CREATING CHATBOT USING PYTHON

PHASE 4: DEVELOPMENT PART 2

CHATBOTS:

Chatbots, also known as conversational agents, are designed with the help of AI (Artificial Intelligence) software. They simulate a conversation (or a chat) with users in a natural language via messaging applications, websites, mobile apps, or phone.

CHATBOT FRAMEWORKS

Microsoft Bot Framework:

Microsoft Bot Framework platform helps you to build, connect, publish, and manage chatbots, which are smart and interactive to give the best user experience. It comes with active learning.
pre-built models to interact with your users on the following.

- Skype
- Slack

- Facebook Messenger
- Website **Wit.ai:**

Not just bots, but with the help of Wit ai, you can create automation for wearable devices, a voice interface for a mobile application, home electronics hardware.

Dialogflow:

Built text or voice-based conversational interfaces for your bots and application. Dialogflow is powered by Google's machine learning, which can be used to connect to users on Google Assistant, Amazon Alexa, Mobile apps, Messenger, websites, Slack, Twitter, and more.

the following to detect intent and agent API.

- PHP
- Go
- Java (Maven)
- Ruby (Gem)
- Python
- C#
- Node.js

Chatbots: Organizing the Chaos of IoT:

❖ IoT, or the Internet of Things, is the collective term for all interconnected devices that operate within the infrastructure of the Internet. IoT devices range from mobile phones and smart watches to connected vehicles and even entire homes and power grids.

❖ The sky is truly the limit with the potential for IoT technology, but there are a few severely limiting factors aligned with current trends; most notably, managing operations, maintaining security and analyzing data.

- The solution to these problems lies with chatbots. Despite the name, artificial intelligence-driven bots are capable of much more than many people believe.

MANAGING IOT DEVICES:

- As IoT devices proliferate into various realms of personal, enterprise, and government sectors, the management and operation of an increasing number of devices can become arduous. The necessity of learning new systems and interfaces leads to general dissatisfaction and contributes to ever-growing app fatigue.

- Chatbots, on the other hand, are programmed to respond to consumer inquiries in a meaningful, conversational way, while at the same time automating tasks that might otherwise be relegated to an app or website.

- Example:

- if a user wanted to turn on the lights before arriving at a connected smart home, they could simply send a text message, via their preferred platform, in human language:
  "Turn on the living room lights". With AI functionality, it can really be that simple.

Chatbot in present Generation:

- Today, we have smart Chatbots powered by Artificial Intelligence that utilize natural language processing (NLP) in order to understand the commands from humans (text and voice) and learn from experience. Chatbots have become a staple customer interaction utility for companies and brands that have an active online existence (website and social network platforms).

- ❖ With the help of Python, Chatbots are considered a nifty utility as they facilitate rapid messaging between the brand and the customer. Let us think about Microsoft's Cortana, Amazon's Alexa, and Apple's Siri. Aren't these chatbots wonderful? It becomes quite interesting to learn how to create a chatbot using the Python programming language.
- ❖ Fundamentally, the chatbot utilizing Python is designed and programmed to take in the data we provide and then analyze it using the complex algorithms for Artificial Intelligence. It then delivers us either a written response or a verbal one. Since these bots can learn from experiences and behavior, they can respond to a large variety of queries and commands.
- ❖ Although chatbot in Python has already started to rule the tech scenario at present, chatbots had handled approximately 85% of the customer-brand interactions by 2020 as per the prediction of Gartner.
- ❖ In light of the increasing popularity and adoption of chatbots in the industry, we can increase the market value by learning how to create a chatbot in Python - among the most extensively utilized programming languages globally.

CHATBOT APPLICATIONS:

Birthday Remainder import

datetime current_date =

datetime.date.today().strftim

e('%Y-%m-%d')

```
current_date_lst =
current_date.split('-')
bday_log = [
   ('Ayushi', ('1999', '10', '19')),    ('Yash', ('1999',
'04', '21')),] add = input('To add birthday type
y:').strip().lower() if add[:1] == 'y':
new = input('Add birthday in format yyyy-mm-dd:')
# print(new_lst) name =
input('Whose bday?') date =
new.split( '-' )
bday_log.append((name, tuple(date)))
for birthday in bday_log:
   # current_dat[1] == birthday[1][1] this will check if current
month is same as birth month  and current date is same as
   # birth date as per preadded log  if current_date_lst[1] ==
birthday[1][1] and current_date_lst[2] == birthday[1][2]:
 age = int(current_date_lst[0]) - int(birthday[1][0])
ordinal_suffix = {1: 'st', 2: 'nd', 3: 'rd', 11: 'th', 12: 'th', 13:
'th'}.get(age % 10 if not 10 < age <= 13 else age % 14, 'th')
print(f" It's {birthday[0]}'s {age}{ordinal_suffix} Birthday")
```

DEVELOPMENT OF CHATBOT:

```
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
```

warnings.filterwarnings("ignore")curses is not supported on this machine (please install/reinstall curses for an optimal experience)

```
print("Processing the Intents.....")
with open('intents.json') as json_data:
intents = json.load(json_data)
```

Processing the Intents.....

```
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'])
```

Looping through the Intents to Convert them to words, classes, documents and ignore_words.......

```
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)
```

```
Stemming, Lowering and Removing Duplicates.......
27 documents
9 classes ['goodbye', 'greeting', 'hours', 'mopeds', 'opentoday', 'payments',
'rental', 'thanks', 'today']
48 unique stemmed words ["'d", "'s", 'a', 'acceiv', 'anyon', 'ar', 'bye', 'can',
'card', 'cash', 'credit', 'day', 'do', 'doe', 'good', 'goodby', 'hav', 'hello', 'help',
'hi', 'hour', 'how', 'i', 'is', 'kind', 'lat', 'lik', 'mastercard', 'mop', 'of', 'on', 'op',
'rent', 'see', 'tak', 'thank', 'that', 'ther', 'thi', 'to', 'today', 'we', 'what', 'when',
'which', 'work', 'yo', 'you']
```

```
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]

   # create our bag of words array     for w in words:         bag.append(1)
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])
```

```
Creating  the data for Model…..
```

Creating an List (Empty) for Output..... Creating Traning Set, Bag of Words for our Model....

```
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)
```

```
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 graph data....
```

```
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....")
```

```
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')
```

```
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.

```
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
```

```
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.         while results:            for i in
intents['intents']:
                # Tag Finding                if i['tag'] == results[0][0]:
                    #  Random  Response  from  High  Order  Probabilities                            return
print(random.choice(i['responses']))

            results.pop(0)
```

ERROR_THRESHOLD = 0.25

```
Das while
True:
    input_data = input("You- ")
answer = response(input_data)
answer
```

OUTPUT:

```
You- Hi Hello, thanks
for visiting You- open
today?
Our hours are 9am-9pm every day You-
mopeds
Hi there, how can I help? You-
rent?
Hi there, how can I help? You-
rent mopeds?
Hello, thanks for visiting
You- rental
Good to see you again
You- Rental Mopends?
Hi there, how can I help?
You- how Much? Good
to see you again
You- much?
```

```
Bye! Come back again soon.
You- much
See you later, thanks for visiting
You- s
See you later, thanks for visiting
```

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.