Vivekanand Education Society's Institute of Technology
Hashu Advani memorial Complex Collector's Colony
R C Marg, Chembur, Mumbai 400074

DEPARTMENT OF INFORMATION TECHNOLOGY

MINI PROJECT REPORT

ON

"Women Healthcare Chatbot"

B.E. (Information Technology)

SUBMITTED BY

Mr. Ninad Rao (57)

Mr. V Krishnasubramaniam (67)

Ms. Shalaka Waghamale (68)

UNDER THE GUIDANCE OF

PROF. Jitendra Madavi

(Academic Year: 2022-2023)

Mumbai University
Vivekanand Education Society's Institute Of Technology, Mumbai

DEPARTMENT OF INFORMATION TECHNOLOGY

Certificate

This is to certify that project entitled

"Women Healthcare Chatbot"

Mr. Ninad Rao (Roll No. 57)
Mr. V Krishnasubramaniam (Roll No. 67)
Ms. Shalaka Waghamale (Roll No. 68)

have satisfactorily carried out the project work, under the head - Data Science Lab at Semester VII of BE-IT in Information Technology as prescribed by the Mumbai University.

Prof. Jitendra Madavi | **External Examiner**

Dr.(Mrs.)Shalu Chopra
H.O.D

Dr.(Mrs.)J.M.Nair
Principal

Date: 20/10/2022
Place: VESIT, Chembur

LO Mapping

LO1: To apply reasoning for a problem in an uncertain domain.

LO2: To discuss the solution after building a Cognitive application.

LO3: To familiarize the students with the basics of Fuzzy Logic and Fuzzy Systems.

LO4: To familiarize the students with Learning Architectures and Frameworks.

LO5: To define and apply metrics to measure the performance of various learning algorithms.

LO6: To enable students to analyze data science methods for real world problems.

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original source. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

- - - - - - - - - - -
(Signature)

Mr
(Ninad Rao, 57)
B.E. INFT

- - - - - - - - - - -
(Signature)

Mr
(V Krishnasubramaniam, 67)
B.E. INFT

- - - - - - - - - - -
(Signature)

Ms
(Shalaka Waghamale, 68)
B.E. INFT

Contents

List of Figures

Abstract

Even today talking about women health is big taboo in India. The society is unbiased about women especially about their health issues hence they face disparity in the society. Women feel hesitant to talk about or speak about their health issues or problems openly. Thus, the goal of this chatbot is to help women to find information and remedial solutions about their health. Chatbots can be used to communicate with text or voice interfaces and receive responses via artificial intelligence. Chatbots are programs designed to automatically interact with incoming messages. Chatbots can be programmed to respond the same each time and respond differently to messages containing specific keywords. In addition, you can use machine learning to adapt your response to your situation. Query is processed by the bot and response will be displayed on web application. This chatbot will provide helpful information instantly. It will also provide prediction about the disease that the women might be suffering. So this bot will help to make right decision and give right advice to women on 24/7 basis. It will act as an helping hand for working women to keep check on their health in their busy routine. At the same time this will also help women in rural areas who are apprehensive to talk about their health issues publicly.

Keywords - ***chatbots, women, support, healthcare, solutions, artificial intelligence, machine learning***

Chapter 1

Introduction

1.1 Introduction

The idea of this system is to create a chatbot that can help girls and women find information and medicinal solutions about their health. Women sometimes tend to ignore their health problems due to their busy schedules and hectic lifestyles. Also normally ladies aren't aware of all the treatments regarding the diseases they have. Many of the diseases can be cured if they are treated early and have proper treatments at the early stages itself. This will even decrease the chances of having major health issues later on in life. On the other hand, sometimes ladies have to visit doctors even for some small problem which is time consuming. Such problems can be solved by using a medical chatbot which can give them proper guidance. Time to time guidance can help girls to remain updated about their health and have a routine check-up of their health. Girls or women can share their health-related problems or symptoms to the bot and the bot will provide proper guidance according to the symptoms they have.

1.2 Problem Statement

The health of Indian women is intrinsically linked to their status in society. Research on women's status has found that the contributions Indian women make to families are often overlooked, and instead, they are viewed as economic burdens. Further, Indian women have low levels of both education and formal labor force participation. They typically have little autonomy, living under the control of first their fathers, then their husbands, and finally their sons. All of these factors exert a negative impact on the health status of Indian women.

1.3 Objectives of the project

A medical chatbot facilitates the job of a healthcare provider and helps improve their performance by interacting with users in a human-like way. There are countless cases where intelligent medical chatbots could help physicians, nurses, therapists, patients, or their families. They can step in and minimize the amount of time they spend on tasks like:

1. Providing health-related information to users

2. Guidance for patient
3. Medication management and dosage
4. Connecting people and organizations with first responders
5. FAQ-type queries (contact details, directions, opening hours and service/treatment details)

It's important to note that despite the fact that chatbots can offer valuable facts and symptoms, they aren't qualified to give an official diagnosis. The main premise behind these talking or texting smart algorithms is to become the first point of contact before any human involvement is needed.

1.4 Functionalities

The healthcare industry is no stranger to emergencies. And time plays a very crucial role in tackling them! Healthcare chatbots provide helpful information instantly, especially in times where every second is important.

1. Serve as 24/7 companions and monitor health status in real-time.
2. Help manage chronic conditions, mental health issues, and behavioral and psychological disorders.
3. Proactively identify symptoms, crosscheck them against medical history, suggest the next steps, and improve the treatment success rate in cases where early diagnosis can play a critical role.
4. Make self-care easier by acting as a virtual assistant and providing timely medical advice.

1.5 Scope

Healthcare industry has a significant shortfall of qualified medical professionals, therefore, it takes time for patients to receive one-on-one consultations. Chatbots in healthcare can provide initial assistance over one-on-one conversations after analyzing patient data. As a healthcare organization gathers more data, it can train the AI chatbot better using Machine Learning (ML). This enables the chatbot to provide more relevant information to the patients.

Nowadays, patients want more information about their medical conditions and treatments, e.g., they might need information about generic and prescription drugs. Chatbots in healthcare have emerged as useful tools for a healthcare provider to provide additional information for patients' satisfaction.

Chapter 2

Analysis and Design

2.1 Analysis of the system

Today's AI systems can interact with users, understand their needs, map their preferences and recommend an appropriate line of action with minimal or no human intervention. The basic foundation of chatbots is providing the best response of any query that it receives. The best response like answering the sender questions, providing sender relevant information, ask follow-up questions and do the conversation in realistic way.

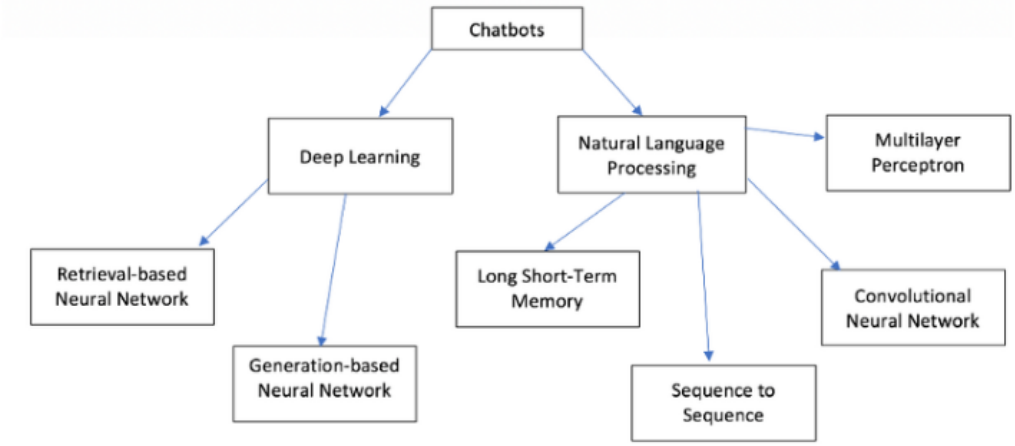

Figure 2.1: Various ways of building a chatbot

The chatbot needs to be able to understand the intentions of the sender's message, determine what type of response message (a follow-up question, direct response, etc.) is required, and follow correct grammatical and lexical rules while forming the response. Some models may use additional meta information from data, such as speaker id, gender, emotion. Sometimes, sentiment analysis is used to allows the chatbot to 'understand' the mood of the user by analysing verbal and sentence structuring clues.

2.2 Design of the proposed system

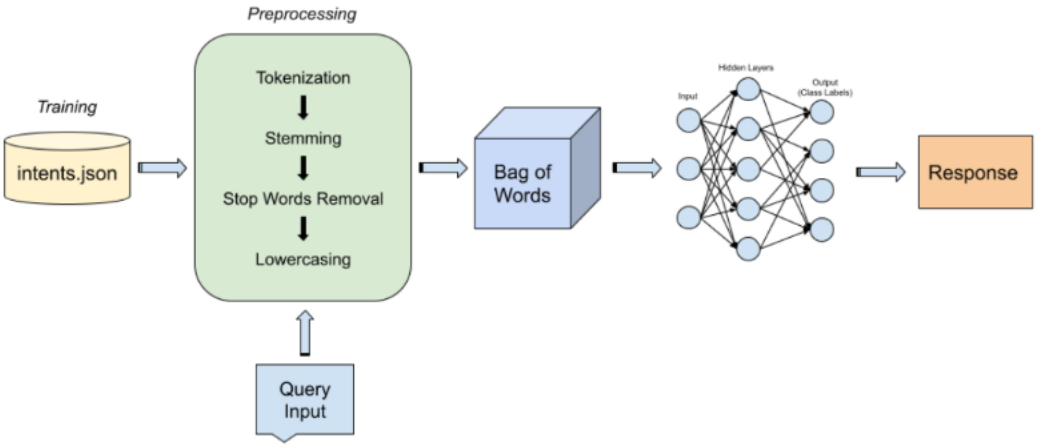

Figure 2.2: Design of the system

Intents.json
The frequently asked questions regarding the health and safety are identified and classified into multiple tags of questions. An Intent is programmatically a JSON object for managing the tags and each Intent object has the following structure:

```
{
      "tag": "common cold symptoms",
      "patterns": [
        "Runny or stuffy nose",
        "Sore throat",
        "Cough",
        "Congestion",
        "Slight body aches or a mild headache",
        "Sneezing",
        "Low-grade fever",
        "Generally feeling unwell (malaise)"
      ],
      "responses": ["It seems that you are suffering from common cold"]
},
```

Each intent object has a ”tag” which acts as the class label for the ML model. The Model tries to classify the input given by the user into these class label tags using the Dictionary based approach.

Preprocessing

1. Tokenization: Splitting string into meaningful units like words, punctuation, numbers etc.

2. Stemming: Generate the root form of the words. Crude heuristic to chop off the ends of the words.

Training Model
Extract all words, tags, X,y pairs from the intents json into different arrays.

1. For each intent, store the tag, loop over all patterns by tokenizing the patterns, adding all these words in all_words and storing these words and the tag as the X,

y pair and from all_words, remove unnecessary words, apply stemming to them and sort them in a set (unique words only).

2. Loop over the X, y pairs by getting the bag of words for each X and append them to X_train and appending y to y_train.
3. Convert X_train and y_train into numpy arrays.
4. Creating dataset class by extending pytorch Dataset, setting the data members to X_train and y_train and override the getitem and len functions.
5. Load dataset using DataLoader and create a new NeuralNet class using pytorch's torch.nn.Module. Get the bag of words as input, create 2 Hidden layers of different possible patterns progressively inputting them and output this model.
6. Use this NeuralNet model class and serialize and save the model data using torch.save

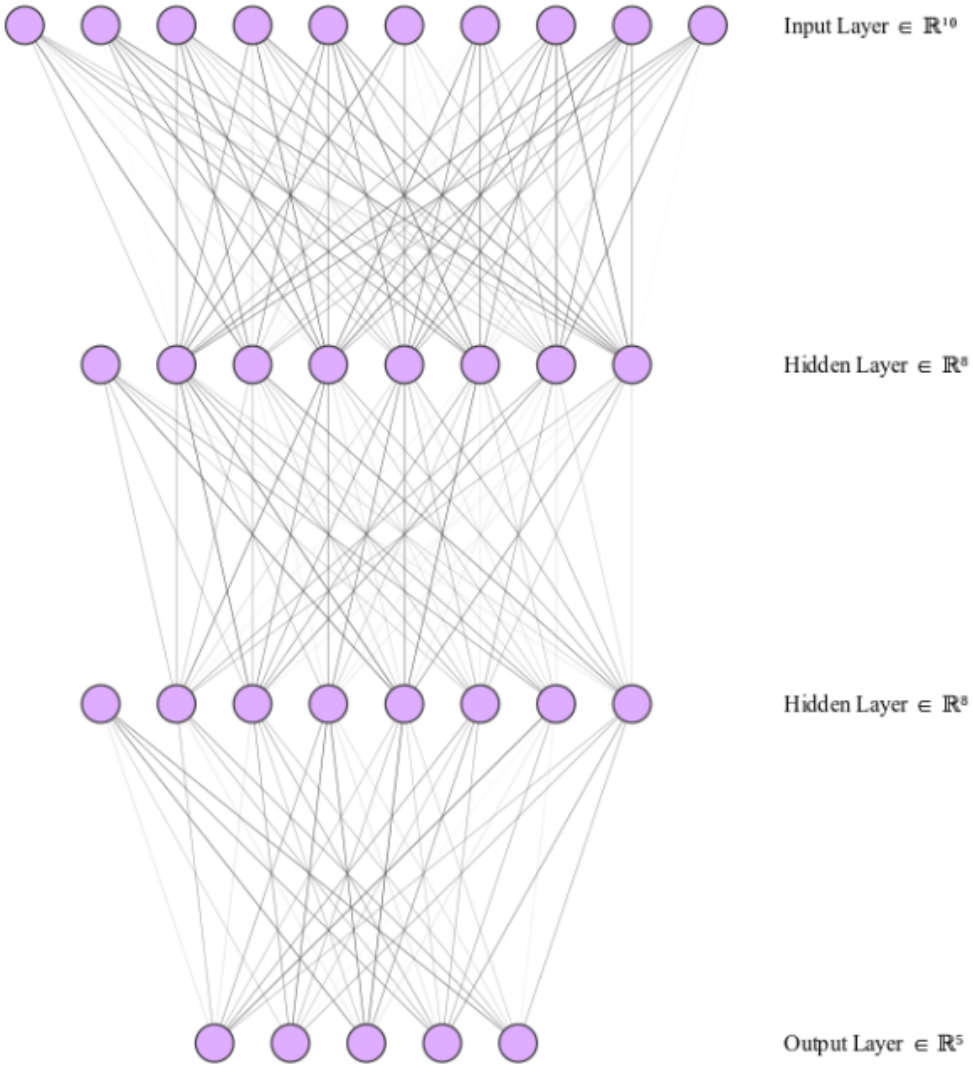

Figure 2.3: Flow of Training the Model

Using the saved model

1. Load the data from the model using torch.load
2. Use the data to create and evaluate the NeuralNet model
3. Get the input sentence from user
4. Tokenize and create bag of words from this
5. Get the predicted class label from this
6. Choose a random output for this predicted class from the intents file, only if the probability of this prediction is greater than 75

Input query
Query taken from the user is also passed through the same preprocessing steps

2.2.1 Architecture/ Block Diagram

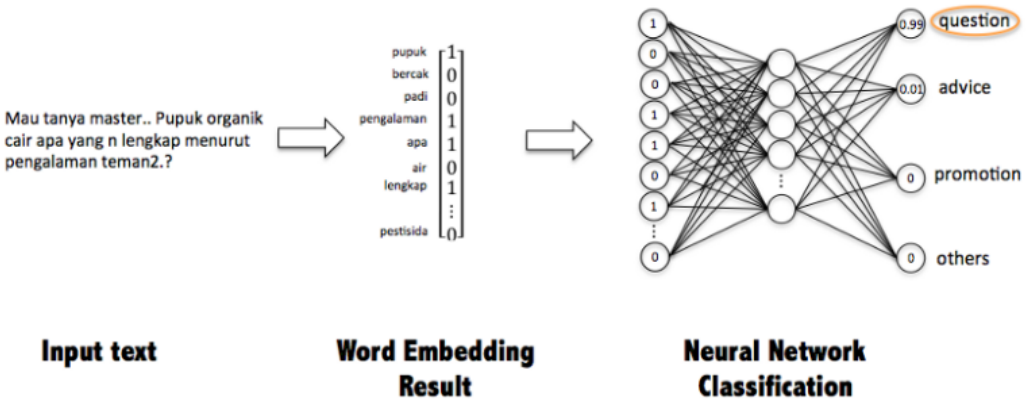

Figure 2.4: Architecture of the system

2.2.2 Algorithms Used

Feed Forward Neural Network
A Feed Forward Neural Network is an artificial neural network in which the connections between nodes does not form a cycle. The opposite of a feed forward neural network is a recurrent neural network, in which certain pathways are cycled. The feed forward model is the simplest form of neural network as information is only processed in one direction. While the data may pass through multiple hidden nodes, it always moves in one direction and never backwards.

Often referred to as a multi-layered network of neurons, feedforward neural networks are so named because all information flows in a forward manner only. The data enters the input nodes, travels through the hidden layers, and eventually exits the output nodes. The network is devoid of links that would allow the information exiting the output node to be sent back into the network.

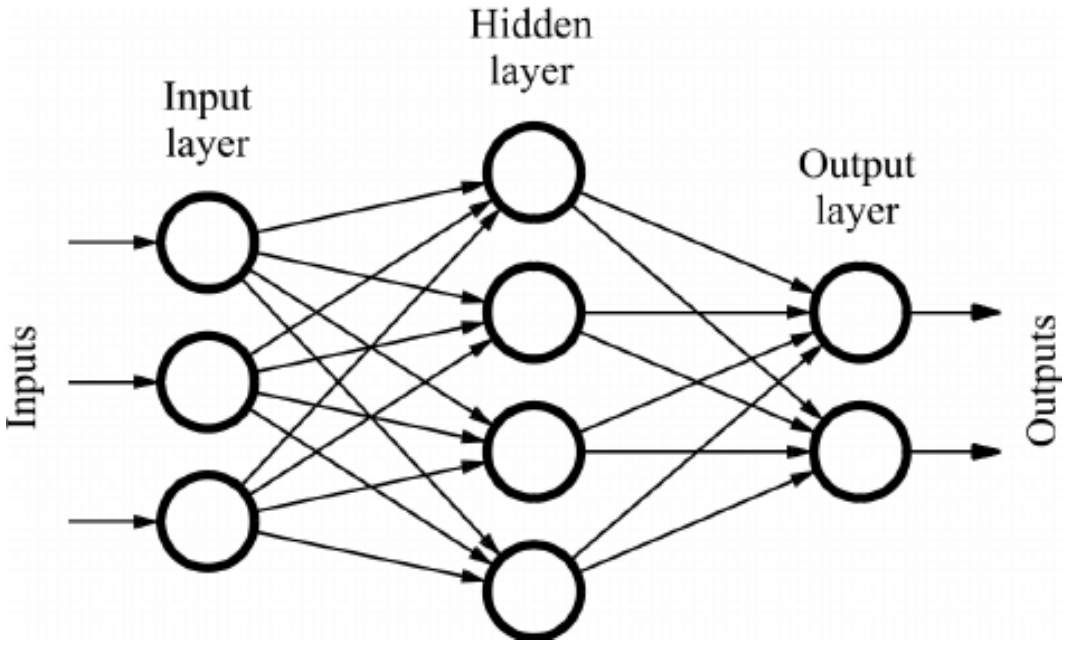

Figure 2.5: Architecture of the system

Natural Language Processing
Natural language processing (NLP) refers to the branch of computer science and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can. NLP combines computational linguistics rule-based modeling of human language with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning, complete with the speaker or writer's intent and sentiment.

Rectified Linear Unit
The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. This function can be represented as:

```
f(x) = max(0, x)
```

where x is an input value. According to above equation, the output of ReLu is the maximum value between zero and the input value. An output is equal to zero when the input value is negative and the input value when the input is positive.

2.2.3 Details of Hardware & Software

We are using the following technical stack for developing the chatbot:

1. Flask (Backend Framework)

2. HTML, CSS and JavaScript

3. Python (Model Creation)

2.3 Tools and Datasets

PyTorch
PyTorch is an optimized Deep Learning tensor library based on Python and Torch and is mainly used for applications using GPUs and CPUs. PyTorch is favored over other Deep Learning frameworks like TensorFlow and Keras since it uses dynamic computation graphs and is completely Pythonic. It allows scientists, developers, and neural network debuggers to run and test portions of the code in real-time. Thus, users don't have to wait for the entire code to be implemented to check if a part of the code works or not.

The two main features of PyTorch are:

1. Tensor Computation (similar to NumPy) with strong GPU (Graphical Processing Unit) acceleration support.

2. Automatic Differentiation for creating and training deep neural networks.

The basic idea behind developing the PyTorch framework is to develop a neural network, train, and build the model. Pytorch uses a torch.nn base class which can be used to wrap parameters, functions, and layers in the torch.nn modules. Any deep learning model is developed using the subclass of the torch.nn module it uses method like forward(input) which returns the output. A simple neural network takes input to add weights and bias to it feed the input through multiple hidden layers and finally returns the output.

Dataset
In order to create a more effective chatbot, one must first compile realistic, task-oriented dialog data to effectively train the chatbot. Without this data, the chatbot will fail to quickly solve user inquiries or answer user questions without the need for human intervention. Actual conversations on online forums are scraped and used to create the intents json file.

2.3.1 Experimental Results (Code and GUI)

Code
Following are the codes for creating the model, training the model and working of the model.

train.py

```
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
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()
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 = [stemmer.stem(w.lower()) 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}")
```

model.py

```
    import torch
    import torch.nn as nn

    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)
            # no activation and no softmax at the end
            return out
```

chat.py

```
    import random
    import json
```

```
import torch
from model import NeuralNet
from nltk_utils import bag_of_words, tokenize

device = torch.device('cuda' if torch.cuda.is_available()
else 'cpu')
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_name = "Sam"

def get_response(msg):
    sentence = tokenize(msg)
    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"]:
                return random.choice(intent['responses'])
    return "I do not understand your query."

if __name__ == "__main__":
    print("Let's chat! (type 'quit' to exit)")
    while True:
        sentence = input("You: ")
        if sentence == "quit":
            break
        resp = get_response(sentence)
        print(resp)
```

Frontend UI Screenshots
Following are the screenshots of the UI developed for the chatbot.

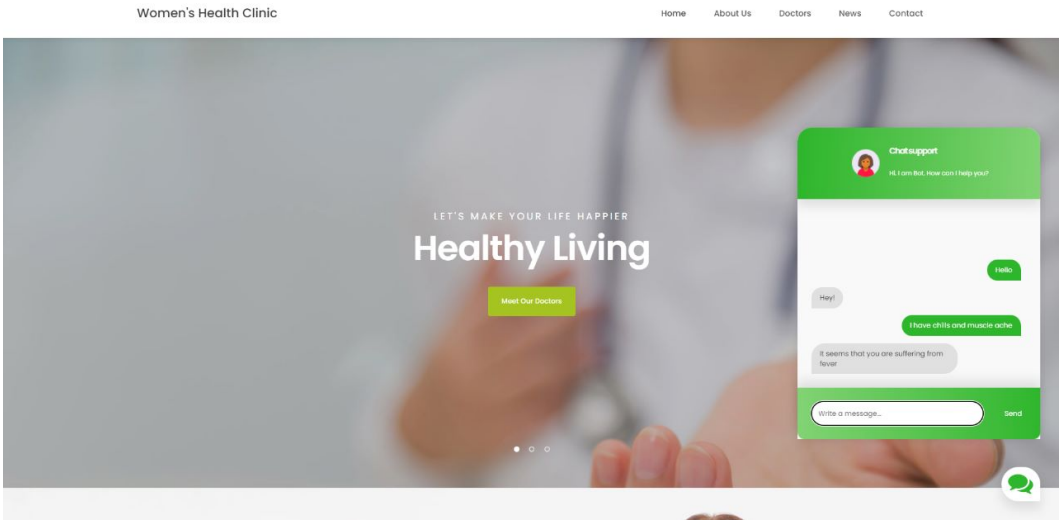

Figure 2.6: UI of the Chatbot

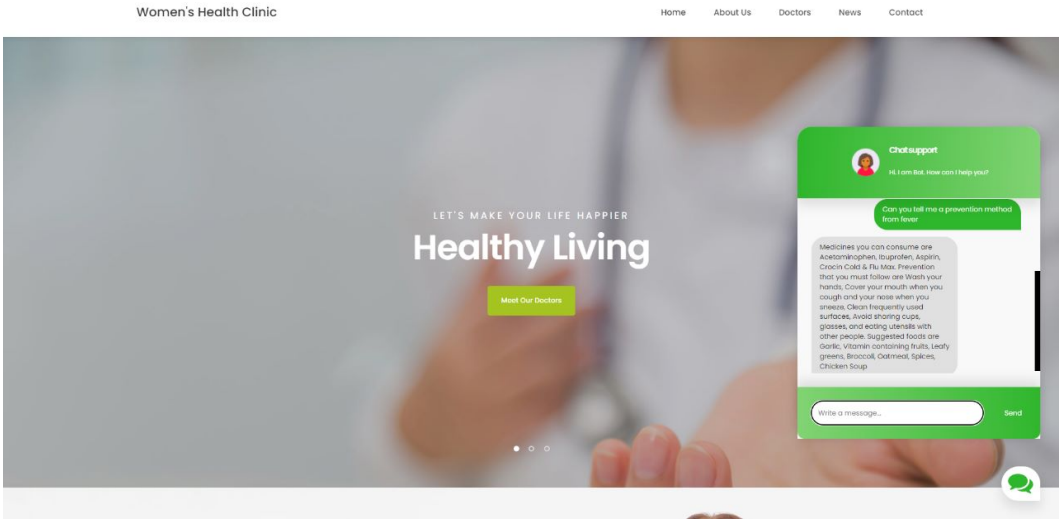

Figure 2.7: Query asked in the chatbot

Chapter 3

Conclusion and Future Work

Our medical chatbot provides medical help to the patients for a few general diseases like fever, cold, typhoid, malaria, jaundice, etc. we have a tendency to area unit inventing the system owing to the requirement for the increasing population of our country. Such systems area unit obtainable in foreign however not in our country. As we all know well regarding it that the numbers of doctors area unit less to serve the requirement of the patient. This state of affairs may be higher understood by walking through the city's government hospitals. Thus, the web application does the implementation of women health chatbot where the girl user can visit the web application at any time and ask question about her symptoms and gets disease prediction. This application can help women to remain updated about their health issues and will prevent them from having any major health issues in their future life.

References

[1] Polekar, S., Wakde, S., Pandare, M., & Shingane, P. (2022). Intelligent Medical Chatbot System For Women's Healthcare. *ITM Web of Conferences, 44,* 03020. https://doi.org/10.1051/itmconf/20224403020

[2] Tejashwini, U., Sushma, H., Panitha, K., & PoojaShri, G. H., (2020, May). Medical Chatbot for Women Health. *International Research Journal of Engineering and Technology (IRJET).* e-ISSN: 2395-0056

[3] Shinde, N. V., Akhade, A., Bagad, P., Bhavsar, H., Wagh, S., & Kamble, A. (2021, June 3). Healthcare Chatbot System using Artificial Intelligence. *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI).* https://doi.org/10.1109/icoei51242.2021.9452902