**Problem and solution for project**

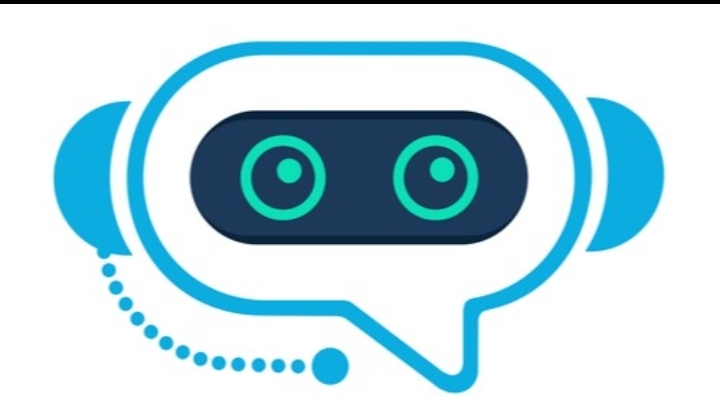
**Project Title : Create a chatbot in python**

**Problem Statement :**

When using an app or website, customers expect outstanding service. They can become disinterested in the app if they can't locate the solution to a question they have. To avoid losing customers and having an adverse effect on your bottom line, you must provide the highest quality service possible while developing a website or application.

**"Mastering Dataset Loading and Preprocessing:**

**Python's AI Chatbot Prowess"**



Natural Language processing is one of the advanced fields of artificial intelligence which makes the systems understand and process the human language. The main use-case of NLP can be seen in chatbot development, spam classification, and text summarization. In today’s article, we’re going to discuss the chatbot application of NLNLP

**ADVANCED CHATBOT**

Chatbots are computer software programs that can interact with humans. With the advancement in machine learning mainly natural language processing, everyone started to create intelligent chatbot systems. You can see different types of chatbots on different websites, chatbots for booking airline tickets on Airline company websites, customer support chatbots in different apps, etc… are such examples. Let create one such chatbot!!.

**Chatbot Development Source: Salesforce**

**Implementation**

**Virtual Environment Creation**

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 LiLibraries

To install the libraries required for this project separately in this enenvironment

**pip install keras nltk tensorflow**

Creating Intents File

First of all, let’s look into our intents\_file.json file. This intents file contains the different patterns of the question that the user might enquire and the possible output for the specific question and a tag for that type of question

**Code :**

{"intents": [

{"tag": "greetings",

"patterns": ["Hello there", "Hey, How are you", "Hey", "Hi", "Hello", "Anybody", "Hey there"],

"responses": ["Hello, I'm your helping bot", "Hey it's good to see you", "Hi there, how can I help you?"],

"context": [""]

},

{"tag": "thanks",

"patterns": ["Thanks for your quick response", "Thank you for providing the valuable information", "Awesome, thanks for helping"],

"responses": ["Happy to help you", "Thanks for reaching out to me", "It's My pleasure to help you"],

"context": [""]

},

{"tag": "no\_answer",

"patterns": [],

"responses": ["Sorry, Could you repeat again", "provide me more info", "can't understand you"],

"context": [""]

},

{"tag": "support",

"patterns": ["What help you can do?", "What are the helps you provide?", "How you could help me", "What support is offered by you"],

"responses": [ "ticket booking for airline", "I can help you to book flight tickets easily"],

"context": [""]

},

{"tag": "goodbye",

"patterns": ["bye bye", "Nice to chat with you", "Bye", "See you later buddy", "Goodbye"],

"responses": [ "bye bye, thanks for reaching", "Have a nice day there", "See you later"],

"context": [""]

}

]

}

The above shows the intents file that we are going to use in our project

**Implementation**

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.

. **Code**: 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 dadata.

**nltk.download('punkt')**

**nltk.download('wordnet')**

**Preprocessing**

Loading the JSON file and reading it. Also, we are initializing some lists for saving the preprocessed and preprocessing data.

**tokenized\_words=[]**

**classes = []**

**doc = []**

**ignoring\_words = ['?', '!']**

**data\_file = open('intents\_file.json').read()**

**intents = json.loads(data\_file)**

We loaded the JSON file. Now we have to perform some preprocessing, we are going to iterate through each of the patterns questions in the intents file and tokenize it. This tokenized text along with the tag is stored as a list. tokenized\_words contains all the different words in the intents file which is tokenized using nltk.

**for intent in intents['intents']:**

**for pattern in intent['patterns']:**

**w = nltk.word\_tokenize(pattern) #tokenizing**

**tokenized\_words.extend(w)**

**doc.append((w, intent['tag']))**

**if intent['tag'] not in classes:**

**classes.append(intent['tag'])**

To perform lemmatization on the data and need to remove the question tag and other ignoring words

**lemmatizer = WordNetLemmatizer()**

**lemmatized\_words = [lemmatizer.lemmatize(words.lower())**

**for words in tokenized\_words**

**if w not in ignoring\_words] #lemmatization**

Then we are sorting the unique lemmatized words and classes.

**lemmatized\_words = sorted(list(set(lemmatized\_words)))**

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

Now saving the lemmatized words and classes into a pickle file

**pickle.dump(lemmatized\_words,open('lem\_words.pkl','wb'))**

**pickle.dump(classes,open('classes.pkl','wb'))**

As the next step, we need to create our training data. The input feature is the bag of words model of questions that the user is asking and the output feature is the tag or class that the input question pattern belongs to.

training\_data = []

empty\_array = [0] \* len(classes)

for d in doc:

bag\_of\_words = []

pattern = d[0]

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

for w in lemmatized\_words:

bag\_of\_words.append(1) if w in pattern else bag\_of\_words.append(0)

output\_row = list(empty\_array)

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

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

random.shuffle(training\_data)

training = np.array(training\_data)

train\_x = list(training[:,0])

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

**Model Creation**

Now we can create our Neural network model. With the help of Keras and TensorFlow library, we are creating the Model. So let’s start the implementation.

Let's creating a Sequential model and then adding layers to this sequential model.

bot\_model = Sequential()

bot\_model.add(Dense(128, input\_shape=(len(x\_train[0]),), activation='relu'))

bot\_model.add(Dropout(0.5))

bot\_model.add(Dense(64, activation='relu'))

bot\_model.add(Dropout(0.5))

bot\_model.add(Dropout(0.25))

bot\_model.add(Dense(len(y\_train[0]), activation='softmax'))

We’ve created our model. Next is to compile our model with the stochastic gradient descent feature.

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

bot\_model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])

Now let’s fit the model

**x\_train = np.array(x\_train)**

**y\_train = np.array(y\_train)**

**hist = bot\_model.fit(x\_train, y\_train, epochs=200, batch\_size=5, verbose=1)**

Let created chatbot model so we can save this model for future use.

**bot\_model.save('chatbot\_model.h5', hist)**

**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.

**CODE:**

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.

**CODE:**

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.

**Conclusion :**

In this article, we’ve briefly discussed chatbot development from scratch. And you got the idea about the working flow or data flow of chatbot making and prediction of the response. The main key insights from this article are