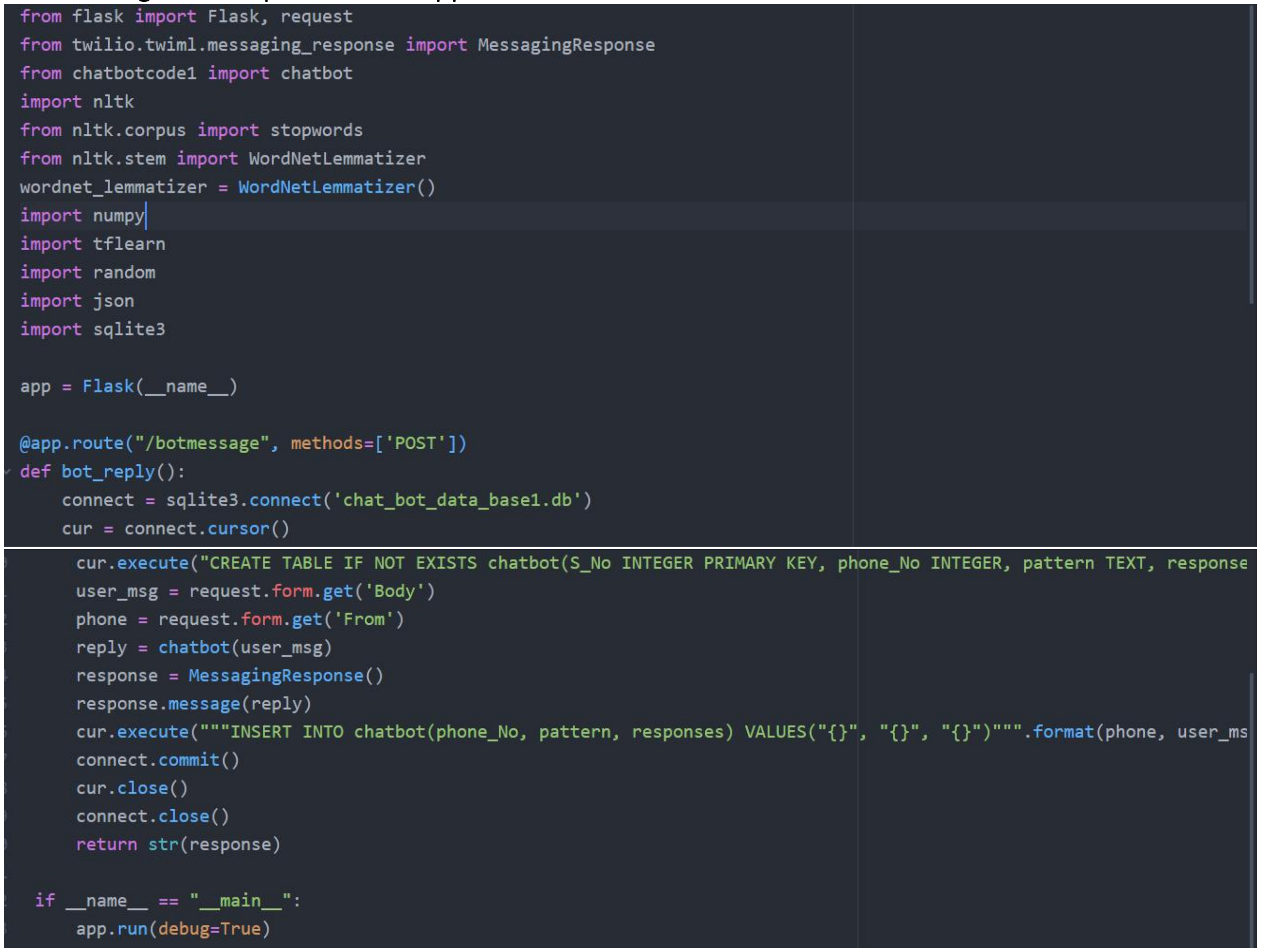
After inserting one can close the cursor as well as connection by using close().



```
cur.close()
connect.close()
```

To run and watch ones created database with SQLite3 they have to install DB Browser for SQLite. Hence these are the changes that one have to make to create and use the database in flask app. In case of echo bot basically the flask app fetched the user message and it returned the same message as a reply. In case of chatbot the bot should chat with the user. To make that working one should edit the flask app code, like basically the chatbot() from python chatbot code will return the bot reply to the particular user message.

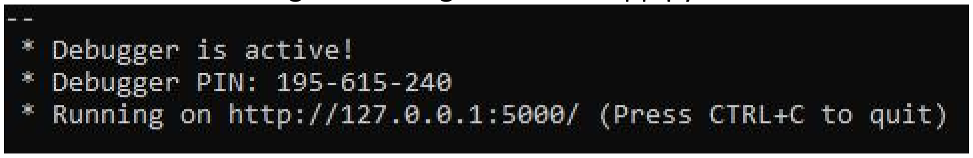
Here this flask app should take that value and let's say store it in some variable like reply and this reply should be passed as a parameter to response.message() function. To do that firstly one should import chatbot() from chatbot code in flask app. Also in case of database creation we need the user phone number too. To get phone number one should call request.form.get() function by sending From as a parameter so that we can store the number in phone variable and move further. The below shown image is the updated flask app from echo bot to chatbot.



```
from flask import Flask, request
from twilio.twiml.messaging_response import MessagingResponse
from chatbotcode1 import chatbot
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
import numpy
import tflearn
import random
import json
import sqlite3

app = Flask(__name__)

@app.route("/botmessage", methods=['POST'])
def bot_reply():
    connect = sqlite3.connect('chat_bot_data_base1.db')
    cur = connect.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS chatbot(S_No INTEGER PRIMARY KEY, phone_No INTEGER, pattern TEXT, response
    user_msg = request.form.get('Body')
    phone = request.form.get('From')
    reply = chatbot(user_msg)
    response = MessagingResponse()
    response.message(reply)
    cur.execute("""INSERT INTO chatbot(phone_No, pattern, responses) VALUES("{}", "{}", "{}")""".format(phone, user_ms
    connect.commit()
    cur.close()
    connect.close()
    return str(response)

if __name__ == "__main__":
    app.run(debug=True)
```

Now we are done with creating a chatbot that will run on WhatsApp. Now to integrate it with WhatsApp firstly one should run the updated flask app in command prompt. By running the flask app in command prompt the model will be trained automatically. Means no need to execute the chatbot code again this app will execute it and if any errors or warnings are encountered in chatbot code it will display them on command prompt itself.

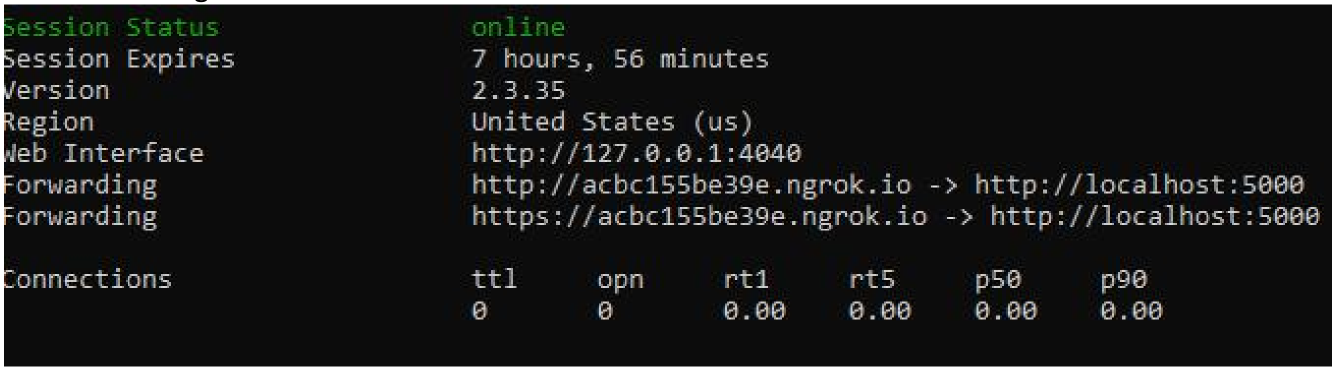
The below shown image is the ting when I run app.py in cmd.



```
--
 * Debugger is active!
 * Debugger PIN: 195-615-240
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

To test whether the bot is working on WhatsApp or not, one can use ngrok which will generate the public URL for some particular session time. To use ngrok one should run ngrok http 5000 command on command prompt.

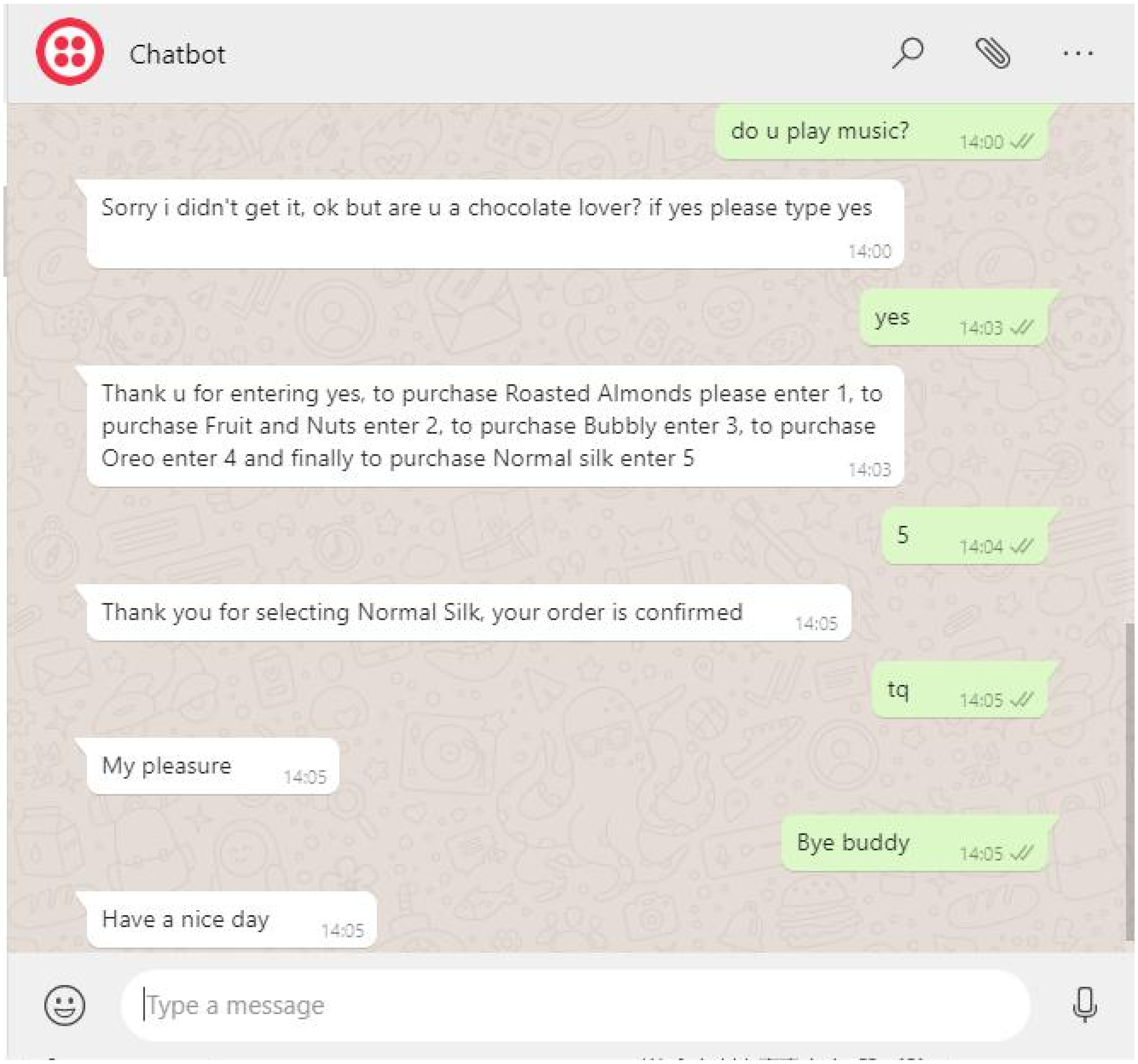
The URL that I got is



```
Session Status                online
Session Expires               7 hours, 56 minutes
Version                       2.3.35
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://acbc155be39e.ngrok.io -> http://localhost:5000
Forwarding                    https://acbc155be39e.ngrok.io -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Now paste the URL in Twilio sandbox and also add /botmessage to the end of the URL. Because by adding /botmessage at the end it will call bot_reply() function present in flask app. Then the final URL will look like: https://acbc155be39e.ngrok.io/botmessage.

The below shown image is the WhatsApp chat that I have done with the chatbot.



do u play music? 14:00

Sorry i didn't get it, ok but are u a chocolate lover? if yes please type yes 14:00

yes 14:03

Thank u for entering yes, to purchase Roasted Almonds please enter 1, to purchase Fruit and Nuts enter 2, to purchase Bubbly enter 3, to purchase Oreo enter 4 and finally to purchase Normal silk enter 5 14:03

5 14:04

Thank you for selecting Normal Silk, your order is confirmed 14:05

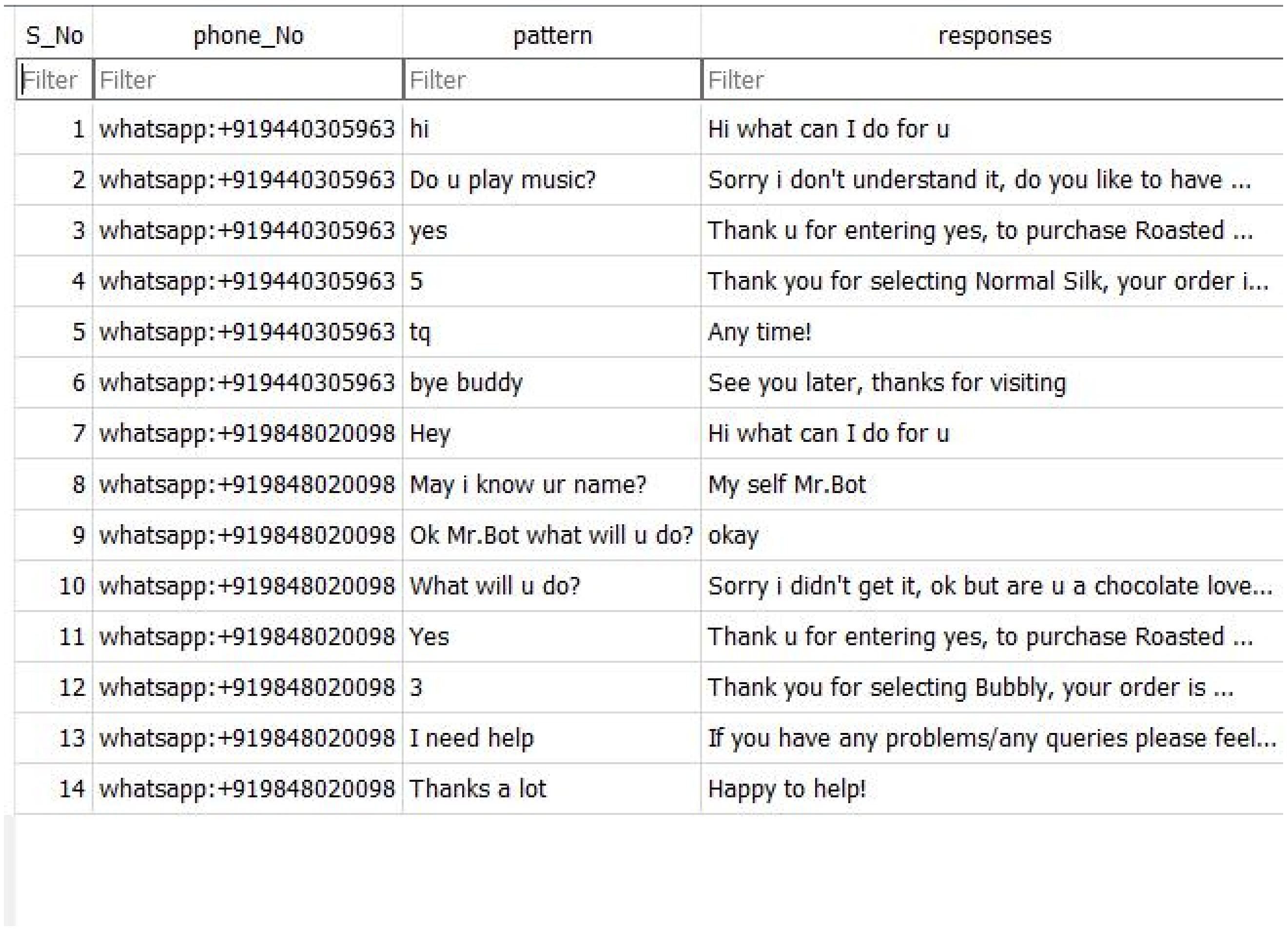
tq 14:05

My pleasure 14:05

Bye buddy 14:05

Have a nice day 14:05

The below shown image is the table named as chatbot that I have created in my database named as chat_bot_data_base.db. But one can open the database in DB Browser and in Browse Data section they can view their table.

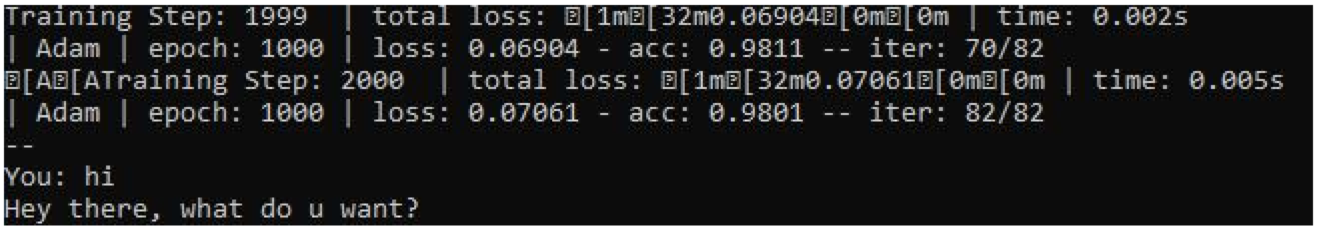


S_No	phone_No	pattern	responses
Filter	Filter	Filter	Filter
1	whatsapp:+919440305963	hi	Hi what can I do for u
2	whatsapp:+919440305963	Do u play music?	Sorry i don't understand it, do you like to have ...
3	whatsapp:+919440305963	yes	Thank u for entering yes, to purchase Roasted ...
4	whatsapp:+919440305963	5	Thank you for selecting Normal Silk, your order i...
5	whatsapp:+919440305963	tq	Any time!
6	whatsapp:+919440305963	bye buddy	See you later, thanks for visiting
7	whatsapp:+919848020098	Hey	Hi what can I do for u
8	whatsapp:+919848020098	May i know ur name?	My self Mr.Bot
9	whatsapp:+919848020098	Ok Mr.Bot what will u do?	okay
10	whatsapp:+919848020098	What will u do?	Sorry i didn't get it, ok but are u a chocolate love...
11	whatsapp:+919848020098	Yes	Thank u for entering yes, to purchase Roasted ...
12	whatsapp:+919848020098	3	Thank you for selecting Bubbly, your order is ...
13	whatsapp:+919848020098	I need help	If you have any problems/any queries please feel...
14	whatsapp:+919848020098	Thanks a lot	Happy to help!

Finally I'm done with creating WhatsApp Business chatbot.

Results/Discussion:

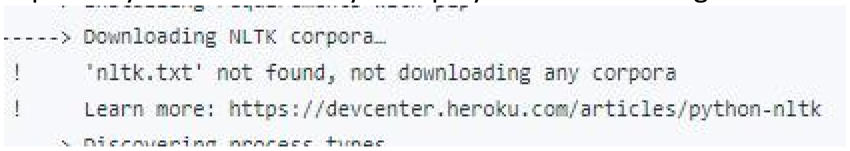
In results by following the above mentioned procedures one can build echo WhatsApp bot, chatbot that runs on command prompt and finally WhatsApp chatbot. lets discuss some thing about the accuracy, loss that one will get when running the chatbot code in command prompt like as I said before I trained my model for 1000 epochs.



```
Training Step: 1999  | total loss: [1m[32m0.06904[0m[0m | time: 0.002s
| Adam | epoch: 1000 | loss: 0.06904 - acc: 0.9811 -- iter: 70/82
[A[ATraining Step: 2000  | total loss: [1m[32m0.07061[0m[0m | time: 0.005s
| Adam | epoch: 1000 | loss: 0.07061 - acc: 0.9801 -- iter: 82/82
--
You: hi
Hey there, what do u want?
```

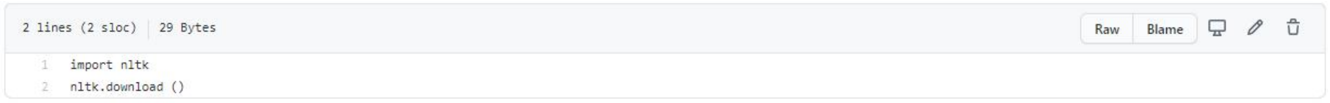
Here the loss that I got is 0.07061 and accuracy is 0.9801. So according to my perception my model is working fine. Because as the loss decreases the models accuracy and intelligence increases. Hence this model is efficient. In methodology section I have discussed about Twilio and its use in this project now lets see how exactly the chatbot code, flask app, Twilio and WhatsApp will work internally. Lets take an example there are two persons A, B here person A will send a WhatsApp message to B then that message will be forwarded to WhatsApp server the server will check the to address and sends message to B. Now we are replacing person B with the available server application this server application is our flask app and chatbot code so this server application will fetch the person A message and create a response and sends reply to WhatsApp server. Now in between server application and WhatsApp server Twilio sandbox will be present so basically WhatsApp sever will forward the person A message to Twilio and that Twilio will invoke the sandbox. Finally the URL present in sandbox will invoke server application. In this way the WhatsApp chatbot works internally. Also in case deep neural networks I used Leaky-ReLU as an activation function for my hidden layers lets discuss more detailed explanation about this firstly in case of sigmoid there will be a problem of vanishing gradient decent. To over come this there is tanh activation function this function will give the derivative values from -1 to 1, this function will avoid the gradient vanishing but if the values are large then this activation function also vanishes the gradient. Now in case of ReLU there will be a problem of dead nodes like if the weights are negative then there respective derivatives will become zero so one can't change the weights if they are negative. To overcome this one should use Leaky-ReLU activation function. Also I can't use softmax activation function for hidden layers so I used Leaky-ReLU. Why I used softmax activation function in output layer is because in our case we have multiple classes so it's recommended to use softmax activation. Because in case of softmax it will give the probabilities of all the output nodes which will sum up to 1. By this way one can select the highest probable node. In case of optimizers I used Adam optimizer. Which is the best and efficient optimizer in neural networks.

Here in this project I designed this chatbot for a company which will sell Cadbury Dairy Milk Silk chocolates. This bot will chat with their customers normally as we do, places the orders, address their queries as well. One more thing in case of echo bot I deployed flask app on cloud application like Heroku but in case of WhatsApp chatbot with chatbot code and updated flask app I haven't deployed it on cloud. Why I did so? Because while deploying the flask app that is present in my github repository on Heroku firstly it displayed some warnings like



```
-----> Downloading NLTK corpora…
 !     'nltk.txt' not found, not downloading any corpora
 !     Learn more: https://devcenter.heroku.com/articles/python-nltk
```

Here finally the flask app was deployed an also I got some URL, I pasted it on Twilio sandbox, at last I added /botmessage to the URL and finally saved it. But when I started chatting with the bot Through WhatsApp I doesn't received any reply from bot, when I checked the logs in Heroku it showed that the app is crashed. Now as it mentioned nltk.txt is not found in github repository I created a nltk.txt file in my repository and typed this commands in it.



```
2 lines (2 sloc) | 29 Bytes

import nltk
nltk.download ()
```

Finally I saved it and started deploying flask app again in Heroku cloud then it showed trace back errors regarding the python files that are initially prebuilt in nltk. Those errors are not from my code its from prebuilt nltk python codes.

So the only improvement that one can do in this project is to deploy the flask app on cloud platforms. The only difference between Ngrok and cloud is that by using Ngrok one have to change the URL for every 8hrs because the URL session will expire after 8hrs, in case of cloud ones the deployment is done perfectly then they can use the WhatsApp chatbot 24/7. One more with Twilio API multiple users can use the WhatsApp chatbot that I have created from different WhatsApp accounts. They only need to send my Twilio WhatsApp sandbox name that is join fur-rain as a message to this number +1 415 523 8886 from WhatsApp.

Conclusion:

In conclusion we have discussed the main objective of the project like why one need to build a WhatsApp Business chatbot. Discussed the problem that large and medium businesses face’s without a chatbot. Finally we have gone through the methodology that how to create an echo bot in WhatsApp, python chatbot code that perfectly runs on command prompt and finally WhatsApp Business chatbot . Firstly we will write a python code to create a chatbot in that code I used the concepts of natural language processing, bag of words and deep neural networks. Then I created a flask application which will take a message from the user and fetches it and get the reply from our chatbot code and finally sends to Twilio. Also discussed how to integrate flask application on Heroku cloud platform. Discussed integrating the flask app on Twilio through some URL. The bot worked fine in WhatsApp I shared the screenshot of that in methodology section. Finally I created WhatsApp Business chatbot.

Appendices:

Echo bot flask app code:

```
from flask import Flask, request
from twilio.twiml.messaging_response import MessagingResponse

app = Flask(__name__)

@app.route("/botmessage", methods=['POST'])
def bot_reply():
      user_msg = request.form.get('Body')
      response = MessagingResponse()
      response.message(user_msg)
      return str(response)

if __name__ == "__main__":
    app.run(debug=True)
```

Intent.json file:

```
{"objective": [
       {"type": "introduction",
        "questions": ["Hi", "Hello", "Is any one there?", "Good morning", "Good evening", "Good
afternoon", "hey", "Excuse me?", "Whats up?"],
        "answers": ["Hey there, what do u want?", "Hi what can I do for u", "Hey buddy what do you
need?"]
       },
       {"type": "name",
         "questions": ["What's your name?", "May i know your name?", "name please", "What should i
call you", "your name?", "Who are you?", "who the hell are you"],
         "answers": ["My self Mr.Bot", "My name is Mr.Bot", "I am Mr.Bot"]
       },
       {"type": "sell",
        "questions": ["What do you sell?", "What's your products?", "May i know the items that you
sell?", "I want to buy some thing", "What do you have?", "Want to buy", "Want to purchase", "Any
items", "products", "items", "What will you do?", "What are you supposed to do?", "may i know what
are you for?", "What you do?"],
        "answers": ["I sell Dairy Milk silk chocolates if you want to purchase please enter yes", "I supply
Dairy Milk chocolates, if you want to purchase please enter yes"]
       },
       {"type": "yes",
        "questions": ["Yes", "Yes I want", "Yes please", "Yes I do purchase"],
        "answers": ["Thank u for entering yes, to purchase Roasted Almonds please enter 1, to purchase
Fruit and Nuts enter 2, to purchase Bubbly enter 3, to purchase Oreo enter 4 and finally to purchase
Normal silk enter 5"]
       },
       {"type": "roasted almonds",
        "questions": ["1"],
        "answers": ["Thank you for selecting Roasted Almonds, your order is confirmed."]
       },
       {"type": "fruits and nuts",
        "questions": ["2"],
        "answers": ["Thank you for selecting Fruit and Nuts, your order is confirmed"]
       },
       {"type": "bubbly",
        "questions": ["3"],
        "answers": ["Thank you for selecting Bubbly, your order is confirmed"]
       },
       {"type": "oreo",
        "questions": ["4"],
        "answers": ["Thank you for selecting Oreo, your order is confirmed"]
       },
       {"type": "normal silk",
        "questions": ["5"],
        "answers": ["Thank you for selecting Normal Silk, your order is confirmed"]
       },
       {"type": "interest",
         "questions": ["not interested", "sorry no interest", "No", "No interest", "No need"],
         "answers": ["Ok ok I think you are not a chocolate lover if you have some time please visit our
website to know about our company and products."]
       },
       {"type": "timings",
        "questions": ["What hours are you open?", "What are your hours?", "When are you open?",
"Timings please", "Working hours", "working time", "time", "hours"],
        "answers": ["I am open 24/7", "my timings is 24/7"]
       },
       {"type": "thanks",
        "questions": ["Thanks", "Thank you", "That's helpful", "Tq", "may thanks", "That's great",
"great"],
        "answers": ["Happy to help!", "Any time!", "My pleasure"]
       },
       {"type": "ending",
        "questions": ["See you again", "Bye", "Goodbye", "See you later", "Okay bye"],
        "answers": ["See you later, thanks for visiting", "Have a nice day", "Bye! Come back again soon."]
       },
       {"type": "ok",
         "questions": ["ok", "okay", "ok ok"],
         "answers": ["ok", "okay"]
       },
       {"type": "queries",
        "questions": ["not understood", "didn't understand", "I don't understand", "I need help", "Could
you please help me?", "I am facing issues", "I have some queries to ask", " I have a problem", "issues",
"need help", "help me", "problems", "queries", "Have some questions", "Have questions", "Have
doubts", "I have some doubts"],
        "answers": ["If you have any problems/any queries please feel free to call our toll free number
05-05050-50505 "]
       }
     ]
}
```

Chatbot code:

```
import json
import tflearn
import random
import numpy
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

with open("intent.json") as dataset:
   v = json.load(dataset)

words = []
types = []
questions = []
```

```
type_question = []

for object in v["objective"]:
    for question in object["questions"]:
        token_words = nltk.word_tokenize(question)
        words.extend(token_words)
        questions.append(token_words)
        type_question.append(object["type"])
    if object["type"] not in types:
        types.append(object["type"])

words = [wordnet_lemmatizer.lemmatize(wrds.lower()) for wrds in words if wrds != "?" and wrds !=
"!" and wrds != "," and wrds != "'"]
types = sorted(types)
words = sorted(list(set(words)))

stop_words_en = set(stopwords.words('english'))
for w in stop_words_en:
    if w in words:
        words.remove(w)

input_data = []
output_data = []
out_type = [0 for _ in range(len(types))]

for x, q in enumerate(questions):
    bag_of_words = []

    wrds = [wordnet_lemmatizer.lemmatize(wo.lower()) for wo in q]

    for word in words:
        if word in wrds:
            bag_of_words.append(1)
        else:
            bag_of_words.append(0)

    output_type = out_type[:]
    output_type[types.index(type_question[x])] = 1

    input_data.append(bag_of_words)
    output_data.append(output_type)

training_data = numpy.array(input_data)
output_data = numpy.array(output_data)

nodes = 70
alpha = 0.01
input_data = tflearn.input_data(shape=[None, len(training_data[0])])
hidden_layer1 = tflearn.fully_connected(input_data, nodes, activation="LeakyReLU")
hidden_layer2 = tflearn.fully_connected(hidden_layer1, nodes, activation="LeakyReLU")
output_layer = tflearn.fully_connected(hidden_layer2, len(output_data[0]), activation="softmax")
output_layer = tflearn.regression(output_layer, optimizer="adam", loss="categorical_crossentropy",
learning_rate=alpha)

model = tflearn.DNN(output_layer)
model.fit(training_data, output_data, n_epoch=1000, batch_size=15, show_metric=True)
model.save("model.tflearn")
no_ans = ["Sorry i don't understand it, do you like to have chocolates? then type yes", "Sorry i didn't
get it, ok but are u a chocolate lover? if yes please type yes"]
def chatbot(user_msg):
    response = model.predict([convert_bag(user_msg, words)])[0]
    response_index = numpy.argmax(response)
    type = types[response_index]
    if response[response_index] > 0.7:
        for t in v["objective"]:
            if t['type'] == type:
                answers = t['answers']
        r = random.choice(answers)
        return r
    else:
        ans = random.choice(no_ans)
        return ans

def convert_bag(user_msg, words):
    user_bag = [0 for _ in range(len(words))]

    u_words = nltk.word_tokenize(user_msg)
    u_words = [wordnet_lemmatizer.lemmatize(word.lower()) for word in u_words]
    stop_words_en = set(stopwords.words('english'))
    for w in stop_words_en:
        if w in u_words:
            u_words.remove(w)
    for word in u_words:
        for i, w in enumerate(words):
            if w == word:
                user_bag[i] = 1
    return numpy.array(user_bag)
```

Flask app updated code with database:

```
from flask import Flask, request
from twilio.twiml.messaging_response import MessagingResponse
from chatbotcode1 import chatbot
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
import numpy
import tflearn
import random
import json
import sqlite3

app = Flask(__name__)

@app.route("/botmessage", methods=['POST'])
def bot_reply():
    connect = sqlite3.connect('chat_bot_data_base.db')
    cur = connect.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS chatbot(S_No INTEGER PRIMARY KEY, phone_No
INTEGER, pattern TEXT, responses TEXT)")
    user_msg = request.form.get('Body')
    phone = request.form.get('From')
    reply = chatbot(user_msg)
    response = MessagingResponse()
    response.message(reply)
    cur.execute("""INSERT INTO chatbot(phone_No, pattern, responses) VALUES("{}", "{}",
"{}")""".format(phone, user_msg, reply))
    connect.commit()
    cur.close()
```

```
  connect.close()
   return str(response)

if __name__ == "__main__":
   app.run(debug=True)
```

References:

http://tflearn.org/
https://www.tensorflow.org/
https://developers.facebook.com/docs/whatsapp/
https://www.twilio.com/blog/build-a-whatsapp-chatbot-with-python-flask-and-twilio
https://www.guru99.com/nltk-tutorial.html
https://www.tutorialspoint.com/python_text_processing/python_remove_stopwords.htm
https://www.tutorialspoint.com/flask/index.htm
https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
https://medium.com/@sdoshi579/optimizers-for-training-neural-network-59450d71caf6
https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e
https://github.com/nikhilkumarsingh/WhatsAppBotTut