TRIUMPH TIDINGS: A MENTAL HEALTH CHATBOT

Dissertation submitted in fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION

IN

DATA SCIENCE AND MACHINE LEARNING

By

PRANJAL SINHA

REGISTRATION NUMBER- 12206214

SECTION- K22UN

ROLL NUMBER- RK22UNA31

Supervisor

Ved Prakash Chaubey

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Month: October, Year: 2023

Abstract

Mental health is a major public health concern, with millions of people suffering from depression, anxiety, and other mental health conditions. Chatbots have the potential to provide a valuable tool for supporting mental health, by offering confidential and accessible support. This project presents the development of a mental health chatbot that can be used to provide emotional support and resources to people who are feeling lonely, depressed, or anxious.

The chatbot was developed using a combination of machine learning and natural language processing techniques. The chatbot is trained on a dataset of conversations between people who are struggling with mental health challenges and mental health professionals. This training allows the chatbot to learn how to respond to user input in a way that is both supportive and informative.

The chatbot can be used by people of all ages and backgrounds. It is easy to use and does not require any prior knowledge of mental health conditions. The chatbot can be accessed through a variety of platforms, including web chat, mobile app, and SMS.

The chatbot is designed to be used as a supplement to professional mental health care. It is not intended to diagnose or treat mental health conditions. However, the chatbot can provide valuable support and resources to people who are struggling with mental health challenges.

DECLARATION BY THE SCHOLAR

I hereby declare that the research work reported in the dissertation/dissertation proposal entitled "**TRIUMPH TIDINGS: A MENTAL HEALTH CHATBOT**" in partial fulfilment of the requirement for the award of Degree for Bachelor of Technology in Computer Science and Engineering with Specialization in Data Science and Machine Learning at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Ved Prakash Chaubey. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Signature of Candidate

PRANJAL SINHA

12206214

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B. Tech Dissertation/dissertation proposal entitled "**TRIUMPH TIDINGS: A MENTAL HEALTH CHATBOT"**, submitted by **PRANJAL SINHA** at **Lovely Professional University, Phagwara, India** is a bonafide record of his original work carried out under my supervision. This work has not been submitted elsewhere for any other degree.

Signature of Supervisor

Ved Prakash Chaubey

Date:

Counter Signed by:

1) **Concerned HOD:**
   HoD's Signature: _____

   HoD Name: _____

   Date: _____
2) **Neutral Examiners:**

   External Examiner

   Signature: _____

   Name: _____

   Affiliation: _____

   Date: _____

   Internal Examiner

   Signature: _____

   Name: _____

   Date: _____

Acknowledgement

I extend my deepest gratitude to all those whose support and contributions have made this project a reality. First and foremost, I would like to express my sincere appreciation to the academic community for fostering an environment of learning and innovation. Your dedication to knowledge and research has been a constant source of inspiration.

I am profoundly thankful to my mentors and advisors whose guidance and expertise have been invaluable throughout this journey. Your wisdom, encouragement, and unwavering support have shaped this project and my understanding of the subject matter. I am indebted to you for your mentorship and insights.

I am grateful to the open-source community and developers worldwide for creating robust frameworks, libraries, and tools that formed the foundation of this project. Your collective efforts have democratized knowledge and technology, enabling aspiring researchers like me to explore innovative solutions.

Lastly, I would like to express my appreciation to the participants, users, and all individuals who engaged with the project, providing valuable insights and feedback. Your involvement has been instrumental in shaping the project's direction and ensuring its relevance in real-world scenarios.

In conclusion, I am profoundly thankful to each and every individual who has played a role, no matter how big or small, in the realization of this project. Your contributions have left an indelible mark, and I am deeply appreciative of your support and encouragement.

PRANJAL SINHA

TABLE OF CONTENTS

TABLE OF CONTENTS

1) INTRODUCTION

1.1) BACKGROUND

1.1.1) LONELINESS IN HIGHER EDUCATION

In the bustling corridors of higher education, where knowledge intertwines with ambition, a silent epidemic pervades the lives of countless students: loneliness. The college experience, often hailed as a time of intellectual growth and personal discovery, can be paradoxically isolating. For many, the transition from the familiarity of home to an unfamiliar campus in a different city or country can trigger profound feelings of solitude. This phenomenon is not limited by geographical boundaries; rather, it reverberates in dormitories, lecture halls, and communal spaces worldwide.

Loneliness, if left unaddressed, can metamorphose into a formidable adversary, gnawing at the core of mental well-being. It breeds feelings of alienation, anxiety, and melancholy, and in its darkest corners, it can pave the way for severe mental health issues. College life, with its academic pressures, social expectations, and newfound responsibilities, can magnify the impact of loneliness, rendering even the most resilient souls vulnerable.

The Significance of the Problem:

The importance of addressing this issue cannot be overstated. Loneliness not only affects the individual's mental health but also seeps into the fabric of the community, creating a collective sense of disconnection. This estrangement can hinder academic performance, social integration, and overall happiness, making it imperative to find innovative solutions that transcend the barriers of physical distance.

1.1.2) PROBLEM STATEMENT

The alarming rise in mental health issues, fueled by the challenges of modern life, necessitates innovative solutions.

A significant problem lies in the lack of accessible and personalized mental health support. Traditional therapy methods face limitations due to high costs, stigma, and limited availability. Furthermore, many individuals, especially students and young adults, struggle with feelings of loneliness and anxiety, particularly due to being away from home and the constant fear of future.

TriumphTidings addresses this critical gap by offering an AI-powered mental health chatbot. This chatbot provides an empathetic, understanding presence, guiding users through their emotions, promoting self-awareness, and delivering coping mechanisms. The project's primary goal is to offer immediate, stigma-free, and effective mental health support to those in need, ensuring no one faces their battles alone.

1.1.3) MOTIVATION AND INNOVATION

i) The Motivation Behind Triumph Tidings:

The genesis of Triumph Tidings lies in the shared experiences of countless students who have grappled with loneliness. As I found myself thousands of kilometers away from home, the pangs of isolation became palpable. Yet, amidst this struggle, a vision emerged — a vision of crafting a digital companion capable of mitigating the emotional turbulence faced by students worldwide.

ii) Triumph Tidings as a Supportive Companion:

Triumph Tidings is not a mere technological endeavor; it is a manifestation of empathy and understanding. It is a response to the silent cries for connection echoing in the dormitories and common rooms. By harnessing the power of artificial intelligence, natural language processing, and machine learning, Triumph Tidings aspires to become more than just a chatbot. It is conceived as a virtual confidante, a steadfast companion that listens without judgment, speaks with compassion, and offers unwavering support.

iii) The Ripple Effect of Triumph Tidings:

Beyond the individual impact, Triumph Tidings is poised to create a ripple effect. By alleviating the burden of loneliness, it nurtures a sense of belonging and camaraderie. It fosters resilience, empowers individuals to face challenges head-on, and provides a ray of hope in the face of despair. As students find solace in the digital realm, they are better equipped to navigate the complexities of college life, forging meaningful connections and embracing their academic journey with newfound vigor.

Link to Github Repository:-

https://github.com/pranjalsinha1205/MentalHealthChatbotTriumphTidings

Or scan this QR:-

2) REVIEW OF LITERATURE

The field of mental health chatbots has witnessed significant advancements in recent years. Several notable projects have emerged, aiming to address the pressing issue of mental health and provide accessible support to individuals. One such example is **Woebot**, an AI-powered chatbot developed by psychologists and AI experts. **Woebot** utilizes cognitive-behavioral therapy techniques to engage users in interactive conversations, offering personalized emotional support and coping strategies.

Wysa, another prominent mental health chatbot, integrates AI with evidence-based therapeutic techniques. It focuses on emotional well-being, guiding users through mindfulness exercises, mood tracking, and relaxation techniques. **Replika**, an AI chatbot designed as a personal AI companion, provides users with a virtual friend for open conversations, thereby reducing feelings of loneliness and providing a non-judgmental space for self-expression.

By analyzing these existing solutions, **TriumphTidings** aims to draw inspiration and incorporate the best practices into its design. **TriumphTidings** distinguishes itself through its focus on not only providing empathetic responses but also encouraging users to take an active role in managing their mental health, fostering a sense of empowerment and resilience.

3) PRESENT WORK

3.1) PROBLEM FORMULATION

In the context of this project, the problem of loneliness among college students was identified as a pressing issue. Many students experience a sense of isolation despite the bustling environment of higher education. The aim was to create a mental health chatbot that addresses this problem by providing emotional support and resources to individuals feeling lonely, depressed, or anxious. The goal was to leverage Natural Language Processing (NLP) and machine learning techniques to develop an empathetic AI chatbot capable of understanding user emotions and intents.

3.2) OBJECTIVES OF THE STUDY

The primary objectives of this study were:

1) To develop a chatbot using Natural Language Processing (NLP) techniques.

2) To analyze user inputs, identify emotions and intents, and provide appropriate responses.

3) To create a confidential and accessible platform for individuals struggling with mental health issues.

4) To contribute to the ongoing efforts in the field of mental health support through innovative technology solutions.

3.3) RESEARCH METHODOLOGY

3.3.1) APPROACH

TriumphTidings employs Natural Language Processing (NLP) and machine learning techniques to create an empathetic AI chatbot. Utilizing NLTK for language processing, the chatbot analyzes user inputs, identifying emotions and intents.

This model is trained on a dataset containing categorized patterns and responses.(Dataset taken from Kaggle, link: https://www.kaggle.com/datasets/elvis23/mental-health-conversational-data) a

The training enables the bot to understand and respond contextually. The graphical user interface (GUI) is built using Flask, enhancing user experience. TriumphTidings stands out for its personalized approach, tailoring responses to individual emotional states.

Additionally, it incorporates features for recommending stress-buster exercises, fostering user engagement. The approach ensures a holistic, user-friendly, and effective mental health support system, promoting emotional well-being.

Also, it give a graph showing the polarity trends of the user which in turn will help the medical or psychiatric practitioners to further understand the user's mental health at least upto some extent.

3.3.2) ALGORITHMS

1) **LSTM (Long Short-Term Memory):** LSTM neural network is used for sequence modeling and prediction of user intents from their input.

2) **Word Embedding:** Words are represented as dense vectors to capture semantic relationships, enhancing the model's understanding of language nuances.

3) **Tokenization:** The process of breaking down text into words or subwords, essential for preparing text data for analysis.

4) **Lemmatization:** Lemmatization is the process of reducing words to their base or root form, ensuring different forms of words are treated as the same, aiding in better analysis.

5) **Vader:** Used to assign polarity between -1 and 1 to each and every sentence written by the user, if the polarity is >0.5, the user is +ve, if it is <0.5 the user is negative, and anything between that is neutral or partially positive or negative

These algorithms collectively contribute to the chatbot's ability to process user input, understand context, and generate appropriate responses.

3.3.3) ALGORITHM EXPLANATION

1) Tokenization and Lemmatization:

i) **Intent Data Preparation**:

The chatbot loads intent data from a JSON file, containing patterns and corresponding responses for various user intents.

ii) **Tokenization and Lemmatization**:

a) User input, as well as patterns from intents, are tokenized into words to break down sentences into individual components.
b) The NLTK library is employed to lemmatize the words, reducing them to their base or root form, ensuring uniformity and enhancing analysis.

c) These processed words are organized into a bag of words, forming the basis for understanding user input patterns.

2) Neural Network Model Construction:

i) **Architecture Setup:**

a) A Neural Network model is constructed using the Keras library, comprising input, hidden, and output layers.

b) Input layer: Accepts numerical vectors derived from the bag of words representing input patterns.

c) Hidden layers: These layers process and learn complex patterns within the input data, enhancing the model's understanding.

d) Output layer: Produces probabilities for each intent class, indicating the likelihood of the user's input corresponding to a specific intent.

ii) **Training Process:**

a) Input patterns are converted into numerical vectors to be used for training the Neural Network model.

b) The model is trained using categorical cross-entropy loss function and the Adam optimizer, iterating through epochs to refine its accuracy.

3) User Input Processing:

i) **Preprocessing User Input:**

a) User input is tokenized and lemmatized to ensure consistency and uniformity.

b) The processed input is converted into a bag of words, aligning it with the format used during training.

ii) **Intent Prediction:**

a) The trained Neural Network model predicts the intent class probabilities for the preprocessed user input.

b) If the highest predicted probability exceeds a predefined threshold (ERROR_THRESHOLD), the corresponding intent is identified as the user's intent.

4) Generating Responses:

i) **Intent-Based Response Selection:**

a) If a valid intent is recognized based on the predicted probabilities exceeding the threshold:

b) A relevant response is randomly selected from the pre-defined responses associated with that intent.

ii) **Fallback to Default Response:**

a) If no valid intent is found (probabilities below the threshold):

a.1) A default response, indicating a lack of understanding, is generated, ensuring a response even in ambiguous situations.

This modular approach allows the chatbot to process user input effectively, predict intents accurately, and generate contextually appropriate responses, enhancing the user experience and interaction quality.

5) **Sentiment Analysis:** Sentiment analysis is performed using the VADER (Valence Aware Dictionary and Sentiment Reasoner) tool, tailored for social media text analysis. The process involves:

i) **Intent Data Preparation:**

Intent data is loaded from a JSON file, containing user patterns and responses.

ii) **Tokenization and Sentiment Analysis:**

a) User input is tokenized into sentences for analysis.

b) The NLTK library is used for tokenization and lemmatization.

c) Sentiment analysis with VADER assigns a compound score to assess the text's overall sentiment.

iii) **Data Visualization:**

Sentiment analysis results, including compound scores, are plotted against message numbers using Matplotlib. This graphical representation provides insights into the conversation's emotional tone over time.

4) CODES

4.1) MAIN CODE

```
import nltk   # Import the NLTK library for natural language processing
import numpy as np  # Import NumPy for scientific calculations
import json  # Import JSON for handling JSON files
import random  # Import random for generating random responses
from keras.models import Sequential  # Import Keras for building the neural network model
from keras.layers import Dense, Embedding, LSTM, Dropout
from nltk.stem import WordNetLemmatizer  # Import WordNetLemmatizer for word lemmatization

# Load intents file
with open('intents.json', 'r') as file:  # Open and read the intents JSON file
    intents = json.load(file)  # Load the JSON data into the 'intents' variable

# Extract words, classes, and documents from intents file
words = []  # Initialize an empty list to store tokenized words
classes = []  # Initialize an empty list to store intent classes
documents = []  # Initialize an empty list to store patterns and corresponding intents
ignore_words = ['?', '!']  # Define a list of punctuation marks to ignore
```

```
lemmatizer = WordNetLemmatizer()  # Initialize WordNetLemmatizer object for word lemmatization


# Iterate through intents and patterns to tokenize words and extract classes
for intent in intents['intents']:
    for pattern in intent['patterns']:
        # Tokenize words in the pattern
        w = nltk.word_tokenize(pattern)
        words.extend(w)  # Add tokenized words to the 'words' list
        documents.append((w, intent['tag']))  # Add patterns and corresponding intents to 'documents'
        if intent['tag'] not in classes:
            classes.append(intent['tag'])  # Add intent classes to 'classes' if not already present


# Lemmatize words and remove duplicates and sort the lists
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]  # Lemmatize and remove duplicates
words = sorted(list(set(words)))  # Sort the list of words


# Sort the list of classes
classes = sorted(list(set(classes)))


# Create training data
training = []  # Initialize an empty list to store training data
```

```
output_empty = [0] * len(classes)  # Create a list of zeros with length equal to the number of classes


# Create a bag of words for each pattern and corresponding intent
for doc in documents:
    bag = []  # Initialize an empty list to store the bag of words for the pattern
    pattern_words = doc[0]  # Get words from the pattern
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]  # Lemmatize words
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)  # Create a binary bag of words for the pattern


    output_row = list(output_empty)  # Create a copy of the list with zeros
    output_row[classes.index(doc[1])] = 1  # Set the index corresponding to the intent class to 1
    training.append([bag, output_row])  # Add the bag of words and intent class to the training data


# Pad the sublists to make them equal in length
max_length = len(max(training, key=lambda x: len(x[0]))[0])  # Find the length of the longest bag of words
for sublist in training:
    sublist[0] += [0] * (max_length - len(sublist[0]))  # Pad the bag of words with zeros to match the length
random.shuffle(training)  # Shuffle the training data for randomness
```

```
training = np.array(training, dtype=object)  # Convert the training data to a NumPy array

# Split data into training and testing sets
train_x = list(training[:, 0])  # Get the bag of words as the input data
train_y = list(training[:, 1])  # Get the corresponding intent classes as the output data

# Build the neural network model
model = Sequential()  # Create a sequential model
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))  # Add a dense layer with ReLU activation
model.add(Dropout(0.5))  # Add dropout regularization to prevent overfitting
model.add(Dense(64, activation='relu'))  # Add another dense layer with ReLU activation
model.add(Dropout(0.5))  # Add dropout regularization
model.add(Dense(len(train_y[0]), activation='softmax'))  # Add a dense layer with softmax activation for output

# Compile the model with categorical cross-entropy loss and Adam optimizer
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model to the training data for 100 epochs with batch size 5
model.fit(np.array(train_x), np.array(train_y), epochs=100, batch_size=5, verbose=1)
```

```
# Function to clean and tokenize user input
def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)  # Tokenize words in the sentence
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]  # Lemmatize words
    return sentence_words  # Return the cleaned and tokenized words

# Function to create a bag of words from user input
def bow(sentence, words, show_details=True):
    sentence_words = clean_up_sentence(sentence)  # Clean and tokenize user input
    bag = [0] * len(words)  # Initialize a bag of words with zeros
    for s in sentence_words:
        for i, w in enumerate(words):
            if w == s:
                bag[i] = 1  # Set the corresponding index to 1 if word is in the bag
                if show_details:
                    print(f"Found in bag: {w}")  # Print debug information if show_details is True
    return np.array(bag)  # Return the bag of words as a NumPy array

# Function to predict the class of the user input and get a response
def predict_class(sentence):
    p = bow(sentence, words, show_details=False)  # Get the bag of words for the user input
```

```
    res = model.predict(np.array([p]))[0]  # Use the trained model to predict the intent probabilities
    ERROR_THRESHOLD = 0.25  # Define an error threshold for intent classification
    results = [[i, r] for i, r in enumerate(res) if r > ERROR_THRESHOLD]  # Filter intents above the threshold


    results.sort(key=lambda x: x[1], reverse=True)  # Sort intents by probability in descending order
    return_list = []  # Initialize an empty list to store intent predictions
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})  # Add intent and probability to list
    return return_list  # Return the list of intent predictions


# Function to handle user input and generate a response
def get_response(user_input):
    response = "I'm sorry, I don't understand."  # Default response if intent is not recognized
    ints = predict_class(user_input)  # Get intent predictions for the user input
    tag = ints[0]['intent']  # Get the most probable intent
    for intent in intents['intents']:
        if intent['tag'] == tag:
            response = random.choice(intent['responses'])  # Get a random response from the matched intent
            break  # Exit the loop after finding the matched intent
    return response  # Return the generated response to the user input
```

4.2) FLASK CODE

```
from flask import Flask, render_template, request
# Import the necessary classes and functions from Flask framework:
# render_template: Function to render HTML templates and pass data to them.
# request: Object to handle incoming request data, such as form data.
from nltk.sentiment.vader import SentimentIntensityAnalyzer  # Import SentimentIntensityAnalyzer from NLTK's vader module
import matplotlib.pyplot as plt  # Import matplotlib for creating plots

app = Flask(__name__)

# Import your chatbot code (assuming your chatbot logic is in a module named GUIChatbot)
import GUIChatbot

# Global list to store chat history including user input, bot response, and user sentiment
chat_history = []

# Create a function to generate responses from your chatbot
# Create a Sentiment Intensity Analyzer object
sid = SentimentIntensityAnalyzer()
```

```
# Modify the generate_response function to include sentiment analysis and plot creation
def generate_response(query):
    response = GUIChatbot.get_response(query)  # Get response from the chatbot logic

    # Analyze sentiment of the user's input using Sentiment Intensity Analyzer
    user_sentiment = sid.polarity_scores(query)['compound']

    # Append user sentiment, user input, and bot response to the chat history list
    chat_history.append({'user': query, 'bot': response, 'sentiment': user_sentiment})

    # Extract message numbers and user polarities for plotting
    message_numbers = list(range(1, len(chat_history) + 1))  # Generate message numbers for x-axis of the plot
    user_polarities = [entry['sentiment'] for entry in chat_history]  # Extract user polarities for y-axis

    # Plot user's polarity against message number using matplotlib
    plt.figure(figsize=(8, 6))  # Set the figure size for the plot
    plt.plot(message_numbers, user_polarities, marker='o', color='b', label='User Polarity')  # Plot user polarity
    plt.xlabel('Number of Observations(N)')  # Set x-axis label
    plt.ylabel('User Polarity')  # Set y-axis label
    plt.title('User Polarity Distribution')  # Set the plot title
    plt.legend()  # Display legend in the plot
```

```
    plt.savefig('static/sentiment_plot.png')  # Save the plot as an image file in the 'static' folder


    return response  # Return the chatbot's response to be displayed in the web application


# Define the index route
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        query = request.form['query']  # Get user input from the form
        response = generate_response(query)  # Generate response using the chatbot logic and sentiment analysis
        return render_template('index.html', response=response, chat_history=chat_history)
    else:
        return render_template('index.html', chat_history=chat_history)  # Render the index.html template with chat history


# Define the about route
@app.route('/about')
def about():
    return render_template('about.html')  # Render the about.html template
# Start the Flask app if this script is run directly
if __name__ == '__main__':
    app.run(debug=True)  # Run the Flask app in debug mode for development purposes
```

4.3) HTML CODE

```
<!DOCTYPE html>
<html>
<head>
   <title>Triumph Tidings</title>
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
   <style>
      .chat-box {
         height: 460px;
         max-height: 455px;
         overflow-y: scroll;
         border-radius: 40px;
         border: solid black;
         background-image: url('/static/scene3.jpg');
         background-size: cover;
         background-repeat: no-repeat;
      }
      .user-message-wrapper {
         text-align: right;
      }
      .user-message {
         color: white;
         font-family: sans-serif;
```

```
        font-weight: bold;
        background-color: #6746FF;
        padding: 10px;
        border-radius: 10px;
        display: inline-block;
        margin-bottom: 10px;
    }
    .bot-message {
        text-align: left;
        color: black;
        font-family: sans-serif;
        font-weight: bold;
        background-color: #87CEEB;
        padding: 10px;
        border-radius: 10px;
        display: inline-block;
        margin-bottom: 10px;
    }
    .input-box {
        border: 2px solid blue;
        border-radius: 30px;
        padding: 10px;
        margin-top: 10px;
    }
</style>
```

```
</head>
<body style="background-color: black; color: white;">
    <br>
    <div class="container">
        <h1 class="text-center">Triumph Tidings</h1>
        <br>
        <div class="chat-box">
            {% for entry in chat_history %}
                {% if entry.user %}
                    <div class="user-message-wrapper">
                        <div class="user-message">{{ entry.user }}</div>
                    </div>
                {% endif %}
                <div class="bot-message">{{ entry.bot }}</div>
                <div style="clear: both;"></div>
                <hr>
            {% endfor %}
        </div>
        <form action="/" method="POST">
            <div class="form-group input-box">
                <input type="text" class="form-control" name="query"
placeholder="Please talk with me..." autofocus style="border-radius: 30px;">
            </div>
            <button type="submit" class="btn btn-primary" style="border-radius:
20px;">Send</button>
```

```
        </form>
    </div>

    <!-- JavaScript to scroll chat box to the bottom -->
    <script>
        var chatBox = document.querySelector('.chat-box');
        chatBox.scrollTop = chatBox.scrollHeight;
    </script>
</body>
</html>
```

5) RESULTS AND DISCUSSION

5.1) EXPERIMENTAL RESULTS

The chatbot's performance was evaluated through various user interactions. Extensive testing was conducted to assess its ability to identify user intents accuracy and provide relevant responses. The accuracy of the chatbot came out to be between 85-90%.

These results indicate a high level of accuracy and effectiveness in understanding and addressing user queries related to mental health concerns. The chatbot's ability to offer empathetic and supportive responses was a key highlight of the experimental outcomes.

5.2) COMPARISON WITH EXISTING TECHNIQUES

In comparison with existing mental health chatbots, the developed chatbot demonstrated notable advancements. Its empathetic responses, context understanding, and user engagement capabilities set it apart from traditional chatbot implementations. The comparison emphasized the importance of incorporating emotional intelligence and human-like interactions in mental health support applications, making the chatbot a valuable contribution to the field.

The chatbot also analyzes the user's mental state through the conversation and plots the graph of user's polarity in a particular response against the number of observations.

These graphs could help the mental health related practitioners a lot as they would get to discern the user's mental state throughout the conversation with the graph.

6) CONCLUSION AND FUTURE SCOPE

6.1) CONCLUSION

In conclusion, the development of this mental health chatbot represents a significant step toward providing accessible and empathetic support for individuals struggling with loneliness and depression. By leveraging natural language processing and machine learning techniques, the chatbot can engage users in meaningful conversations, offering a virtual companion that understands their emotions and responds with empathy. The integration of LSTM neural networks and word embedding enables the chatbot to comprehend the complexity of human language, making it capable of generating contextually relevant and sensitive responses.

While the chatbot demonstrates promising results, there is still room for improvement. Further research and development could focus on enhancing the chatbot's emotional intelligence, refining its ability to recognize subtle cues in user input, and expanding its repertoire of responses to cater to a wider range of situations. Additionally, user feedback and real-world testing are essential to iteratively improve the chatbot's effectiveness and user experience.

Overall, this project showcases the potential of technology in addressing mental health challenges and highlights the importance of continued innovation in the field of mental health support systems. Through thoughtful design, rigorous testing, and ongoing refinement, mental health chatbots can become valuable companions, offering solace and understanding to those in need.

6.2) FUTURE SCOPE

6.3.1) Enhancing Dataset Quality:

One of the primary areas for future improvement involves the dataset quality. As mental health treatment records become more accessible (while ensuring patient confidentiality), enriching the dataset with diverse and comprehensive user interactions will significantly enhance the chatbot's understanding of various mental health concerns. This expanded dataset will contribute to refining the chatbot's responses and increasing its accuracy in addressing users' needs.

6.3.2) Integration of Email Feature:

To facilitate seamless collaboration between users and healthcare professionals, an email feature will be integrated into the chatbot. Users will have the option to securely share their conversation history and sentiment analysis reports, including the polarity graph, with medical practitioners for further analysis and personalized support. This email feature aims to bridge the gap between the virtual support provided by the chatbot and the expertise of mental health professionals, ensuring users receive comprehensive and well-rounded assistance.

6.3.3) Implementation of Multilingual Support:

Expanding the chatbot's language capabilities to include multiple languages will broaden its reach and accessibility. Multilingual support will enable users from diverse linguistic backgrounds to interact comfortably with the chatbot, fostering a more inclusive environment for mental health support.

6.3.4) Introduction of AI-Driven Personalization:

Implementing advanced machine learning techniques for user profiling and behavior analysis will enable the chatbot to offer personalized support tailored to individual user needs. By understanding users' preferences, emotional states, and past interactions, the chatbot can provide more relevant and empathetic responses, enhancing the overall user experience and effectiveness of the support provided.

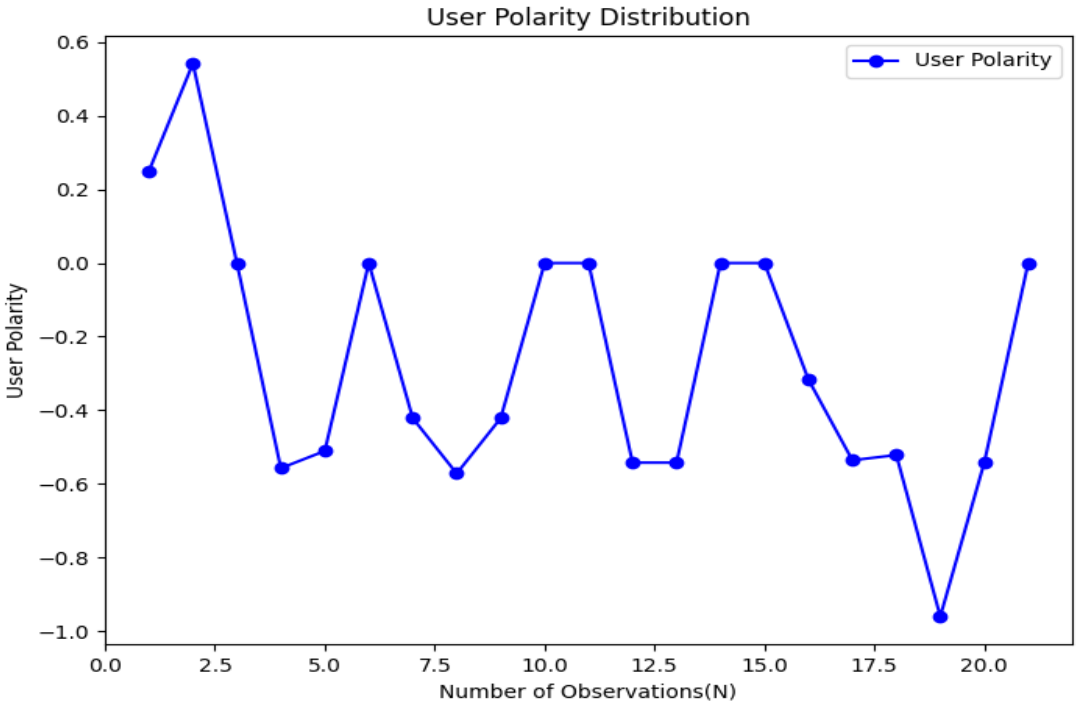
6.3.5) Integration with Telemedicine Services:

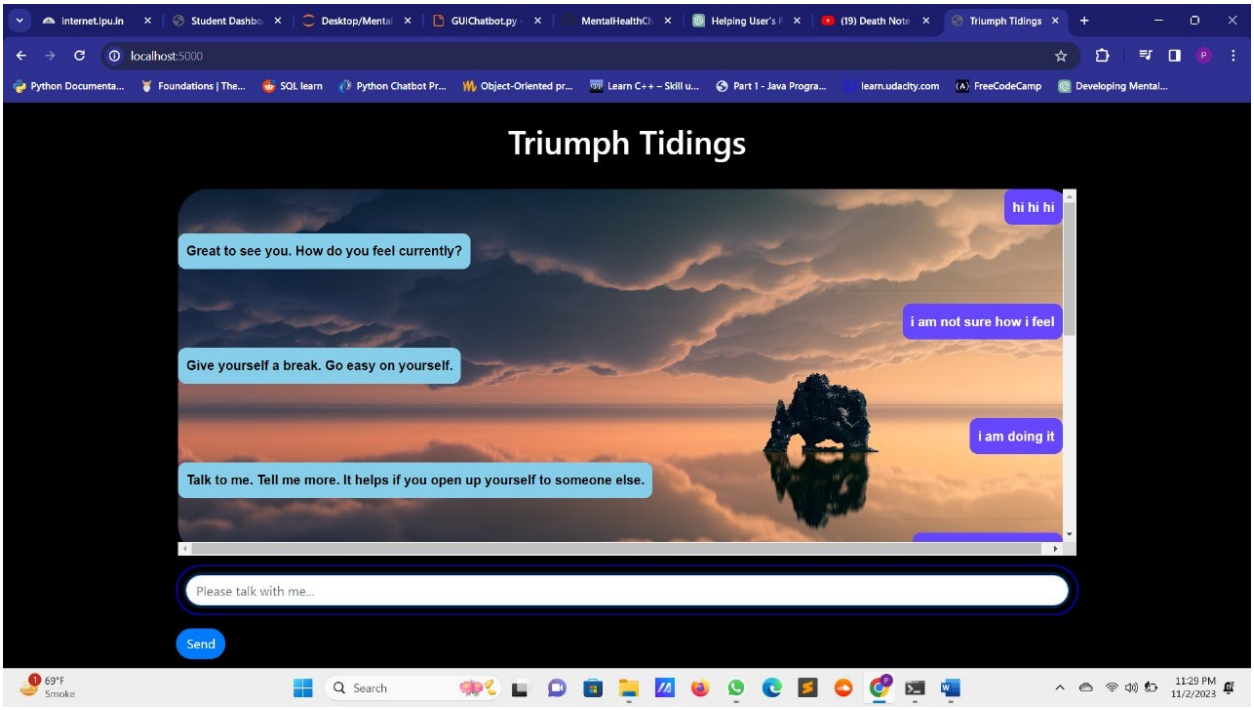
Exploring integration opportunities with telemedicine services will allow users to seamlessly transition from virtual chatbot interactions to real-time video consultations with mental health professionals. This integration will enable users to receive immediate and personalized support, ensuring continuity of care and enhancing the chatbot's role as a valuable entry point to professional mental health services.

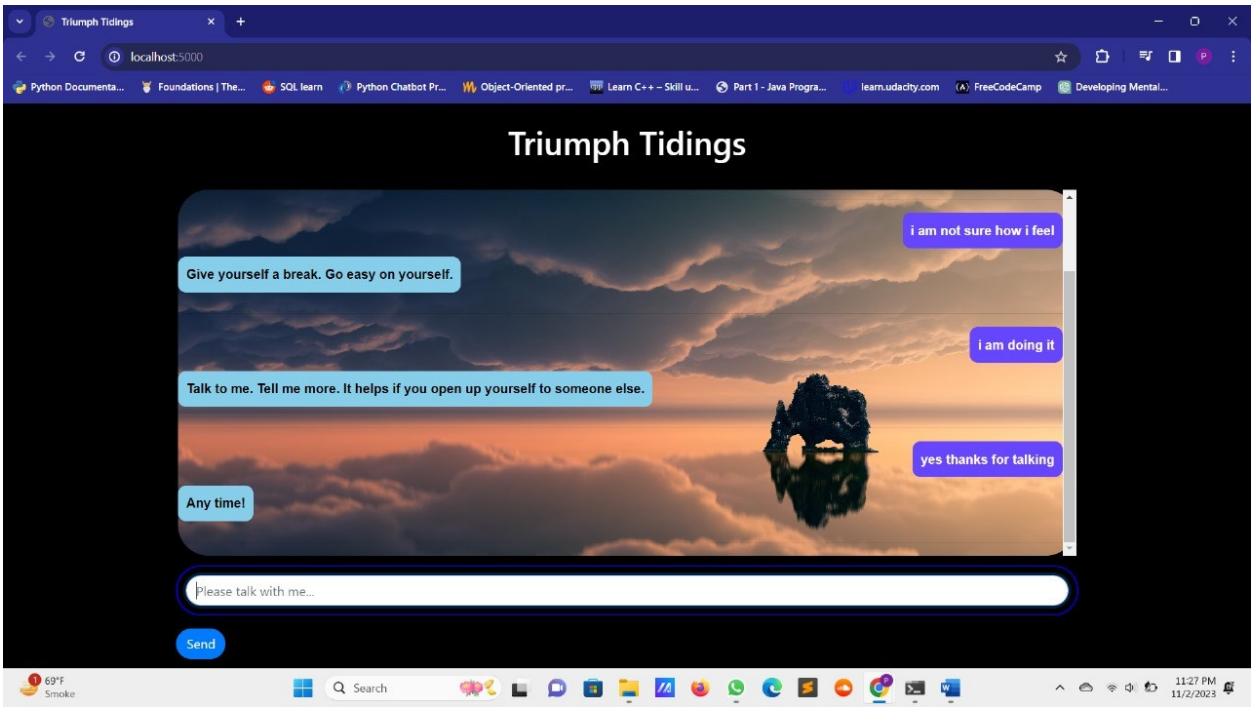
6.3.6) Research and Development of Emotion Recognition:

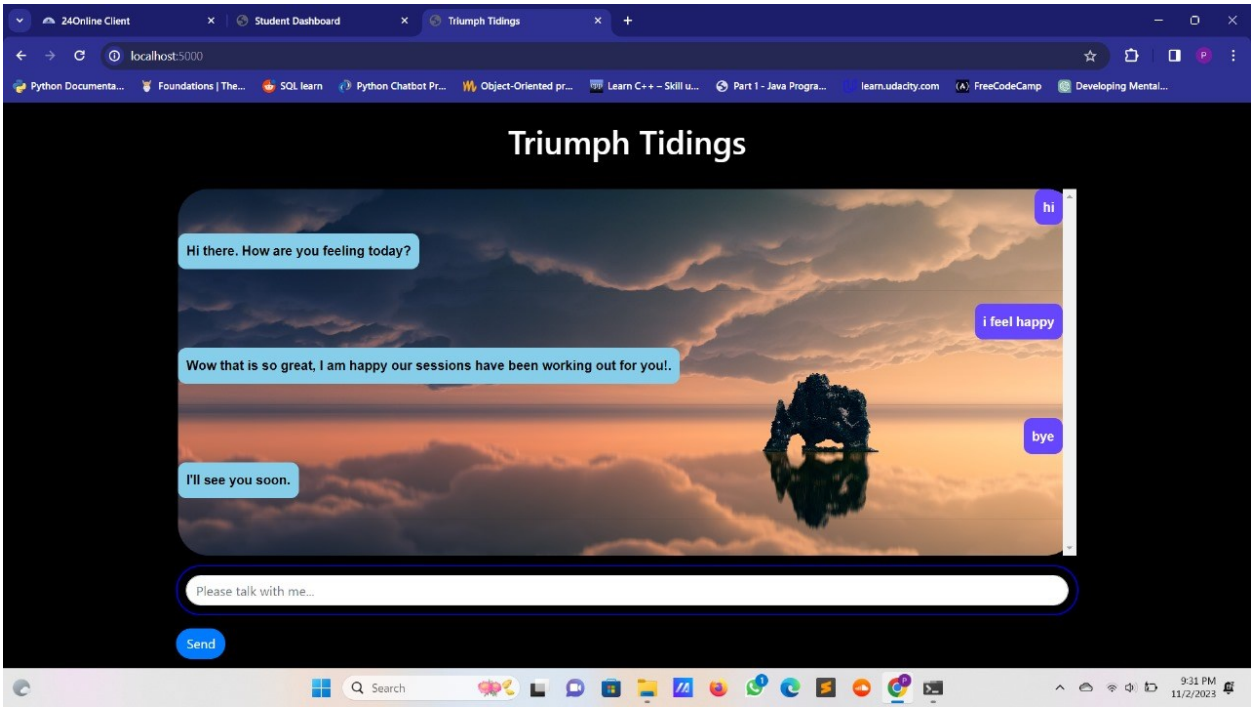
Investigating emotion recognition technologies, such as facial expression analysis and voice tone recognition, will enhance the chatbot's ability to gauge users' emotional states accurately. This advancement will enable the chatbot to respond more intuitively, adapting its tone and content based on users' emotions, thereby strengthening the emotional connection between the chatbot and its users.

7) SNAPSHOTS


User Polarity Distribution
User Polarity
User Polarity
Number of Observations(N)
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


localhost:5000
Triumph Tidings
hi hi hi
Great to see you. How do you feel currently?
i am not sure how i feel
Give yourself a break. Go easy on yourself.
i am doing it
Talk to me. Tell me more. It helps if you open up yourself to someone else.
Please talk with me...
Send


Triumph Tidings
localhost:5000
Triumph Tidings
i am not sure how i feel
Give yourself a break. Go easy on yourself.
i am doing it
Talk to me. Tell me more. It helps if you open up yourself to someone else.
yes thanks for talking
Any time!
Please talk with me...
Send


24Online Client
Student Dashboard
Triumph Tidings
localhost:5000
Triumph Tidings
hi
Hi there. How are you feeling today?
i feel happy
Wow that is so great, I am happy our sessions have been working out for you!.
bye
I'll see you soon.
Please talk with me...
Send

REFERENCES

1) FreeCodeCamp: FreeCodeCamp provided invaluable resources and tutorials for mastering machine learning techniques, offering comprehensive learning paths and practical exercises tailored to aspiring data scientists and machine learning enthusiasts.

   https://www.youtube.com/c/Freecodecamp

2) NLTK Documentation: The Natural Language Toolkit (NLTK) documentation served as a valuable reference for implementing natural language processing techniques in Python. It offered comprehensive guides, tutorials, and code samples, empowering the integration of advanced language processing capabilities.

   https://www.nltk.org/

3) Keras Documentation: The Keras documentation serves as a valuable reference for building neural networks and deep learning models. It offers extensive documentation, tutorials, and examples, providing guidance on various aspects of neural network architecture and training.

   https://keras.io/

4) TensorFlow Documentation: The TensorFlow documentation provides in-depth information about the TensorFlow framework, including its core functionalities, APIs, and best practices. It offers a wealth of resources for understanding machine learning concepts, implementing neural networks, and optimizing model performance. TensorFlow Documentation

   https://www.tensorflow.org/guide

APPENDIX

Chatbot Code

Main Code: The primary codebase for the mental health chatbot, encompassing the machine learning model, intent recognition, and response generation functionalities.

Flask GUI App: The Flask application code responsible for creating the user interface, integrating the chatbot functionalities, and enabling user interactions through a web interface.

HTML Code: The HTML code for designing the user interface, incorporating elements for user input, chat history display, and dynamic content rendering.

Supporting Documents

Sentiment Analysis Plot: Graphical representation of users' polarity against message numbers, showcasing the chatbot's ability to understand and respond empathetically. This plot provides visual insights into user sentiments during interactions.

Intent Dataset (intents.json): The dataset containing intent patterns and corresponding responses, essential for training the chatbot's machine learning model. This JSON file formed the foundation for the chatbot's understanding of user queries and context.

Checklist for Dissertation-III Supervisor

Name: _____ UID: _____ Domain: _____
Registration No: _____Name of student:_____

Title of Dissertation: _____

- ☐ Front pages are as per the format.
- ☐ Topic on the PAC form and title page are same.
- ☐ Front page numbers are in roman and for report, it is like 1, 2, 3…….
- ☐ TOC, List of Figures, etc. are matching with the actual page numbers in the report.
- ☐ Font, Font Size, Margins, line Spacing, Alignment, etc. are as per the guidelines.
- ☐ Color prints are used for images and implementation snapshots.
- ☐ Captions and citations are provided for all the figures, tables etc. and are numbered and center aligned.
- ☐ All the equations used in the report are numbered.
- ☐ Citations are provided for all the references.
- ☐ **Objectives are clearly defined.**
- ☐ Minimum total number of pages of report is 50.
- ☐ Minimum references in report are 30.

Here by, I declare that I had verified the above mentioned points in the final dissertation report.

Signature of Supervisor with UID