**CHAPTER 1**

**INTRODUCTION**

Chatbot are stimulations that can understand human language, process it, and interact back with humans while performing specific tasks. For example, a chatbot can be employed as a helpdesk executive. Joseph Weizenbaum created the first chatbot in 1966, named Eliza.

Chatbot is an AI chatbot that uses natural language processing to create humanlike conversational dialogue. The language model can respond to questions.

Chatbots can automate tasks performed frequently and at specific times. This gives employees time to focus on more important tasks and prevents customers from waiting to receive responses.

**CHAPTER 2**

**OBJECTIVES**

Chatbots allow businesses to connect with customers in a personal way without the expense of human representatives. For example, many of the questions or issues customers have are common and easily answered. That's why companies create FAQs and troubleshooting guides.

**CHAPTER 3**

**LANGUAGE SPECIFICATION**

**3.1 Python?**

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. Python is one of the most popular languages for machine learning and has several powerful machine learning libraries, including TensorFlow and scikit-learn.

**3.2 JavaScript?**

JS stands for JavaScript. It is a text-based, lightweight, cross-platform, and interpreted scripting programming language.

This language is very popular for developing web pages.

It can be used both on the client-side and server-side.

JS is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.

**3.3 CSS?**

CSS is the acronym of “Cascading Style Sheets”. CSS is a computer language for laying out and structuring web pages (HTML or XML). This language contains coding elements and is composed of these “cascading style sheets” which are equally called CSS files (. css).

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more.

**3.4 JSON?**

JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).JSON stores data and the data can be easily accessed using key words.

**3.5 NLTK?**

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

**3.6 PyTorch?**

PyTorch is a fully featured framework for building deep learning models, which is a type of machine learning that's commonly used in applications like image recognition and language processing. Written in Python, it's relatively easy for most machine learning developers to learn and use.

**CHAPTER 4**

**SYSTEM SPECIFICATION**

**4.1 SOFTWARE REQUIREMENTS:**

Operating system: Windows 7 to 11

Language: Python,Html,Javascript,Css

**4.2 HARDWARE REQUIREMENTS:**

Processor: Intel core i5 to i7 and 8 Octa

Ram: 32GB

System type: 64bit Operating system

Hard disk: 500GB

**CHAPTER 5**

**IMPLEMENTATION**

* **HTML**

<!DOCTYPE html>

<html lang="en">

<link rel="stylesheet" href="style.css">

<head>

<meta charset="UTF-8">

<title>Chatbot</title>

</head>

<body>

<div class="container">

<canvas id="canvas"></canvas>

<div class="chatbox">

<div class="chatbox\_\_support">

<div class="chatbox\_\_header">

<div class="chatbox\_\_image--header">

<img src="https://img.icons8.com/color/48/000000/circled-user-female-skin-type-5--v1.png" alt="image">

</div>

<div class="chatbox\_\_content--header">

<h4 class="chatbox\_\_heading--header">Chat support</h4>

<p class="chatbox\_\_description--header">Hi. My name is ENO. How can I help you?</p>

</div>

</div>

<div class="chatbox\_\_messages">

<div></div>

</div>

<div class="chatbox\_\_footer">

<input type="text" placeholder="Write a message...">

<button class="chatbox\_\_send--footer send\_\_button">Send</button>

</div>

</div>

<div class="chatbox\_\_button">

<button><img src="./images/chatbox-icon.svg" /></button>

</div>

</div>

</div>

<script src="app.js"></script>

<script src="main.js"></script>

</body>

</html>

* **PYTHON**

import numpy as np

import random

import json

import torch

import torch.nn as nn

from torch.utils.data import Dataset, DataLoader

from nltk\_utils import bag\_of\_words, tokenize, stem

from model import NeuralNet

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

intents = json.load(f)

all\_words = []

tags = []

xy = []

# loop through each sentence in our intents patterns

for intent in intents['intents']:

tag = intent['tag']

# add to tag list

tags.append(tag)

for pattern in intent['patterns']:

# tokenize each word in the sentence

w = tokenize(pattern)

# add to our words list

all\_words.extend(w)

# add to xy pair

xy.append((w, tag))

# stem and lower each word

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

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

# remove duplicates and sort

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)

# create training data

X\_train = []

y\_train = []

for (pattern\_sentence, tag) in xy:

# X: bag of words for each pattern\_sentence

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

X\_train.append(bag)

# y: PyTorch CrossEntropyLoss needs only class labels, not one-hot

label = tags.index(tag)

y\_train.append(label)

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

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

# Hyper-parameters

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

# support indexing such that dataset[i] can be used to get i-th sample

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

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

# we can call len(dataset) to return the size

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)

# Loss and optimizer

criterion = nn.CrossEntropyLoss()

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

# Train the model

for epoch in range(num\_epochs):

for (words, labels) in train\_loader:

words = words.to(device)

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

# Forward pass

outputs = model(words)

# if y would be one-hot, we must apply

# labels = torch.max(labels, 1)[1]

loss = criterion(outputs, labels)

# Backward and optimize

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 complete. file saved to {FILE}')

* **JAVASCRIPT**

class Chatbox {

constructor() {

this.args = {

openButton: document.querySelector('.chatbox\_\_button'),

chatBox: document.querySelector('.chatbox\_\_support'),

sendButton: document.querySelector('.send\_\_button')

}

this.state = false;

this.messages = [];

}

display() {

const {openButton, chatBox, sendButton} = this.args;

openButton.addEventListener('click', () => this.toggleState(chatBox))

sendButton.addEventListener('click', () => this.onSendButton(chatBox))

const node = chatBox.querySelector('input');

node.addEventListener("keyup", ({key}) => {

if (key === "Enter") {

this.onSendButton(chatBox)

}

})

}

toggleState(chatbox) {

this.state = !this.state;

// show or hides the box

if(this.state) {

chatbox.classList.add('chatbox--active')

} else {

chatbox.classList.remove('chatbox--active')

}

}

onSendButton(chatbox) {

var textField = chatbox.querySelector('input');

let text1 = textField.value

if (text1 === "") {

return;

}

let msg1 = { name: "User", message: text1 }

this.messages.push(msg1);

fetch($SCRIPT\_ROOT + 'predict', {

method: 'POST',

body: JSON.stringify({ message: text1 }),

mode: 'cors',

headers: {

'Content-Type': 'application/json'

},

})

.then(r => r.json())

.then(r => {

let msg2 = { name: "Sam", message: r.answer };

this.messages.push(msg2);

this.updateChatText(chatbox)

textField.value = ''

}).catch((error) => {

console.error('Error:', error);

this.updateChatText(chatbox)

textField.value = ''

});

}

updateChatText(chatbox) {

var html = '';

this.messages.slice().reverse().forEach(function(item, index) {

if (item.name === "Sam")

{

html += '<div class="messages\_\_item--visitor">' + item.message + '</div>'

}

else

{

html += '<div class="messages\_\_item--operator">' + item.message + '</div>'

}

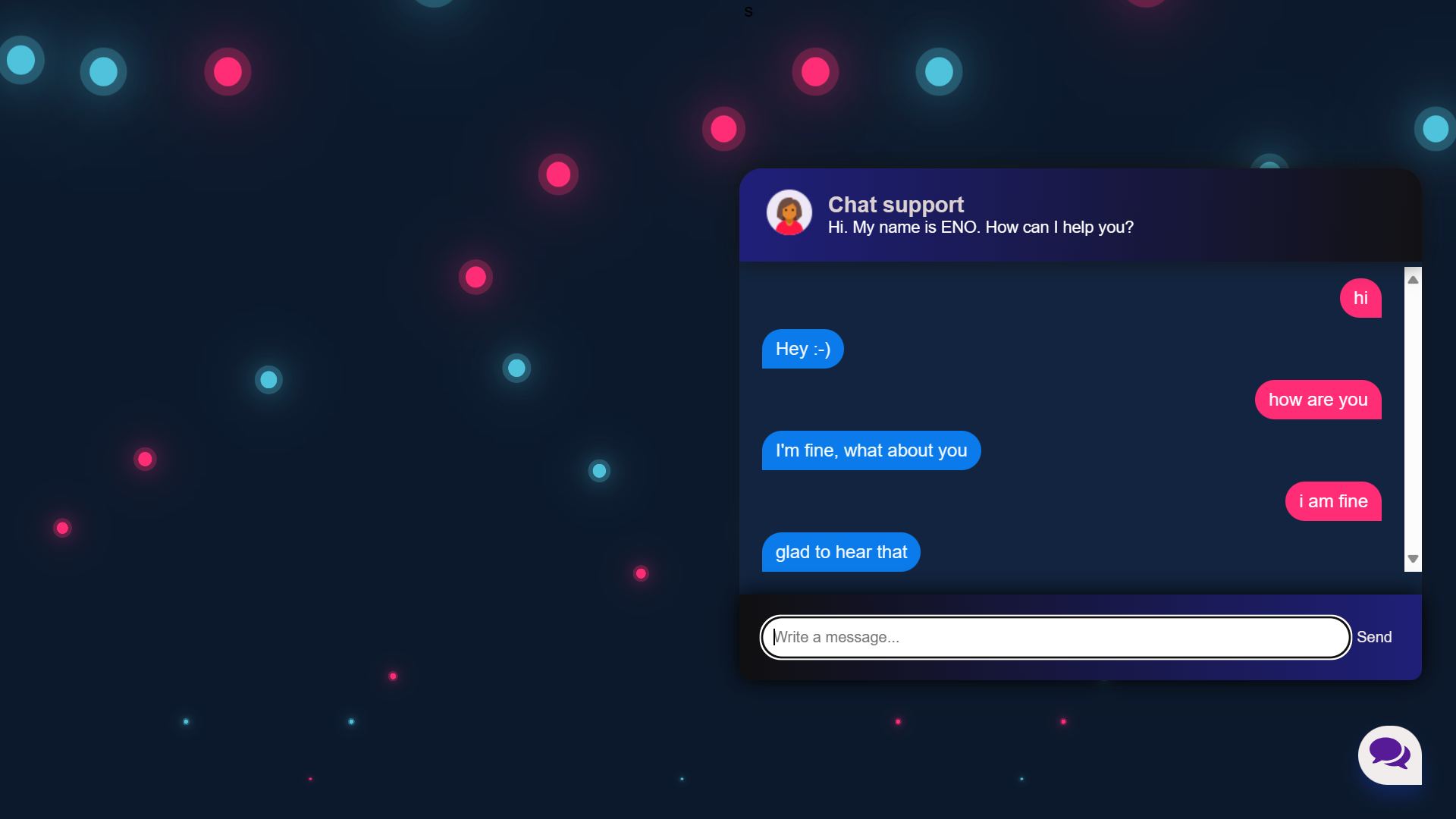
});

const chatmessage = chatbox.querySelector('.chatbox\_\_messages');

chatmessage.innerHTML = html;

}

**RESULT**



**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.

**REFERENCES**

1.https://www.google.com

2.https://www.geeksforgeeks.org

3.https://www.youtube.com