## Project 3: Create a chatbot in Python.

**Steps to create a chatbot in python :**

### **Step-1: Connecting with Google Drive Files and Folders**

The first step is to create a folder by the name 'ChatBot' in your Google drive. After that, upload the 'intents.json' file in this folder and write the following code in a Google Colab notebook: This will allow us to access the files that are there in Google Drive.

### **Step-2: Importing Relevant Libraries**

The next step is the usual one where we will import the relevant libraries, the significance of which will become evident as we proceed.

### **Step-3: Reading the JSON file**

It's time to load our data into Python. We will use json.loads() for this and store the content of the file into a variable named 'data' (type: dict). So, as you can see, the dataset has an object called intents. The dataset has about 16 instances of intents, each having its own tag, context, patterns, and responses. If you thoroughly go through your dataset, you’ll understand that patterns are similar to the interactive statements that we expect from our users whereas responses are the replies to those statements. And, a tag is a one-word summary of the user’s query.

Now, the task at hand is to make our machine learn the pattern between patterns and tags so that when the user enters a statement, it can identify the appropriate tag and give one of the responses as output. And, the following steps will guide you on how to complete this task.

### **Step-4: Identifying Feature and Target for the NLP Model**

Now, we will extract words from patterns and the corresponding tag to them. This has been achieved by iterating over each pattern using a nested for loop and tokenizing it using [nltk.word\_tokenize](http://www.nltk.org/api/nltk.tokenize.html?highlight=word_tokenize" \l "module-nltk.tokenize" \o "nltk.word_tokenize" \t "_blank). The words have been stored in data\_X and the corresponding tag to it has been stored in data\_Y**.**

Two more lists: words and classes containing all tokens and corresponding tags have also been created. For the list words, the punctuations have not been added by using a simple conditional statement and the words have been converted into their root words using NLTK's [WordNetLemmatizer()](https://www.nltk.org/api/nltk.stem.html" \l "nltk.stem.wordnet.WordNetLemmatizer" \o "WordNetLemmatizer()" \t "_blank). This is an important step when writing a chatbot in Python as it will save us a lot of time when we will feed these words to our deep learning model. At last, both the lists have been sorted and these functions have been used to remove any duplicates.

### **Step-5: Making the data Machine-friendly**

In this step, we will convert our text into numbers using the bag-of-words (bow) model. Those who are aware of this must-have now understood the significance of the listed words and classes. And if you are not, don’t worry, we got your back. ðŸ˜‰

The two lists words and classes act as a vocabulary for patterns and tags respectively. We’ll use them to create an array of numbers of size the same as the length of vocabulary lists. The array will have values 1 if the word is present in the pattern/tag being read (from data\_X) and 0 if it is absent.   
The data has thus been converted into numbers and stored in two arrays: train\_X and train\_Y where the former represents features and the latter represents target variables.

### **Step-6: Building the Neural Network Model**

Next, we will create a neural network using Keras’ Sequential model. The input to this network will be the array train\_X created in the previous step. These would then traverse through the model of 3 different layers with the first having 128 neurons, the second having 64 neurons, and the last layer having the same number of neurons as the length of one element of train\_Y (Obvious, right?). Next, to reach the correct weights, we have chosen the Adam optimizer and defined our error function using the categorical cross-entropy function. And, the metric we have chosen is accuracy.

### **Step-7: Pre-processing the User’s Input**

In this step of the tutorial on how to build a chatbot in Python, we will create a few easy functions that will convert the user’s input query to arrays and predict the relevant tag for it. Our code for the Python Chatbot will then allow the machine to pick one of the responses corresponding to that tag and submit it as output.

### **Step-8: Calling the Relevant Functions and interacting with the ChatBot**

We now just have to take the input from the user and call the previously defined functions. Now, you can play around with your ChatBot as much as you want. To improve its responses, try to edit your intents.json [here](https://jsoneditoronline.org/#left=local.boxowu&right=local.daxeyo) and add more instances of intents and responses in it.

**Implementation**

Before starting the coding part of our chatbot development, let’s create a virtual environment for the chatbot. The python library that we are using to create the virtual environment is “virtualenv”.So first of all let’s install virtualenv(In the command prompt)

Now we can create our virtual environment named my\_env, so take the terminal in the vscode or any code editor and write the below code.

virtualenv my\_env

Next is to activate our virtual environment

Activation in windows power shell

my\_envScriptsactivate.ps1

Activation in command prompt

my\_envScriptsactivate.bat

The virtual environment is activated.

**Installation of Libraries**

Now we have to install the libraries required for this project separately in this environment.

pip install keras nltk tensorflow

Now Let’s start to create a machine learning model which can respond to the user query based on the intents file.

Importing some of the required libraries for our project.

import numpy as np

import nltk

import json

import pickle

import re

import random

from keras.models import Sequential

from keras.layers import Dense, Activation, Dropout

from tensorflow.keras.optimizers import SGD

from nltk.stem import WordNetLemmatizer

We have to download some nltk packages for processing the data.

nltk.download('punkt')

nltk.download('wordnet')

**Testing the Model**

Now let’s take another python file for testing and creating our actual chatbot

Importing the required libraries.

import pickle

import numpy as np

import json

from keras.models import load\_model

import random

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

Next is to load our models and pickle files that we are saved during the training timing

intents\_file = json.loads(open('intents.json').read())

lem\_words = pickle.load(open('lem\_words.pkl','rb'))

classes = pickle.load(open('classes.pkl','rb'))

bot\_model = load\_model('chatbot\_model.h5')

Creating a function that takes the user input as a parameter for performing some preprocessing techniques like tokenization and stemming.

def cleaning(text):

words = nltk.word\_tokenize(text)

words = [lemmatizer.lemmatize(word.lower()) for word in words]

return words

Our model requires numerical features for the prediction of classes, so we are creating another function for creating the bag of words model for the preprocessed text.

def bag\_ow(text, words, show\_details=True):

sentence\_words = cleaning(text)

bag\_of\_words = [0]\*len(words)

for s in sentence\_words:

for i,w in enumerate(words):

if w == s:

bag\_of\_words[i] = 1

return (np.array(bag\_of\_words))

Creating a prediction function for predicting the classes or tags of the question that are asked by the user.

def class\_prediction(sentence, model):

p = bag\_ow(sentence, lem\_words,show\_details=False)

result = bot\_model.predict(np.array([p]))[0]

ER\_THRESHOLD = 0.30

f\_results = [[i,r] for i,r in enumerate(result) if r > ER\_THRESHOLD]

f\_results.sort(key=lambda x: x[1], reverse=True)

intent\_prob\_list = []

for i in f\_results:

intent\_prob\_list.append({"intent": pred\_class[i[0]], "probability": str(i[1])})

return intent\_prob\_list

Now we are having the predicted classes or tags based on the inquiry of the user. As you can see in the intents file there are more than one response for each tag, so we are creating a function for selecting a random response from the predicted tag and sending it as a bot response.

def getbotResponse(ints, intents):

tag = ints[0]['intent']

intents\_list = intents['intents']

for intent in intents\_list:

if(intent['tag']== tag):

result = random.choice(intent['responses'])

break

return result

def bot\_response(text):

ints = class\_prediction(text, bot\_model)

response = getbotResponse(ints, intents)

return response

**Interacting with Chatbot**

We created several functions for the working of the chatbot. So let’s talk to our chatbot.

for i in range(3):

text = input("You : ")

print("Bot : ",bot\_response(text))

Output

You : hey

Bot : Hi there, how can I help you?

You : what help can you do

Bot : ticket booking for airline

You : bye

Bot : See you later

As you can see the chatbot is responding very well to us.