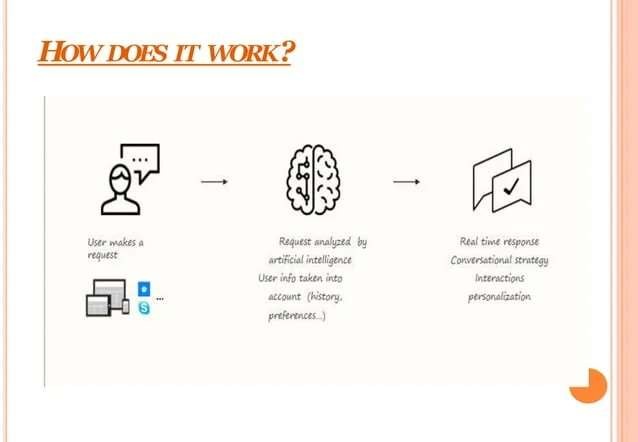
**THE FINAL DOCUMENTATION**

**“THE CHATBOT”**

**INTRODUCTION**

Chat bot is typically perceived as engaging software entity which humans can talk to. It can be interesting, inspiring and intriguing. It appears everywhere, from old ancient HTML pages to modern advanced social networking websites, and from standard computers to fashionable smart mobile devices

**WORKING PROCESS**



**TYPES OF CHATBOT :**

1. Flow-oriented Chatbot

2.Artificially intelligent Chatbot

3. Hybrid Chatbot

4. Human supportedbots

**Ensemble Methods:**

Ensemble methods combine predictions from multiple models to create a more accurate and robust prediction system. Some popular ensemble methods include:

* Voting
* Stacking
* Boosting
* Bagging (Bootstrap Aggregating)

**TECHNIQUES AND ARCHITECTURE**

* Transformers
* Autoencoders
* Transfer Learning
* Regularization Techniques

PYTHON PROGRAM TO DEVELOP A BASIC CHATBOT

#install packages

# pip install "chatterbot==1.0.0"

# pip install pytz # OR

# pip install -r requirements.txt

# import required packages from chatterbot import ChatBot

from chatterbot.conversation import Statement

from chatterbot.trainers import ChatterBotCorpusTrainer

# create ChatBot

chatBot = ChatBot('ChatBot')

# create ChatBot trainer

trainer = ChatterBotCorpusTrainer(chatBot)

# Train ChatBot with English language corpus # you can train with different language

# or with your custom .yam file trainer.train("chatterbot.corpus.english")

# Greeting from chat bot print("Hi, I am ChatBot")

# keep communicating with ChatBot while True:

# take user input/query query = input(">>>")

# response from ChatBot

# put query on Statement format to avoid runtime alert messages # Statement(text=query, search\_text=query)

print(chatBot.get\_response(Statement(text=query, search\_text=query)))

LIVE PROGRAM WITH OUTPUT

from IPython.core.display import Image, display display(Image('Untitled.png')) #Used in Tensorflow Model import numpy as np import tensorflow as tf import tflearn

import random

#Usde to for Contextualisation and Other NLP Tasks. import nltk

from nltk.stem.lancaster import LancasterStemmer stemmer

= LancasterStemmer()

#Other import json import pickle

import warnings

|  |
| --- |
| words = [] classes = [] documents = [] ignore\_words = ['?'] print("Looping through the Intents to Convert them to words, classes, documents and ignore\_words.......") for intent in intents['intents']: for pattern in intent['patterns']:  # tokenize each word in the sentence w = nltk.word\_tokenize(pattern)  # add to our words list |
| words.extend(w)  # add to documents in our corpus documents.append((w, intent['tag']))  # add to our classes list  if intent['tag'] not in classes:  classes.append(intent['tag']) |

print("Stemming, Lowering and Removing Duplicates ")

words = [stemmer.stem(w.lower()) for w in words if w not in ignore\_words] words

= sorted(list(set(words)))

# remove duplicates

classes = sorted(list(set(classes)))

print (len(documents), "documents") print (len(classes), "classes", classes)

print (len(words), "unique stemmed words", words)

print("Shuffling Randomly and Converting into Numpy Array for Faster Processing ")

random.shuffle(training) training = np.array(training)

print("Creating Train and Test Lists ") train\_x = list(training[:,0])

train\_y = list(training[:,1])

print("Building Neural Network for Out Chatbot to be Contextual ")

print("Resetting graph data ") tf.reset\_default\_graph()

print("Shuffling Randomly and Converting into Numpy Array for Faster Processing ")

random.shuffle(training) training = np.array(training)

if w in pattern\_words else bag.append(0)

# output is a '0' for each tag and '1' for current tag output\_row = list(output\_empty)

output\_row[classes.index(doc[1])]=1

training.append([bag, output\_row])

bag.append(1)

# create our bag of words array for w in words:

print("Creating the Data for our Model ")

training = [] output = []

print("Creating an List (Empty) for Output ") output\_empty

= [0] \* len(classes)

print("Creating Traning Set, Bag of Words for our Model ") for

doc in documents:

# initialize our bag of words bag = []

# list of tokenized words for the pattern

pattern\_words = doc[0] # stem each word

pattern\_words = [stemmer.stem(word.lower()) for word in pattern\_words]

net = tflearn.input\_data(shape=[None, len(train\_x[0])]) net = tflearn.fully\_connected(net, 8) net = tflearn.fully\_connected(net, 8)

net = tflearn.fully\_connected(net, len(train\_y[0]), activation='softmax') net = tflearn.regression(net) print("Training. .. ")

model = tflearn.DNN(net, tensorboard\_dir='tflearn\_logs') print("Training the Model ")

model.fit(train\_x, train\_y, n\_epoch=1000, batch\_size=8, show\_metric=True) print("Saving the Model ") model.save('model.tflearn')

Creating an List (Empty) for Output\

Creating Traning Set, Bag of Words for our Model....

Shuffling Randomly and Converting into Numpy Array for Faster Processing......

Creating Train and Test Lists.....

Building Neural Network for Out Chatbot to be Contextual Resetting

Training Step: 3999 | total loss: 0.06984 | time: 0.011s

| Adam | epoch: 1000 | loss: 0.06984 - acc: 0.9976 -- iter: 24/27 Training Step: 4000 | total loss: 0.07164 | time: 0.014s

| Adam | epoch: 1000 | loss: 0.07164 - acc: 0.9978 -- iter: 27/27

--

Saving the Model.......

INFO:tensorflow:E:\FreeBirdsCrew\Chatbot\model.tflearn is not in all\_model\_checkpoint\_paths. Manually adding it.

ERROR\_THRESHOLD = 0.25

Das while True:

def clean\_up\_sentence(sentence):

# It Tokenize or Break it into the constituents parts of Sentense. sentence\_words = nltk.word\_tokenize(sentence) # Stemming means to find the root of the word.

sentence\_words = [stemmer.stem(word.lower()) for word in sentence\_words]

return sentence\_words

# Return the Array of Bag of Words: True or False and 0 or 1 for each word of bag that exists in the Sentence def bow(sentence, words, show\_details=False): sentence\_words = clean\_up\_sentence(sentence) bag = [0]\*len(words) for s in sentence\_words:

for i,w in enumerate(words):

if w == s:

bag[i] = 1 if show\_details: print ("found in bag: %s" % w) return(np.array(bag))

ERROR\_THRESHOLD = 0.25

print("ERROR\_THRESHOLD = 0.25")

def classify(sentence):

# Prediction or To Get the Posibility or Probability from the Model results = model.predict([bow(sentence, words)])[0] # Exclude those results which are Below Threshold results = [[i,r] for i,r in enumerate(results) if r>ERROR\_THRESHOLD] # Sorting is Done because heigher Confidence Answer comes first.

results.sort(key=lambda x: x[1], reverse=True)

return\_list = [] for r in results:

return\_list.append((classes[r[0]], r[1])) #Tuppl -> Intent and Probability return return\_list

**input\_data = input("You- ") answer = response(input\_data) answer**

def response(sentence, userID='123', show\_details=False): results = classify(sentence)

# That Means if Classification is Done then Find the Matching Tag. if results:

# Long Loop to get the Result.

intents['intents']:

while results:

for i in

# Tag Finding if i['tag'] == results[0][0]:

# Random Response from High Order Probabilities return print(random.choice(i['responses']))

results.pop(0)

**CONCLUSION:-**

A chatbot is one of the simple ways to transport data from a computer without having to think for proper keywords to look up in a search or browse several web pages to collect information; users can easily type their query in natural language and retrieve information.