PROJECT REPORT

MIMIC LIKE HUMAN

By

ANANT KOTHIVAR (22BEC506)

DARSHIL MAVADIYA (22BEC508)

A Training Report Submitted to
Instituet Of Technology, Nirma University

22th April, 2024

Instituet Of Technology, Nirma University

---: CONTENTS :---

1. Overview of Project

Our Chat Bot is a computer program that can talk to humans in natural language, the way we interact with each other. It can replace a human for many tasks of answering queries. A chatbot is an agent that interacts with users using natural language. Several applications of chatbots such as Customer Service, call centers etc. uses Artificial Intelligence Mark-up Language to chat with users.

One of the prime goals of chatbots is to resemble an intelligent human and make it difficult for the receiver of the conversation to understand the real working along with various architecture and capabilities for their usage has widely broadened. These chatbots can prove sufficient to fool the user into believing they are "talking" to a human being, but are very limited in improving their knowledge base at runtime, and usually have little to no means of keeping track of all the conversation data. Chatbots makes use of machine learning to reach artificial intelligence helping them to understand the user query and provide an appropriate response. The chatbots are developed using the Artificial Intelligence Markup Language for communicating or interacting with the user. This consist a software which will be made up using Artificial Intelligence and will help user to chat with machine.

2. Objective of Project

❖ **The objectives of this project are:**

- To analyze users queries and understand users' messages.
- To provide an answer to the query of the user very effectively.
- To save the time of the user since s/he does not have to personally go to the college for inquiry.
- This system will help the student to be updated about the college activities.
- The system will reply using an effective GUI which implies that as if a real person is talking to the user.

❖ **The major features of the chat bot are:**

- College admission related queries could be answered through it.
- Viewing user profiles and retrieves attendance and grade/ pointers.
- College students can get information about examinations to be held.
- College students can fetch particulars about placement activities.

3. Introduction

❖ What is Chatbot?

- A chatbot is an intelligent piece of software that is capable of communicating and performing actions similar to a human. Chatbots are used a lot in customer interaction, marketing on social network sites and instantly messaging the client. There are two basic types of chatbot models based on how they are built; Retrieval based and Generative based models. Below some information about model are describe:

1. Retrieval based Chatbots:

A retrieval-based chatbot uses predefined input patterns and responses. It then uses some type of heuristic approach to select the appropriate response. It is widely used in the industry to make goal-oriented chatbots where we can customize the tone and flow of the chatbot to drive our customers with the best experience.

2. Generative based Chatbots:

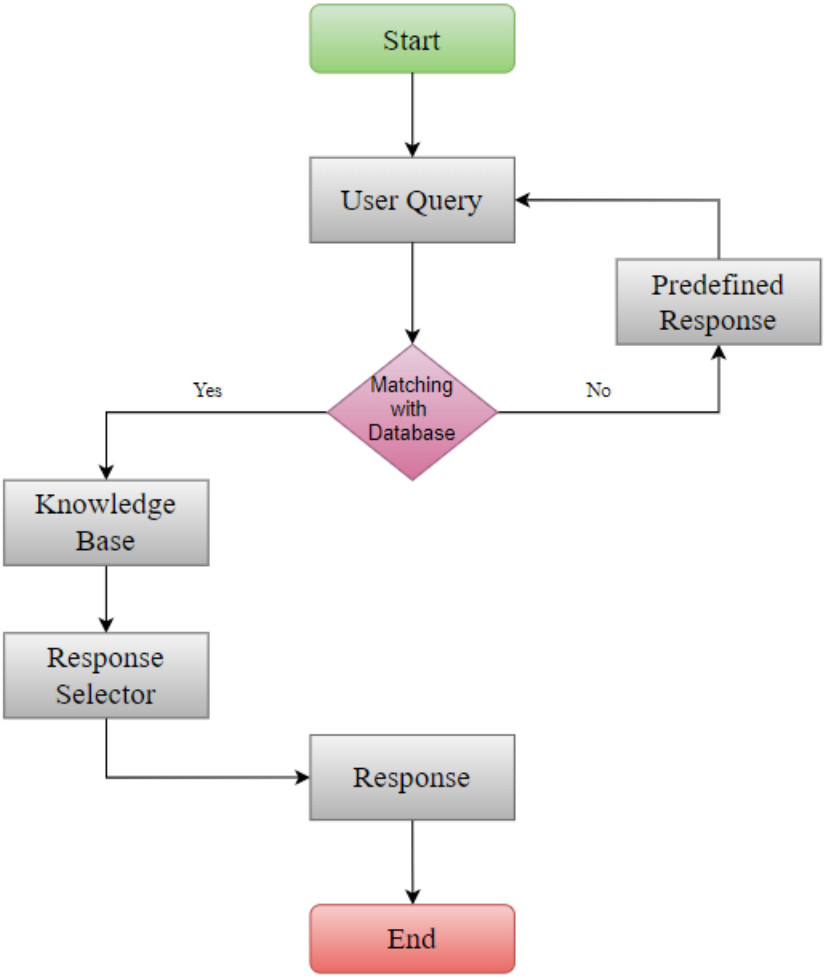
A generative model is not based on some predefined responses. They are based on sequence to sequence neural networks. It is the same idea as machine translation. In machine translation, we translate the source code from one language to another language but here, we are going to transform input into an output. It needs a large amount of data and it is based on Deep Neural networks.

❖ **Why we need Chatbots?**

- **Cost and Time Effective**- Humans cannot be active on-site 24/7 but chatbots can and the replying power of chatbots is much fast than humans.
- **Cheap Development cost**- With the advancement in technology many tools are developed that help easy development and integration of chatbots with little investment.
- **Human Resource**- Today Chatbots can also talk with text o speech technology so it gives the feel as a human is talking on another side.
- **Business Branding**- Businesses are changing with technology and chatbot is one out of them. Chatbot also helps in advertising, branding of organization product and services and give daily updates to users.

4. Diagrams

Use case:



5. Methodology

- The incremental build model is a method of software development where the product is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. This model combines the elements of the waterfall model with the iterative philosophy of prototyping. The basic algorithm that will be implemented for working of this proposed system is as follows:

  Step 1: Start.

  Step 2: Get the user query. (INPUT)

  Step 3: Pre-processing of the query E.g. suppose there is this year" So, we are going to remove these stop words like „is", „the" using pre-processing technique.

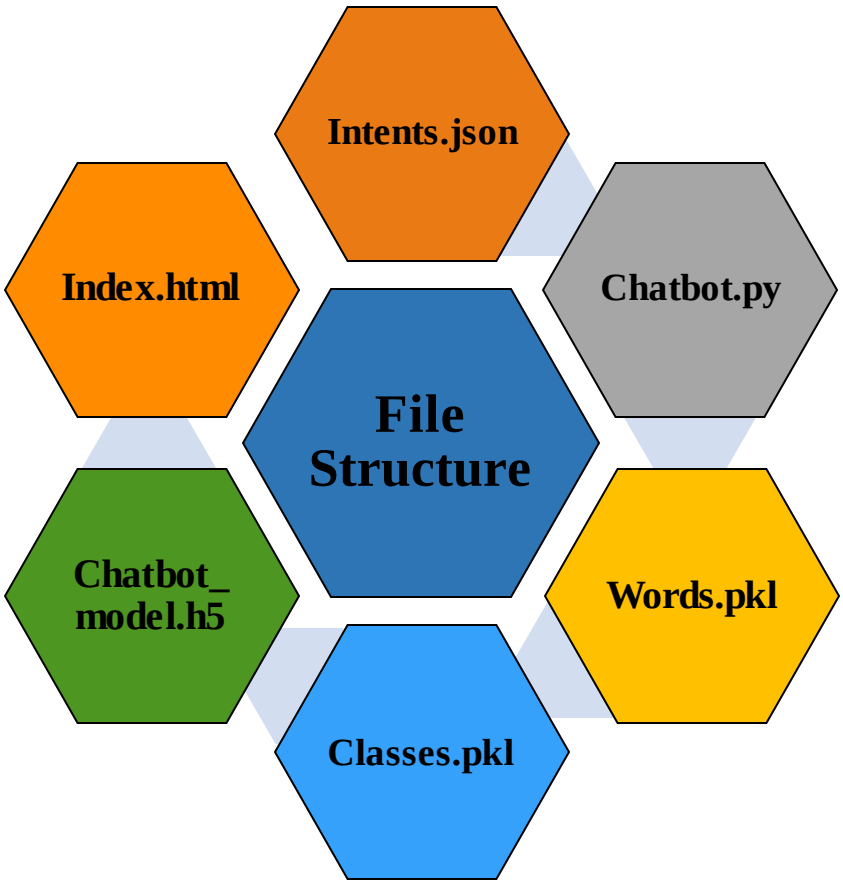
  Step 4: Fetch the remaining only keywords from the query.

  Step 5: Match the fetched keywords with the keywords in Knowledge base, and provide an appropriate response. The keywords will be matched with the help of keyword matching algorithm.

  Step 6: Return the query response as an output to the user.

  Step 7: Exit.

6. File Structure



7. Steps to create Chatbot

1. Import and load the data file
2. Preprocess data
3. Create training and testing data
4. Build the model
5. Predict the response
6. Source code of GUI
7. Run the chatbot

1. **Import and load the data file:**

- First, make a file name as train_chatbot.py. We import the necessary packages for our chatbot and initialize the variables we will use in our Python project.
- The data file is in JSON format so we used the json package to parse the JSON file into Python.

- Code:

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle
import warnings
warnings.filterwarnings('ignore')

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

words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('E:\intents.json').read() # read json file
intents = json.loads(data_file) # load json file
```

- This is how our intense.json file looks like:

```
{
        "intents": [
                {
                        "tag": "greeting",
                        "patterns": [
                                "Hi",
                                "How are you?",
                                "Is anyone there?",
                                "Hello",
                                "Good day",
                                "What's up",
                                "how are ya",
                                "heyy",
                                "whatsup"
                        ],
                        "responses": [

                                "Hello!",
                                "Good to see you again!",
                                "Hi there, how can I help?"
                        ],
                        "context_set": ""
                },
                {
                        "tag": "goodbye",
                        "patterns": [
                                "cya",
                                "see you",
                                "bye bye",
                                "See you later",
                                "Goodbye",
                                "I am Leaving",
                                "Bye",
                                "Have a Good day",
                                "talk to you later",
                                "ttyl",
                                "i got to go",
                                "gtg"
                        ],
                        "responses": [
                                "Sad to see you go :(",
                                "Talk to you later",
                                "Goodbye!",
                                "Come back soon"
                        ],
                        "context_set": ""
                },
                {
                        "tag": "creator",
```

2. **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. Based on the requirements we need to apply various operations to preprocess the data.
- 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.

```
for intent in intents['intents']:
    for pattern in intent['patterns']:
        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)# add each elements into list
        #combination between patterns and intents
        documents.append((w, intent['tag']))#add single element into end of list
        # add to tag in our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

- 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.

```
# lemmatize, lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents\n", documents, "\n")
# classes = intents[tag]
print (len(classes), "classes\n", classes, "\n")
# words = all words, vocabulary
print (len(words), "unique lemmatized words\n", words, "\n")
pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))
```

3. Create training and testing data

- Now, we will create the training data in which we will provide the input and the output. Our input will be the pattern and output will be the class our input pattern belongs to. But the computer doesn't understand text so we will convert text into numbers.

```
# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words
    pattern_words = doc[0]
    # convert pattern_words in lower case
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    # create bag of words array,if word match found in current pattern then put 1 otherwise 0.[row * colm(263)]
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # in output array 0 value for each tag ang 1 value for matched tag.[row * colm(8)]
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

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

4. Build the model

- We have our training data ready, now we will build a deep neural network that has 3 layers. We use the Keras sequential API for this. After training the model for 200 epochs, we achieved 100% accuracy on our model. Let us save the model as 'chatbot_model.h5'.

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

5. Predict the response

- To predict the sentences and get a response from the user to let us create a new file 'chatapp.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.

- Again we import the necessary packages and load the 'words.pkl' and 'classes.pkl' pickle files which we have created when we trained our model:

```
intents = json.loads(open('E:/intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
```

- 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 preprocessing and then predict the class.

```
# Utility Methods
def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    #print(sentence_words)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    #print(sentence_words)
    return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence
def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
#     print(sentence_words)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
#     print(bag)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1

                if show_details:
                    print ("found in bag: %s" % w)
                #print ("found in bag: %s" % w)
#     print(bag)
    return(np.array(bag))
def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
#     print(p)
    res = model.predict(np.array([p]))[0]
#     print(res)
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
#     print(results)
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
#     print(results)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
#     print(return_list)
    return return_list
```

- After predicting the class, we will get a random response from the list of intents:

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

6. Here is the full source code for the GUI:

```
import nltk

from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np
import re

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

def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
```

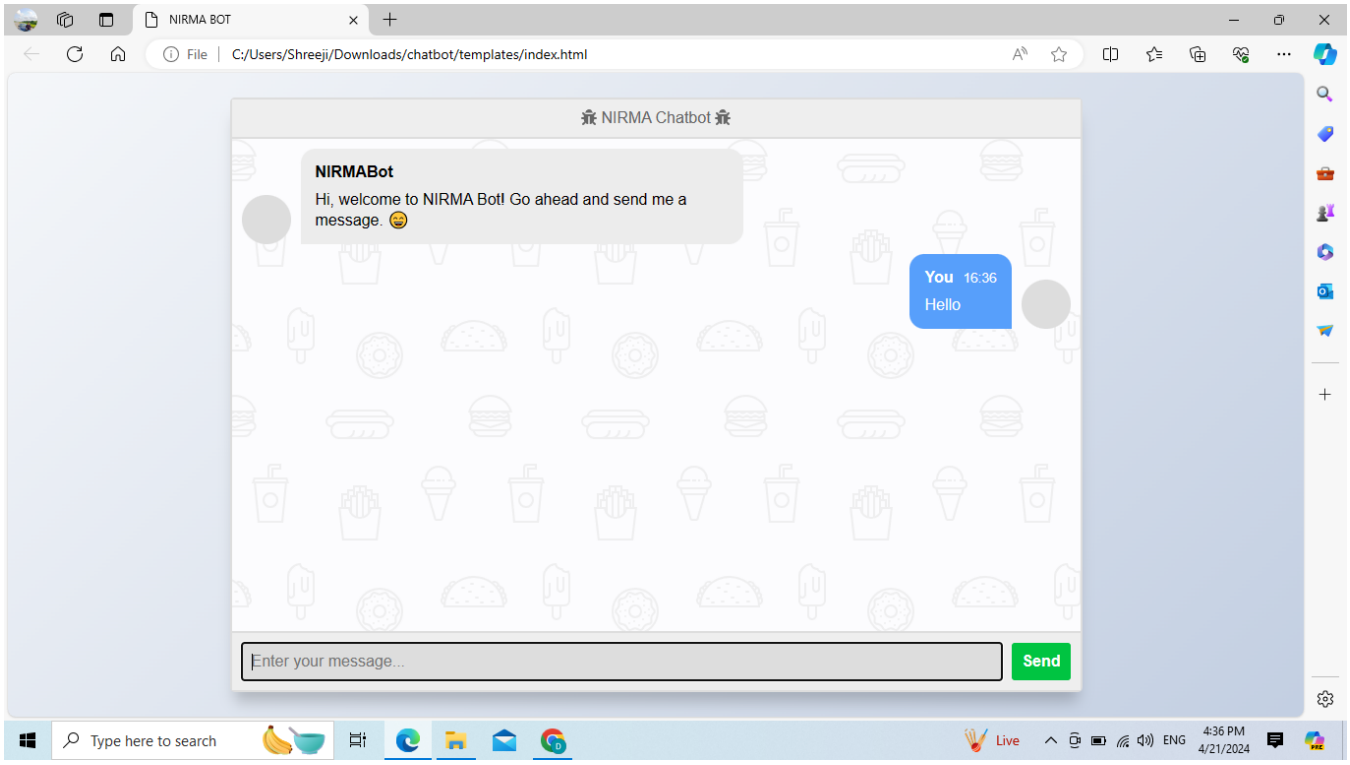
```
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in
sentence_words]
    return sentence_words

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

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

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

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

def chatbot_response(msg):
    ints = predict_class(msg, model)
```

```
    res = getResponse(ints, intents)
    return res


#Creating GUI with tkinter
import tkinter
from tkinter import *
import webbrowser

def openweb():
    webbrowser.open(url,new=new)

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

    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))

        res = chatbot_response(msg)
        if 'href' in res:
            links = re. findall("href=[\"\'](.*?)[\"\']", res)[0]
            ChatLog.insert(END, "Bot: " +   links +'\n\n')
        else:
            ChatLog.insert(END, "Bot: " + res + '\n\n')

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


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

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

ChatLog.config(state=DISABLED)

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

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

#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="29", height="5", font="Arial")
#EntryBox.bind("<Return>", send)


#Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)

base.mainloop()
```

7. Run the chatbot:

- To run the chatbot, we have two main files; train_chatbot.py and chatapp.py.



8. Conclusion

❖ **Summary:**

- Artificial Intelligent is the fastest growing technology everywhere in the word. With the help of Artificial Intelligent and Knowledgeable databases. We can make the transformation in the pattern matching and virtual assistance. This system is developing chatbot based on a web based system so with the combination of Artificial Intelligent Knowledgeable database and virtual assistance. We can develop such chat bot which will make a conversion between human and machine and will satisfy the question raised by the user.
- 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 are looking forward to implementing bot in their workflow.

❖ **Future Scope:**

- The future scope of this chatbot application will be:
    - More efficient chat bot.
    - Available in Gujarati or Hindi.
    - Become voice assistance.

9. References

- For Numpy:
    - Indian AI production
      https://youtu.be/zeGTmpkdXYU

- For Tensorflow:
    - Indian AI production
      https://youtu.be/x178e1Ykujo
    - Codebasics
      https://youtu.be/yfsTZbwgMSE

- For Keras:
    - Codebasics
      https://youtu.be/yfsTZbwgMSE

- For Nltk:
    - Sentdex
      https://youtu.be/FLZvOKSCkxY
    - Knowledge Shelf
      https://youtu.be/a8E2gksDXDQ

- For Chatbot implementation:
  https://data-flair.training/blogs/python-chatbot-project/

- For Chatbot GUI:
  https://youtu.be/a37BL0stIuM