|  |  |  |
| --- | --- | --- |
| **SL.No** | **Contents** | **Page No.** |
| 1 | Abstract | 03 |
| 2 | Introduction (to technology - AI and Chatbot) | 03-04 |
| 3 | Literature Survey / Review | 04 |
| 4 | Problem Statement | 04 |
| 5 | Methodology | 05 |
| 6 | Modules identified and code | 05-09 |
| 7 | Outcomes / Results and Discussions | 10 |
| 8 | Conclusion | 10 |
| 9 | References | 10 |

**ABSTRACT:**

The main aim and objective of this project is to build a AI Chatbot for Medical Appointment booking.

We can see chatbots are being encouraged and adopted by many of the industries and universities like “Make my Trip”.

In this project we tried to build a chatbot focused for Medical World and trained it in such a way that other than the company Information it can also entertain some of the basic medical queries.

**INTRODUCTION:**

Chatbots are basically AI intelligence bots which can interact with the user or customers depends upon the usage.

It is an application of Artificial Intelligence and Machine Learning­. Now-a-days technology is increasing rapidly. In this technological world every industry is trying to automate things to provide better services. One of the great application of automation would be chatbot.

There are basically two types of Chatbots:

**Command based**: Chatbots that function on predefined rules and can answer to only limited queries or questions.

Users need to select an option to determine their next step.

**Intelligent/AI Chatbots**: Chatbots that leverage Machine Learning and Natural Language Understanding to understand the user’s language and are intelligent enough to learn from conversations with their users.

You can converse via text, speech or even interact with a chatbot using graphical interfaces.

Here in this project, we created an AI Chatbot which is focused for Medical Hospitals and trying to solve of the use cases of hospitals by training it in such a way that it can answer some of the basics queries of Hospitals as well other than just the customer services.

**LITERATURE REVIEW:**

The AI Chatbot is being implemented by many of the industries. The basic concept behind it is NLP and Neural Network.

Chatbot can be built by many different ways and many different libraries are available in python like “chatterbot”.

The concept that we used is building a neural network by Torch and nltk.

Also, for more clarification we took help from Google, StackOverflow and Github

**Problem Statement:**

Build interesting and effective AI Chatbot.

**Methodology:**

The methodology that we used for this application is very simple.

We used the concept of Natural Language Processing and building our own Neural network by using nltk.

We have used a json file “intents.json” for training our model. The intents.json is the file which consists of some sample chats and each chat block under a “tag”.

We have provided some basic chats regarding the Medical Appointment Bookings and some basics terminologies and some basic queries related to Medical World.

We used this file for training the model and the model is being trained by using fit() method of tflearn and saved the model as “Chatbot”.

All the trained data is being stored in a file named “data.pth” and further this file is used for giving response.

**Modules identified and code:**

* numpy
* random
* json
* nltk
* PorterStemmer
* torch

**CODE:**

All the trained data is being stored in a file named “data.pth” and further this file is used for giving response

import numpy as np

import random

import json

import nltk

from nltk.stem.porter import PorterStemmer

nltk.download('punkt')

import torch

import torch.nn as nn

from torch.utils.data import Dataset, DataLoader

stemmer = PorterStemmer()

def tokenize(sentence):

return nltk.word\_tokenize(sentence)

def stem(word):

return stemmer.stem(word.lower())

def bag\_of\_words(tokenized\_sentence, words):

sentence\_words= [stem(word) for word in tokenized\_sentence]

bag= np.zeros(len(words), dtype=np.float32)

for idx, w in enumerate(words):

if w in sentence\_words:

bag[idx]= 1

return bag

class NeuralNet(nn.Module):

def \_\_init\_\_(self, input\_size, hidden\_size, num\_classes):

super(NeuralNet, self).\_\_init\_\_()

self.l1= nn.Linear(input\_size, hidden\_size)

self.l2= nn.Linear(hidden\_size, hidden\_size)

self.l3= nn.Linear(hidden\_size, num\_classes)

self.relu= nn.ReLU()

def forward(self, x):

out= self.l1(x)

out= self.relu(out)

out= self.l2(out)

out= self.relu(out)

out= self.l3(out)

return out

with open('intents.json', 'r') as f:

intents= json.load(f)

all\_words= []

tags= []

xy= []

for intent in intents['intents']:

tag= intent['tag']

tags.append(tag)

for pattern in intent['patterns']:

w= tokenize(pattern)

all\_words.extend(w)

xy.append((w, tag))

ignore\_words= ['?', '.', '!']

all\_words= [stem(w) for w in all\_words if w not in ignore\_words]

all\_words= sorted(set(all\_words))

tags= sorted(set(tags))

print(len(xy), "patterns")

print(len(tags), "tags:", tags)

print(len(all\_words), "unique stemmed words:", all\_words)

**#Train data**

X\_train= []

y\_train= []

for (pattern\_sentence, tag) in xy:

bag= bag\_of\_words(pattern\_sentence, all\_words)

X\_train.append(bag)

label= tags.index(tag)

y\_train.append(label)

X\_train= np.array(X\_train)

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

**#parameter tuning**

num\_epochs= 1000

batch\_size= 8

learning\_rate= 0.001

input\_size= len(X\_train[0])

hidden\_size= 8

output\_size= len(tags)

print(input\_size, output\_size)

class ChatDataset(Dataset):

def \_\_init\_\_(self):

self.n\_samples= len(X\_train)

self.x\_data= X\_train

self.y\_data= y\_train

def \_\_getitem\_\_(self, index):

return self.x\_data[index], self.y\_data[index]

def \_\_len\_\_(self):

return self.n\_samples

dataset= ChatDataset()

train\_loader= DataLoader(dataset=dataset,

batch\_size=batch\_size,

shuffle=True,

num\_workers=0)

device= torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

model= NeuralNet(input\_size, hidden\_size, output\_size).to(device)

criterion= nn.CrossEntropyLoss()

optimizer= torch.optim.Adam(model.parameters(), lr=learning\_rate)

**#Model training**

for epoch in range(num\_epochs):

for (words, labels) in train\_loader:

words= words.to(device)

labels= labels.to(dtype=torch.long).to(device)

outputs= model(words)

loss= criterion(outputs, labels)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

if (epoch+1)%100== 0:

print (f'Epoch [{epoch+1}/{num\_epochs}], Loss: {loss.item():.4f}')

print(f'final loss: {loss.item():.4f}')

data= {

"model\_state": model.state\_dict(),

"input\_size": input\_size,

"hidden\_size": hidden\_size,

"output\_size": output\_size,

"all\_words": all\_words,

"tags": tags

}

FILE= "data.pth"

torch.save(data, FILE)

print(f'Training done. file saved to {FILE}')

with open('intents.json', 'r') as json\_data:

intents= json.load(json\_data)

FILE= "data.pth"

data= torch.load(FILE)

input\_size= data["input\_size"]

hidden\_size= data["hidden\_size"]

output\_size= data["output\_size"]

all\_words= data['all\_words']

tags= data['tags']

model\_state= data["model\_state"]

model= NeuralNet(input\_size, hidden\_size, output\_size).to(device)

model.load\_state\_dict(model\_state)

model.eval()

bot= "Doctor AI"

print("Let's chat! (type 'quit' to exit)")

while True:

sentence= input("You: ")

if sentence== "quit":

break

sentence= tokenize(sentence)

X= bag\_of\_words(sentence, all\_words)

X= X.reshape(1, X.shape[0])

X= torch.from\_numpy(X).to(device)

output= model(X)

\_,predicted= torch.max(output, dim=1)

tag= tags[predicted.item()]

probs= torch.softmax(output, dim=1)

prob= probs[0][predicted.item()]

if prob.item() > 0.75:

for intent in intents['intents']:

if tag== intent["tag"]:

print(f"{bot}: {random.choice(intent['responses'])}")

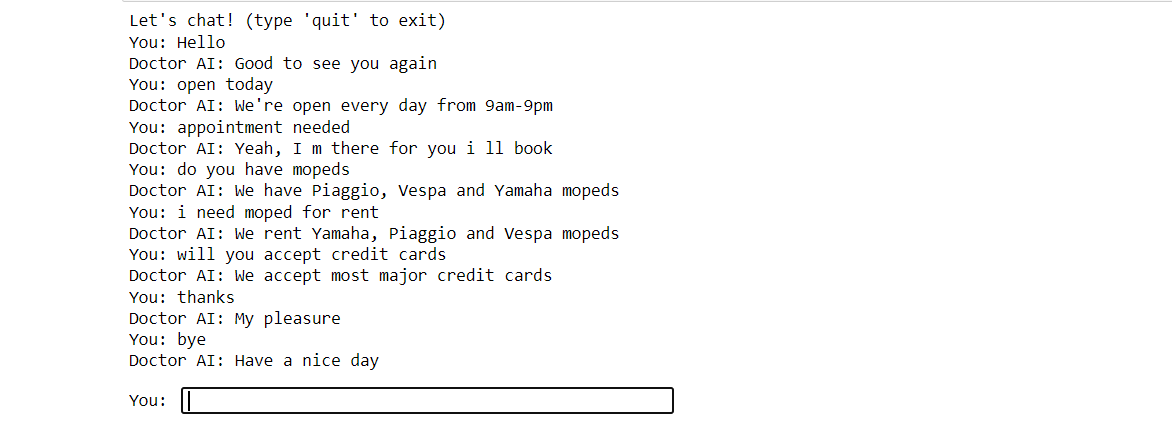
else:

print(f"{bot}: I do not understand, say again please..")

**RESULTS AND DISCUSSION:**

Our AI Chatbot is intelligent as it can answer the question even if it not the exact same as we provided in training data.

It also reply in the same context as the user is taking with.



**CONCLUSION:**

We conclude that chatbots build with NLP and Neural Network are more efficient than just by using pre-defined libraries of python for building chatbot. AI Chatbots can save time as well as labour work and are more efficient by providing 24\*7 services.

**REFERENCES:**

Google,

Github,

Official docs of python,Torch, etc.