

**GOVERNMENT COLLEGE OF ENGINEERING BARGUR**

**( AUTONOMOUS)**

**PROJECT TITLE: CHATBOT DEPLOYMENT WITH IBM CLOUD WATSON ASSISTANT**

**TEAM MEMBERS:**

**CHANDRU D**

**NAVINKUMAR N**

**MANIKANDAN V**

**BHARATH P**

**JAISURYA G**

**Building the project by loading and preprocessing the database:**

1. Import the libraries:

import tensorflow

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout , Activation, Flatten , Conv2D, MaxPooling2D

from tensorflow.keras.optimizers import SGD

import random

import json

import pickle

These all are the libraries we will require to create our project. nltk is required to tokenize words and sentences and also to lemmatize words. Basically nltk will be required to preprocess our data(we have to perform certain operations on our data as we will be working on a json file which contains words, and sentences).

And we will be using Tensorflow for creating our model, numpy to convert our data into array form. Random to generate random responses according to the user message. Json to read the json file. Pickle to save our labels and words.

2. Declaring Constants:

In this step, we declare some constants that will be required while separating sentences.

words=[]

labels = []

docs = []

ignore\_list = ['?', '!']

3. Loading our dataset that is intents.json file:

Load the json dataset using json.loads() method.

dataset = open('intents.json').read()

intents = json.loads(dataset)

4. Preprocess Data:

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

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

#tokenize each word

word\_token = nltk.word\_tokenize(pattern)

words.extend(word\_token)

#add documents in the corpus

docs.append((word\_token, intent['tag']))

# add to our labels list

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

labels.append(intent['tag'])

As we are working on text data, we need to perform certain operations or say preprocessing on data before creating a model to train on that data. So, in the above code we first iterate through our intents and patterns, and we tokenize each sentence present in that pattern (Tokenizing means breaking text into small parts like words), and then append each tokenize word into the words list. And in this we also create a list of labels for our tags.

5. Lemmatizing Each word:

# lemmatize each word, and sort words by removing duplicates:

words = [lemmatizer.lemmatize(word.lower()) **for** word **in** words **if** word not **in** ignore\_list]

words = sorted(list(set(words)))

# sort labels:

labels = sorted(list(set(labels)))

In this code, we lemmatize each word (Lemmatizing means converting a word into its lemma form) and also remove duplicate words from the list and sort words and labels list.

6. Save words and labels list (using pickle):

Now we will save our words and labels list that we have created using the pickle library.

pickle.dump(words,open('words.pkl','wb'))

pickle.dump(labels,open('labels.pkl','wb'))

7. Creating our Training data:

# creating our training data:

training\_data = []

# creating an empty array for our output (with size same as length of labels):

output = [0]\*len(labels)

**for** doc **in** docs:

bag\_of\_words = []

pattern\_words = doc[0]

#lemmatize pattern words:

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

**for** w **in** words:

**if** w **in** pattern\_words:

bag\_of\_words.append(1)

**else**:

bag\_of\_words.append(0)

output\_row = list(output)

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

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

In this piece of code, we create our training data in which we will provide the input that is bag\_of\_words that will be pattern and ouput\_row which will be the output which tells us in which label our pattern belongs to. As the computer doesn’t understand text, that is why we have converted it to numbers.

8. Shuffle and Convert our Training data to array:

We shuffle our training data using random.shuffle() method, and also convert our data to a numpy array using numpy library.

# convert training\_data to numpy array and shuffle the data:

random.shuffle(training\_data)

training\_data = np.array(training\_data)

9. Splitting the data into x\_train and y\_train:

Splitting our training data into x\_train and y\_train. X\_train consist of words and y\_train consists of its corresponding label.

# Now we have to create training list:

x\_train = list(training\_data[:,0])

y\_train = list(training\_data[:,1])

10. Model Creation:

# Creating Model:

model = Sequential()

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

model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.5))

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

In this model, we will create 3 fully connected layers in which there is one input layer and one output layer. As you can see in the above code.

11. Compile and Fit our model to find the accuracy:

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

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

In this, we will be using a Stochastic gradient descent(sgd) optimizer with Nesterov accelerated gradient.

# fit the model

history = model.fit(np.array(x\_train), np.array(y\_train), epochs=200, batch\_size=5, verbose=1)

12. Save the model:

Now after creating the model we will save our model using save() method.

model.save('chatbot\_Application\_model.h5', history)

13. Final step to predict the sentences and get responses:

Now, we have to create one more python file in which we load our model, we load our words list, labels list that we have saved above. As we know that our model will only predict the label in which it belongs to, so we have to create certain functions which will identify the label and provide random responses from the list of responses.

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import pickle

import numpy as np

import json

import random

from keras.models import load\_model

model = load\_model('chatbot\_Application\_model.h5')

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

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

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

**To run our model we have to provide the input in the same way as we have done while creating our model. So for this we have created a function which will perform text operations and then predict the label.**

**def** bank\_of\_words(s,words, show\_details=**True**):

bag\_of\_words = [0 **for** \_ **in** range(len(words))]

sent\_words = nltk.word\_tokenize(s)

sent\_words = [lemmatizer.lemmatize(word.lower()) **for** word **in** sent\_words]

**for** sent **in** sent\_words:

**for** i,w **in** enumerate(words):

**if** w == sent:

bag\_of\_words[i] = 1

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

**def** predict\_label(s, model):

# filtering out predictions

pred = bank\_of\_words(s, words,show\_details=**False**)

response = model.predict(np.array([pred]))[0]

ERROR\_THRESHOLD = 0.25

final\_results = [[i,r] **for** i,r **in** enumerate(response) **if** r>ERROR\_THRESHOLD]

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

return\_list = []

**for** r **in** final\_results:

return\_list.append({"intent": labels[r[0]], "probability": str(r[1])})

**return** return\_list

**After prediction, now we will create a function which will give responses from the list of intents.**

**def** Response(ints, intents\_json):

tags = ints[0]['intent']

list\_of\_intents = intents\_json['intents']

**for** i **in** list\_of\_intents:

**if**(i['tag']== tags):

response = random.choice(i['responses'])

break

**return** response

**def** chatbot\_response(msg):

ints = predict\_label(msg, model)

response = Response(ints, intents)

**return** response

**Now after responses in this we have created a function which will make user and Bot interact:**

**def** chat():

print("Start chat with ChatBot of ProjectGurukul")

**while** **True**:

inp = input("You: ")

**if** inp.lower() == 'quit':

break

response = chatbot\_response(inp)

print("\n BOT: " + response + '\n\n')

chat()

14. Now Run this python file.