

THERAPIST CHATBOT USING EMBEDDING AND LONG-SHORT-TERM-MEMORY IN NLP

Submitted in partial fulfillment of the

requirements for the award of

Bachelor of Engineering Degree in Electronics and Communication Engineering

by

GANTA ANJALI RAGHAVA NAGA SATYA SURYA (39130137)

GUTTALA BINDU POOJITHA (39130159)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600 119

APRIL-2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Final Year Project Report is the bonafide work of **Ganta Anjali Raghava Naga Satya Surya (39130137)** and **Guttala Bindu Poojitha (39130159)** who have done the project work as an individual who carried out the project entitled **"Therapist Chatbot using Embedding and Long-Short-Term-Memory in NLP"** under my supervision from Nov 2022 to April 2023.



Internal Guide

Dr. G. JEGAN, M.E., Ph.D.



Head of the Department
Dr. T. RAVI, M.E., Ph.D.

Submitted for Viva-voce Examination held on 20/04/2023.



Internal Examiner



External Examiner

DECLARATION

We Ganta Anjali Raghava Naga Satya Surya (39130137) and Guttala Bindu Poojitha (39130159) hereby declare that the Project Report on “**Therapist Chatbot using Embedding and Long-Short-Term-Memory in NLP**” done by us under the guidance of **Dr. G. Jegan, M.E., Ph.D.** at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Electronics and Communication Engineering**.

DATE: 20/04/2023.

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATES

1. G. Bindu Poojitha
2. Anjali Raghava

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA M.E., Ph. D, Dean**, School of Electronics, **Dr. T. RAVI M.E., Ph.D.**, Head of the Department of Electronics and Communication Engineering for providing us with the necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr.G.JEGAN M.E., Ph.D**, for his valuable guidance, suggestions and constant encouragement that paved the way for the successful completion of my project work.

We wish to express our thanks to all the Teaching and Non-teaching staff members of the **Department of Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Now a day's stress management must be diagnosed and managed before it becomes severe. People seek help from healthcare facilities and psychologists to deal with stress. The project recognizes the need for stress prediction and management in young people and employees working in demanding environments with missed deadlines. Chatbots, which provide an interactive way to get in contact with users in behavioral health therapies powered by artificial intelligence, could be a scalable solution to the identified problems. Neural networks and machine learning techniques such as decision trees are used to train the data and guess the most accurate stress level that could be related to the patient's details, and the steps to be taken are then displayed.

The existing system is not done in real-time and it wasn't implemented in real-time user inputs also the accuracy and dataset usage are really poor in the existing system. The proposed system has higher efficiency in terms of accuracy, precision, and recall value. Also, the proposed system can be used for real-time complex use cases.

Keywords— Bot, Chatbot, Emotion, Mental Health, NLP (Natural Language Processing), Stress, Therapy.

TABLE OF CONTENT

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	V
	LIST OF FIGURES	IX
1	INTRODUCTION	
	1.1 Stress Management	1
	1.2 Artificial Intelligence and NLP	3
	1.3 Exploratory Data Analysis Method (EDA)	8
	1.4 Term frequency-inverse document frequency (TF-IDF)	8
2	LITERATURE SURVEY	
	2.1 Literature Review	13
	2.2 Summary of Survey	15
	2.3 Objective and Problem Statement	16
3	EXISTING SYSTEM	
	3.1 Architecture Diagram of Chatbot	19
	3.2 Algorithms Used in Chatbot	19
	3.3 Drawbacks in Chatbot	20
4	PROPOSED SYSTEM	
	4.1 System Architecture of Therapist Chatbot	21
	4.2 Algorithms Used in Therapist Chatbot	22
	4.3 Analysis and Methodologies of Therapist Chatbot	25
	4.4 Implementation of Therapist Chatbot	33
	4.5 Platform Specification	34
	4.6 Financial Report on Estimated Costing	36
	4.7 Transitions/Software to Operations Plan	37
5	IMPLEMENTATION DETAILS	
	5.1 Development/Deployment Setup	38
	5.2 Algorithms	39
	5.3 Testings	39

CHAPTER No.	TITLE	PAGE No.
6	RESULTS AND DISCUSSIONS	42
7	CONCLUSION	
	7.1 Conclusion	45
	7.2 Future Work	45
	7.3 Research Issues	46
	7.4 Implementation Issues	48
	REFERENCES	52
	APPENDIX	
	A. SOURCE CODE	54
	B. RESEARCH PAPER	66

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1.1	Stress Management	1
1.2	Symptoms in Stress	2
3.1	Architecture Diagram of Existing System	19
4.1	Architecture Diagram of Therapist Chatbot	21
4.2.	Data Flow Diagram	24
4.2.	Collaboration Diagram	24
4.2.	ER Diagram	25
4.2.	Sequence Diagram	25
6.1	Registration or Login page of Health Care Chatbot	42
6.2	UI after successful login, which is about to start after clicking on the start button.	43
6.3	This page appears after clicking on the start button, which shows some random questions.	43
6.4	This is the doctor's page, where we can go through their profile and book an appointment according to the slots.	44
6.5	Clinic appointment, patient details, and payment details page	44

CHAPTER 1

INTRODUCTION

1.1 STRESS MANAGEMENT

Stress management is a wide spectrum of techniques and psychotherapies aimed at controlling a person's stress level, especially chronic stress, usually to improve everyday functioning. Stress produces numerous physical and mental symptoms which vary according to each individual's situational factors. These can include a decline in physical health, such as headaches, chest pain, fatigue, and sleep problems as well. Stress management is one of the keys to a happy and successful life in modern society. Life often delivers numerous demands that can be difficult to handle, but stress management provides several ways to manage anxiety and maintain overall well-being.



Fig 1.1 Stress Management

➤ Types of Stress

There are three types of Stress

➤ Acute stress

Acute stress is the most common form of stress among humans worldwide.

➤ Episodic acute stress

This type of stress occurs when an individual gets regular bouts of acute stress.

➤ Chronic stress

Chronic stress is unlike acute stress. It has a wearing effect on people that can become a very serious health risk if it continues over a long period. Chronic stress affects both the mind and body. It produces both physical and psychological symptoms that can take a toll on a person's ability to function normally in their daily life.

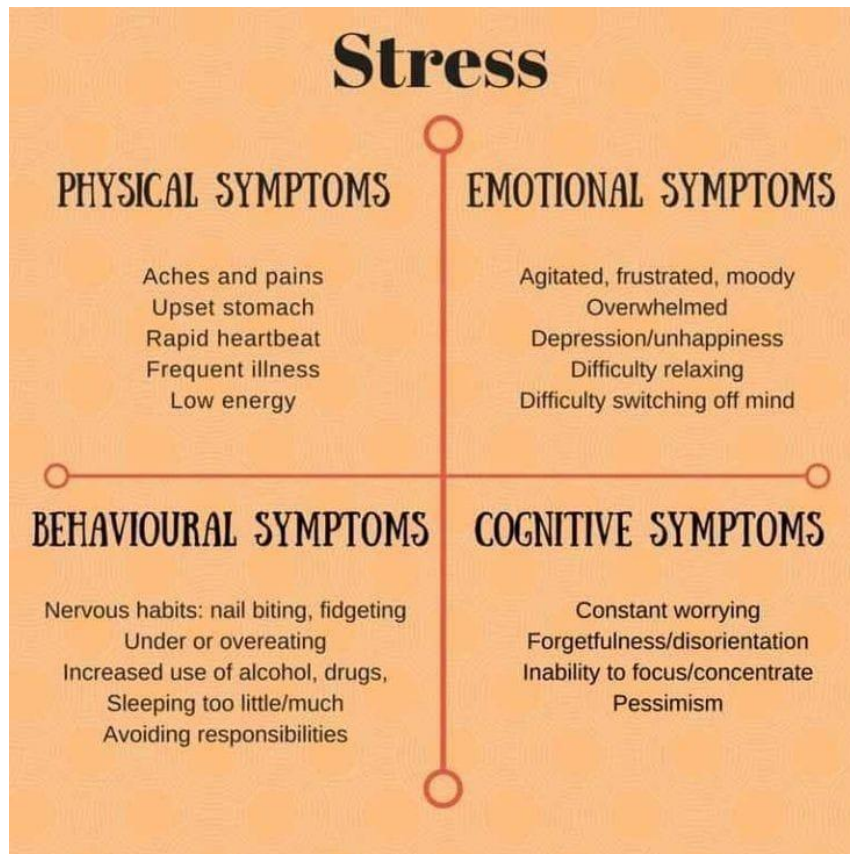


Fig 1.2 Symptoms in stress

➤ Effects of Stress

There are some physical changes that a person experiences during stress. These physical changes include- headaches, irritability, mood swings, sleep disorders, etc. Stress that's left unchecked can contribute to many health problems, such as high blood pressure, heart disease, obesity and diabetes. Prolonged stress can adversely affect health, leading to conditions such as

1. Fatigue
2. Depression

3. Atherosclerosis
4. Fertility problems
5. Breathing problems
6. Weight gain or loss
7. Abnormal heartbeat
8. High blood pressure
9. Heart-related disorders

Managing Stress

1. Always have positive thoughts
2. Have a healthy and balanced diet
3. Exercise regularly
4. Ensure adequate sleep
5. Practice meditation and breathing techniques

1.2 APPLICATIONS OF ARTIFICIAL INTELLIGENCE (AI)

Artificial Intelligence (AI) in the assessment and treatment of mental health has gained traction in recent years, thanks to the rapid development of chatbots.

Experts feel that the promise has yet to be realized because the majority of chatbots discovered in the literature make a conscious decision based on a predetermined response.

User privacy and scientific evaluations of early depression diagnosis are not assured in present systems due to customized chatbot platforms.

The work proposes and creates an algorithm based on modern bot platforms for early depression diagnosis to address mental health symptoms, motivated by existing system shortcomings. Artificial intelligence is the most important methodology for creating intelligent bots.

Artificial intelligence (AI) is a branch of computer science that focuses on the development of intelligent machines that function and react in the same way that humans do. Artificial Intelligence (AI) has a long history in computers, dating back to the 1940s. Scientists have been utilizing AI to test their research on how the human brain works since the dawn of time. Artificial Intelligence is the process of teaching computers

to learn, analyze, think, and solve problems. The implementation of a counseling chatbot based on artificial intelligence is shown in this study. It offers users counseling to help them solve their difficulties. The chatbot offers advice to the user based on their current mood. To engage in a discourse that is as natural as possible with people. Chatbots should be able to analyze all inputs given by people and judge their moods to give output.

1.2.1 Natural Language Processing (Project Domain)

Natural Language Processing, or NLP for short, is broadly defined as the automatic manipulation of natural language, like speech and text, by software.

Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it's ingesting. Some of these tasks include the following:

- **Speech recognition**, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions. What makes speech recognition especially challenging is the way people talk—quickly, slurring words together, with varying emphasis and intonation, in different accents, and often using incorrect grammar.
- **Part of speech tagging**, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies 'make' as a verb in 'I can make a paper plane,' and as a noun in 'What make of car do you own?'
- **Word sense disambiguation** is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in 'make the grade' (achieve) vs. 'make a bet' (place).
- **Named entity recognition**, or NEM, identifies words or phrases as useful entities. NEM identifies 'Kentucky' as a location or 'Fred' as a man's name.
- **Co-reference resolution** is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., 'she' = 'Mary'), but it can also involve

identifying a metaphor or an idiom in the text (e.g., an instance in which 'bear' isn't an animal but a large hairy person).

- **Sentiment analysis** attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text.
- **Natural language generation** is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

1.2.2 Segmentation and Word Tokenization Sentence

Sentence Segmentation breaks the piece of text into various sentences and Word Tokenization means breaking the sentence into individual words known as tokens. We can tokenize them whenever we encounter a space, we can train a model in that way. Even punctuations are considered individual tokens as they have some meaning. In linguistic analysis of a natural language text, it is necessary to clearly define what constitutes a word and a sentence. Defining these units presents different challenges depending on the language being processed, but neither task is trivial, especially when considering the variety of human languages and writing systems. Natural languages contain inherent ambiguities, and much of the challenge of Natural Language Processing (NLP) involves resolving these ambiguities. In this chapter we will discuss the challenges posed by text segmentation, the task of dividing a text into linguistically-meaningful units at the lowest level characters representing the individual graphemes in a language's written system, words consisting of one or more characters, and sentences consisting of one or more words.

Text segmentation is a frequently overlooked yet essential part of any NLP system, since the words and sentences identified at this stage are the fundamental units passed to further processing stages such as morphological analyzers, part-of-speech taggers, parsers, and information retrieval systems. Tokenisation is the process of breaking up the sequence of characters in a text by locating the word boundaries, the points where one word ends and another begins. For computational linguistics purposes, the words thus identified are frequently referred to as tokens. In written languages where no word boundaries are explicitly marked in the writing system, tokenisation is also known as word segmentation, and this term is frequently used synonymously with tokenisation. Sentence segmentation is the process of determining the longer processing units

consisting of one or more words. This task involves identifying sentence boundaries between words in different sentences since most of the languages have punctuation marks that occur at sentence boundaries, sentence segmentation is frequently referred to as sentence boundary detection, sentence boundary disambiguation, or sentence boundary recognition. All these terms refer to the same task: determining how a text should be divided into sentences for further processing. In practice, sentence and word segmentation cannot be performed successfully independent from one another. For example, an essential subtask in both word and sentence segmentation for English is identifying abbreviations, because a period can be used in English to mark an abbreviation as well as to mark the end of a sentence. In the case of a period marking an abbreviation, the period is usually considered a part of the abbreviation token, whereas a period at the end of a sentence is usually considered a token in and of itself. Tokenising abbreviations is complicated further when an abbreviation occurs at the end of a sentence, and the period marks both the abbreviation and the sentence boundary. This chapter will provide an introduction to word and sentence segmentation in a Palmer Tokenisation and Sentence Segmentation variety of languages. We will begin in Section 2 with a discussion of the challenges posed by text segmentation, and emphasize the issues which must be considered before implementing a tokenisation or sentence segmentation algorithm. The section will describe the dependency of text segmentation algorithms on the language being processed and the character set in which the language is encoded. It will also discuss the dependency on the application that uses the output of the segmentation and the dependency on the characteristics of the specific corpus being processed. In Section 3 we will introduce some common techniques currently used for tokenisation. The first part of the section will focus on issues that arise in tokenising languages in which words are separated by whitespace. The second part of the section will discuss tokenisation techniques in languages where no such whitespace word boundaries exist. In Section 4 we will discuss the problem of sentence segmentation and introduce some common techniques currently used to identify sentences boundaries in texts.

1.2.3 Predicting Parts of Speech for each Token

Predicting whether the word is a noun, verb, adjective, adverb, pronoun, etc. This will help to understand what the sentence is talking about. This can be achieved by feeding the tokens (and the words around them) to a pre-trained part-of-speech classification model. This model was fed a lot of English words with various parts of speech tagged to

speech. Again, the models don't understand the 'sense' of the words, it just classifies them based on their previous experience. It's pure statistics.

1.2.4 Lemmatization Feeding the model with the root word

For example – There's a Buffalo grazing in the field. Buffaloes are grazing in the field. Here, both Buffalo and Buffaloes mean the same. But the computer can confuse it with two different terms as it doesn't know anything. So, we have to teach the computer that both terms mean the same. We have to tell a computer that both sentences are talking about the same concept. So, we need to find out the most basic form or root form or lemma of the word and feed it to the model accordingly. Similarly, we can use it for verbs too. 'Play' and 'Playing' should be considered as same.

- **Identifying stop words**

There are various words in the English language that are used very frequently like 'a', 'and', 'the' etc. These words make a lot of noise while doing statistical analysis. We can take these words out. Some NLP pipelines will categorize these words as stop words, they will be filtered out while doing some statistical analysis. They are needed to understand the dependency between various tokens to get the exact sense of the sentence. The list of stop words varies and depends on what kind of output are you expecting.

- **Dependency Parsing**

This means finding out the relationship between the words in the sentence and how they are related to each other. We create a parse tree in dependency parsing, with the root as the main verb in the sentence. If we talk about the first sentence in our example, then 'is' is the main verb and it will be the root of the parse tree. We can construct a parse tree of every sentence with one root word (main verb) associated with it. We can also identify the kind of relationship that exists between the two words. In our example, 'San Pedro' is the subject, and 'island' is the attribute. Thus, the relationship between 'San Pedro' and 'is', and 'island' and 'is' can be established. Just like we trained a Machine Learning model to identify various parts of speech, we can train a model to identify the dependency between words by feeding many words. It's a complex task though. In 2016, Google released a new dependency parser Parsey McParseface which used a deep learning approach.

- **Finding Noun Phrases**

We can group the words that represent the same idea. For example – It is the second- largest town in the Belize District and the largest in the Belize Rural South constituency. Here, tokens ‘second’, ‘largest’, and ‘town’ can be grouped as they together represent the same thing ‘Belize’. We can use the output of dependency parsing to combine such words. Whether to do this step or not completely depends on the end goal, but it’s always quick to do this if we don’t want much information about which words are adjectives, rather than focus on other important details.

- **Named Entity Recognition (NER)**

San Pedro is a town on the southern part of the island of Ambergris Caye in the Belize District of the nation of Belize, in Central America. Here, the NER maps the words with real-world places. The places that exist in the physical world. We can automatically extract the real-world places present in the document using NLP. If the above sentence is the input, NER will map it this way: San Pedro - Geographic Entity Ambergris Caye - Geographic Entity Belize - Geographic Entity Central America - Geographic Entity NER systems look for how a word is placed in a sentence and make use of other statistical models to identify what kind of word it is. For example – ‘Washington’ can be a geographical location as well as the last name of any person. A good NER system can identify this. Kinds of objects that a typical NER system can tag: People’s names. Company names. Geographical locations Product names. Date and time. Amount of money.

- **Coreference Resolution**

San Pedro is a town on the southern part of the island of Ambergris Caye in the Belize District of the nation of Belize, in Central America. According to 2015 mid-year estimates, the town has a population of about 16, 444. It is the second-largest town in the Belize District and the largest in the Belize Rural South constituency. Here, we know that ‘it’ in sentence 6 stands for San Pedro, but for a computer, it isn’t possible to understand that both the tokens are the same because it treats both sentences as two different things while it’s processing them. Pronouns are used with a high frequency in English literature and it becomes difficult for a computer to understand that both things are the same.

1.3 EXPLORATORY DATA ANALYSIS METHOD (EDA)

Exploratory Data Analysis (EDA) is an approach to analyzing the data using visual techniques. It is used to discover trends, and patterns, or to check assumptions with the help of statistical summaries and graphical representations. EDA is the process of investigating the dataset to discover patterns, and anomalies (outliers), and form hypotheses based on our understanding of the dataset. EDA involves generating summary statistics for numerical data in the dataset and creating various graphical representations to understand the data better. EDA assists Data science professionals in various ways: -

1. Getting a better understanding of data
2. Identifying various data patterns
3. Getting a better understanding of the problem statement

EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate. The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, and find interesting relations among the variables.

1.4 TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF)

It is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches for information retrieval, text mining, and user modeling. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. tf-idf is one of the most popular term-weighting schemes today. TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a

set of documents. It has many uses, most importantly in automated text analysis, and is very useful for scoring words in machine learning algorithms for Natural Language Processing (NLP).

TF-IDF was invented for document search and information retrieval. It works by increasing proportionally to the number of times a word appears in a document but is offset by the number of documents that contain the word. So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times since they don't mean much to that document in particular. However, if the word Bug appears many times in a document, while not appearing many times in others, it probably means that it's very relevant. For example, if what we're doing is trying to find out which topics some NPS responses belong to, the word Bug would probably end up being tied to the topic Reliability, since most responses containing that word would be about that topic.

1.4.1 How is TF-IDF calculated?

TF-IDF for a word in a document is calculated by multiplying two different metrics: The term frequency of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by the length of a document, or by the raw frequency of the most frequent word in a document. The inverse document frequency of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm. So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, will approach 1. Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

1.4.2 Why is TF-IDF used in Machine Learning?

Machine learning with natural language is faced with one major hurdle – its algorithms usually deal with numbers, and natural language is, well, text. So, we need to transform that text into numbers, otherwise known as text vectorization. It's a

fundamental step in the process of machine learning for analyzing data, and different vectorization algorithms will drastically affect results, so you need to choose one that will deliver the results you're hoping for. Once you've transformed words into numbers, in a way that machine learning algorithms can understand, the TF-IDF score can be fed to algorithms such as Naive Bayes and Support Vector Machines, greatly improving the results of more basic methods like word counts. Why does this work? Simply put, a word vector represents a document as a list of numbers, with one for each possible word of the corpus. Vectorizing a document is taking the text and creating one of these vectors, and the numbers of the vectors somehow represent the content of the text. TF-IDF enables us to give us a way to associate each word in a document with a number that represents how relevant each word is in that document. Then, documents with similar, relevant words will have similar vectors, which is what we are looking for in a machine learning algorithm.

TF-IDF (Term Frequency-Inverse Document Frequency) is a commonly used technique in machine learning for text data preprocessing. It is used to convert a collection of text documents into numerical feature vectors, which can then be used to train machine learning models for various natural language processing tasks such as document classification, text clustering, and information retrieval.

The TF-IDF technique assigns a weight to each word in a document based on its frequency in the document (TF) and its frequency across all documents in the corpus (IDF). The intuition behind this is that words that are frequent in a particular document but rare in other documents are likely to be more important for that document's content, and should be assigned a higher weight.

1) TF-IDF is used in machine learning for several reasons:

2) Feature extraction: It helps in extracting relevant features from text data and converting them into numerical vectors that can be used as input to machine learning models.

3) Dimensionality reduction: By assigning a weight to each word, TF-IDF reduces the dimensionality of the feature space and avoids the curse of dimensionality.

4) Text classification: It helps in classifying documents into different categories by

identifying the important words that distinguish them.

5) Information retrieval: It helps in retrieving relevant documents from a large corpus based on a user's query by ranking them according to their TF-IDF scores.

Overall, TF-IDF is a powerful and widely used technique in machine learning for processing text data and has proven to be effective in a variety of natural language processing tasks.

1.4.3 Applications of TF-IDF

Determining how relevant a word is to a document, or TD-IDF, is useful in many ways, for example, Information retrieval. TF-IDF was invented for document search and can be used to deliver results that are most relevant to what you're searching for. Imagine you have a search engine and somebody looks for LeBron. The results will be displayed in order of relevance. That's to say the most relevant sports articles will be ranked higher because TF-IDF gives the word LeBron a higher score. Every search engine you have ever encountered likely uses TF-IDF scores in its algorithm. Keyword Extraction: TF-IDF is also useful for extracting keywords from the text. How? The highest-scoring words of a document are the most relevant to that document, and therefore they can be considered keywords for that document. Pretty straightforward. TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure used to evaluate the importance of a term within a document or a corpus of documents. Here are some applications of TF-IDF:

Information Retrieval: One of the most common applications of TF-IDF is in information retrieval systems such as search engines. TF-IDF helps in ranking the search results by giving higher weights to the documents that contain the query terms more frequently.

Text Classification: TF-IDF is also used in text classification tasks such as spam filtering, sentiment analysis, and topic modeling. It helps in identifying the most important words or phrases in a document that can be used to categorize it into a particular class.

Keyword Extraction: TF-IDF can be used to extract keywords from a document or a set of documents. The words that have the highest TF-IDF scores are considered to be the most important and relevant keywords.

Content-Based Recommendation: TF-IDF can be used to recommend similar items to users based on their past behavior or preferences. The system identifies the items that have similar keywords and calculates the TF-IDF scores to recommend the most relevant ones.

Information Extraction: TF-IDF can be used to extract information from unstructured data such as social media posts or news articles. The system identifies the most important words or phrases and extracts the relevant information based on their TF-IDF scores.

Overall, TF-IDF is a powerful tool for analyzing and extracting information from text data, and it has a wide range of applications in various fields such as natural language processing, information retrieval, and machine learning.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW:

REFERENCE TITLE: Md Zia Uddin et al., 2022, “Deep learning for prediction of depressive symptoms in a large textual dataset,” Springer Open Access, Neural Computing, and Applications Volume 34, pages 721–744.

INFERENCE:

This study presents an effective method for identifying texts describing self-perceived depressive symptoms using a Recurrent Neural Network (RNN) based on Long Short-Term Memory (LSTM). The method is used on a huge dataset derived from a public online information channel for young people in Norway. The dataset is made up of text-based queries submitted by kids via this information channel. The features are then offered as a result of a one-time process on robust features taken from a reflection of likely depression symptoms pre-defined by medical and psychological experts.

REFERENCE TITLE: Romit Vinod Kankaria, Aman Agrawal, 2021, “RAAH.ai: An interactive chatbot for stress relief using Deep Learning and Natural Language Processing”, 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT).

INFERENCE:

The system employs chatbot NLP approaches to determine the emotion/problem that the user is attempting to express. It responds to the user in the voice of a loved one using CNN and RNN. We created a Tacotron-based architecture using the LibriSpeech and VCTK datasets, which led to a similarity percentage of 91 percent and a response time of 4 seconds, which was better than the other methods investigated throughout the literature survey.

REFERENCE TITLE: Payam Kaywan, Khandakar Ahmed, Yuam Miao, Ayman Ibaida, Bruce Gu, “DEPRA: An Early Depression Detection Analysis Chatbot”, Springer, 2021.

INFERENCE

Because of the increasing development of chatbots, the use of Artificial Intelligence (AI) in the assessment and treatment of mental health has gained traction in recent years.

Recent attempts have yielded promising results, such as the facilitation of recognizing the depression level in patients' profiles, which has increased the goal of finding a solution to assist medical professionals in detecting depression. To address mental health symptoms, we conceived and created DEPRA, which is based on modern bot platforms with early depression identification. DEPRA is based on Dialog flow as a conversation interface and is trained using individualized utterances collected from a specific population.

REFERENCE TITLE: Tushar Sharma, Jitendra Parihar, and Saurabh Singh, "Intelligent Chatbot for Stress Prediction and Management", 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), IEEE, 2021.

INFERENCE:

The Stress Management and Prediction application is an end-user support and consultation project. Here is a standalone application that may be embedded with web pages or phone apps. The application receives numerous details as well as the stress related to those details. The software allows users to share their stress-related problems. The paper employs neural networks and machine learning techniques such as decision trees to train the data and guess the most correct stress level that might be related to the patient's details, and it then displays the appropriate measures.

REFERENCE TITLE: Payam Kaywan, Khandakar Ahmed, Yuam Miao, Ayman Ibaida, Bruce Gu, 2019, "Combating Depression in Students using an Intelligent ChatBot: A Cognitive Behavioral Therapy," 2019 IEEE 16th India Council International Conference (INDICON).

INFERENCE:

The authors developed an intelligent social therapeutic chatbot in this paper that divides the text into emotion labels such as Happy, Joy, Shame, Anger, Disgust, Sadness, Guilt, and Fear. Furthermore, depending on the emotion label, determines the users' mental condition, such as stress or depression, utilizing conversation data from the users. We used three common deep learning classifiers for emotion detection: Convolutional Neural Network (CNN), Recurrent Neural Network (CNN), and Hierarchical Attention Network (HAN) (HAN). In particular, the suggested chatbot methodology is domain-specific, with the chatbot attempting to prevent pessimistic actions and rebuild more constructive

attitudes through user engagement.

2.2 SUMMARY OF SURVEY

To address mental health problems, we imagined and developed DEPRA, a platform based on modern bot platforms that can detect depression early. DEPRA is taught using customized utterances collected from a specific demographic and is built on Dialog flow as a conversation interface.

In this work, the authors create an intelligent social therapeutic chatbot that splits the text into emotion labels like Happy, Joy, Shame, Anger, Disgust, Sadness, Guilt, and Fear. Furthermore, based on the emotion label, it identifies the users' mental state, such as whether they are anxious or depressed, by analyzing their chat data.

To determine the emotion or problem that the user is seeking to express, the system uses chatbot NLP methods. It speaks back to the user in the voice of a friend or family member.

Using a Recurrent Neural Network (RNN) based on Long Short-Term Memory, this study proposes an excellent method for recognizing texts describing self-perceived depressed symptoms (LSTM). The approach is used for a large dataset generated from a publicly accessible online information channel for Norwegian youth.

The paper uses neural networks and machine learning techniques like decision trees to train the data and guess the most accurate stress level that could be associated with the patient's specifics, after which it shows the right measures.

A real-time artificial human discourse that is customized and personalized to a person's wants is required. In addition, there is a security requirement to avoid voice spoofing, which could lead to fraud. As a result, we developed a system that allows a person to interact with others. The system uses natural language processing (NLP) techniques to determine the emotion or problem that the user is attempting to express. It responds to the user with the voice of a loved one, using CNN and RNN. This article is part of a research study that looks into the use of chatbots to provide online emotional support to humans that are personalized to their specific circumstances. We've created a socially interactive system based on a variety of empirical investigations. In this study, we develop a chatbot as part of a stress management framework aimed at assisting healthcare workers in managing their work-related stress. Our framework uses chatbots and robots to have conversations with people to generate a stress score using a Sense of Coherence (SOC) model and then provide support based on the SOC score.

The goal of this project is to create an HR chatbot using artificial intelligence capabilities such as natural language understanding. Allowing people to communicate with the chatbot using natural language input and training the chatbot in proper ways so that it can respond.

When compared to RNN-based models, this model takes less time to train and achieves state-of-the-art sequence transduction performance. We used the transformer model for Bengali general knowledge chatbots in this study, which was based on the Bengali general knowledge Question Answer (QA) dataset.

we figured that we often tend to not give importance to our product carrier during research. This paper was studied to explain the scenario of chatbots in these times and how impactful it is in various parts of the world. The fact that a chatbot should emote makes it important for our application which is listed as a strength in this paper and various surveys available. A few key issues were discussed in this particular research such as Malfunctioning chatbots and unanswered questions, Information security and data protection, Data protection of personal and sensitive data, and Artificial/non-human conversation.

Studied two approaches which included speaker adaptation as well as speaker encoding. The dataset used is a) the LibriSpeech dataset (Panayotov et al., 2015) and b) the VCTK dataset (Veaux et al., 2017). Both the approaches used are evaluated using two techniques namely, i) Speaker Classification and ii) Speaker Verification. There were some issues identified in this research by us such as 1) the Speaker Adaptation technique requires high cloning time and memory requirement 2) the Speaker Encoding technique does not attain the naturalness required in voice modulation 3) 33 minutes of per speaker data are required for both the techniques which make a new user integration difficult 4) A large number of samples with exponentially increasing iterations is required for a good accuracy which is time-consuming.

2.3 OBJECTIVE AND PROBLEM STATEMENT

2.3.1 Aim of the project

The purpose of the research is to create a machine learning-based model for identifying stress in people. To foresee stress management strategies for persons who have been diagnosed with a mental illness. To efficiently foresee and control stress in

individuals, using chatbot software.

2.3.2 Scope of the Project

Mental health chatbots may be effective in assisting patients with anxiety and depression, as well as offering critical support to mental healthcare clinicians, especially when resources are scarce.

The scope of a mental health chatbot project can vary depending on the goals and objectives of the project. Here are some possible areas that a mental health chatbot project can cover:

1) Assessing and screening: A mental health chatbot can be designed to assess and screen users for mental health disorders, such as anxiety and depression, by asking relevant questions and identifying symptoms.

2) Providing information and resources: The chatbot can provide users with information about mental health issues, treatment options, and resources for seeking help

3) Offering support and encouragement: A mental health chatbot can provide emotional support and encouragement to users who are struggling with mental health issues, through positive messaging, motivational quotes, or affirmations.

4) Offering coping strategies and techniques: The chatbot can provide users with coping strategies and techniques to help manage stress, anxiety, and other mental health issues.

5) Referring to professionals: The chatbot can refer users to mental health professionals, such as therapists or psychiatrists, if the user requires additional support or treatment.

6) Personalizing the experience: The chatbot can personalize the user experience by tailoring the conversation to the user's specific needs, preferences, and mental health history.

7) Collecting data and insights: The chatbot can collect data and insights about user interactions, which can be used to improve the chatbot's effectiveness, personalize the experience further, and identify emerging mental health issues or trends.

Overall, the scope of the project will depend on the specific goals and objectives of the mental health chatbot project, as well as the needs and preferences of the target audience.

2.3.3 Problem Statement

It is proposed an intelligent social therapeutic chatbot that detects an individual's stress levels in this project. Based on the emotion label, determines the users' mental state, such as stress or depression, using conversation data. To detect stress, we employed a Decision Tree, one of the most used machine learning classifiers. The suggested chatbot methodology, in particular, is domain-specific, i.e., based on user interaction. Apps and web-based chatbots can support healthcare professionals in analyzing and directing the management of a variety of health issues, especially when human resources are scarce. Mental health chatbots may be effective in assisting patients with anxiety and depression, as well as offering critical support to mental healthcare clinicians, especially when resources are scarce. Mental health issues such as depression, anxiety, and stress are becoming increasingly common in today's society. However, many individuals do not have access to affordable or convenient mental health services. Moreover, there is still a stigma attached to seeking mental health care, which can prevent individuals from seeking help when they need it.

Therefore, there is a need for a mental health chatbot that can provide individuals with a confidential, convenient, and accessible platform to discuss their mental health concerns. The chatbot should be able to provide evidence-based advice and support, as well as direct individuals to appropriate resources or services if needed. Additionally, the chatbot should be able to monitor and track individuals' mental health over time to provide tailored support and identify any changes or concerns.

CHAPTER 3

EXISTING SYSTEM

3.1 ARCHITECTURE DIAGRAM OF MEDICAL CHATBOT

Using machine learning to improve psychological results is a method that has been used in previous projects. Supervised learning with a Convolutional Neural Network has been used to achieve a better prediction of stress levels in students. The initial problem with the resulting table was the sheer amount of dataset taken into use and how it was structured. This led to poor accuracy and inefficient results. It also caused false assumptions and low precision. In the existing systems, user privacy, and scientific evaluations of early depression detection are not guaranteed due to the customized chatbot platforms.

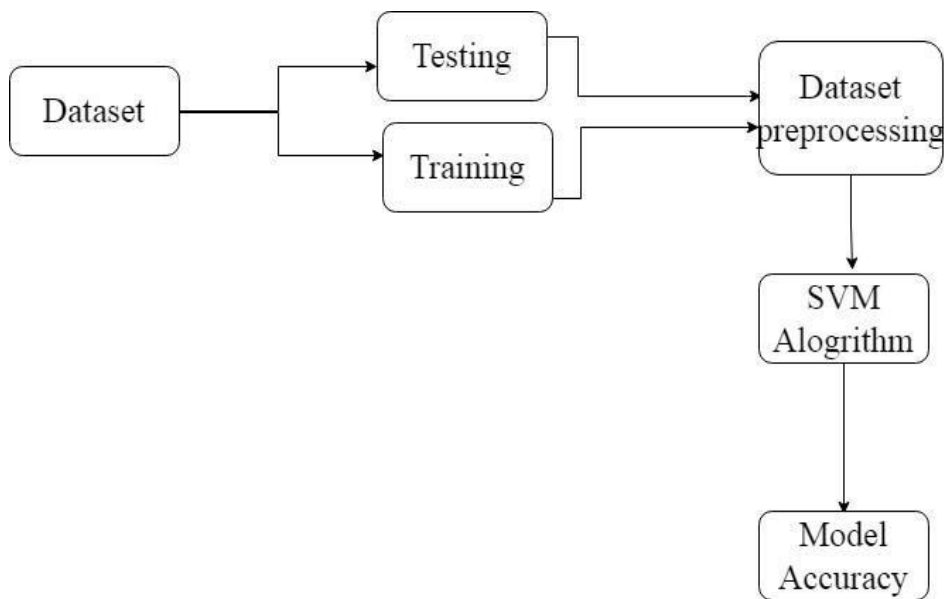


FIG 3.1: Architecture Diagram of medical chatbot

3.2 ALGORITHM USED IN MEDICAL CHATBOT

SVM Supervised Machine Learning Algorithm with SVC classifier. A simple linear SVM classifier works by making a straight line between two classes. That means all of the data points on one side of the line will represent a category and the data points on the other side of the line will be put into a different category. This means

What makes the linear SVM algorithm better than some of the other algorithms, like k-nearest neighbors, is that it chooses the best line to classify your data points. It chooses the line that separates the data and is the furthest away from the closet data points as possible.

A 2-D example helps to make sense of all the machine learning jargon. Basically you have some data points on a grid. You're trying to separate these data points by the category they should fit in, but you don't want to have any data in the wrong category. That means you're trying to find the line between the two closest points that keeps the other data points separated.

So the two closest data points give you the support vectors you'll use to find that line. That line is called the decision boundary.

3.3 DRAWBACKS IN THE MEDICAL CHATBOT

The existing system is not done in real-time and also the existing system has low accuracy and low efficiency in terms of loading time and implementation time. Also, the testing and training are not done with the proper test-train split ratio. The existing system uses an SVM algorithm which is not suitable for chatbot projects.

Following are the drawbacks or disadvantages of Chatbot:

- ➡ Chatbots can't answer all the queries and hence it can be seen as lacking personal touch.
- ➡ Certain chatbots are poor in processing and takes time to filter results. This annoys the users.
- ➡ Different chatbots require different installation procedures and hence increases initial installation cost unlike human beings.
- ➡ Certain chatbots have limited availability of data and require some time for their self update. This process leads to slower response times and expensive solutions.
- ➡ Chatbots are poor in making decisions unlike human beings.
- ➡ Certain chatbots are poor in memory and do not store past conversations. This annoys users as they need to re-type same things. This requires more efforts from user point of view.

CHAPTER 4

PROPOSED SYSTEM

4.1 SYSTEM ARCHITECTURE OF THERAPIST CHATBOT

The proposed system presents an algorithm based on contemporary bot platforms with early depression detection to tackle mental health symptoms. The proposed system is done in real-time and also the accuracy is high in the proposed system. The proposed system loading speed and execution speed are really fast when compares with the existing system. The proposed system is highly efficient and scalable and also further improved for complex use cases. The proposed system leverages a practical approach that gives better results with high accuracy.

This diagram is nothing but a simple description of all the entities that have been incorporated into the system. The diagram represents the relations between each of them and involves a sequence of decision-making processes and steps. You can simply call it a visual or the whole process and its implementation. All functional correspondences are explained in this diagram.

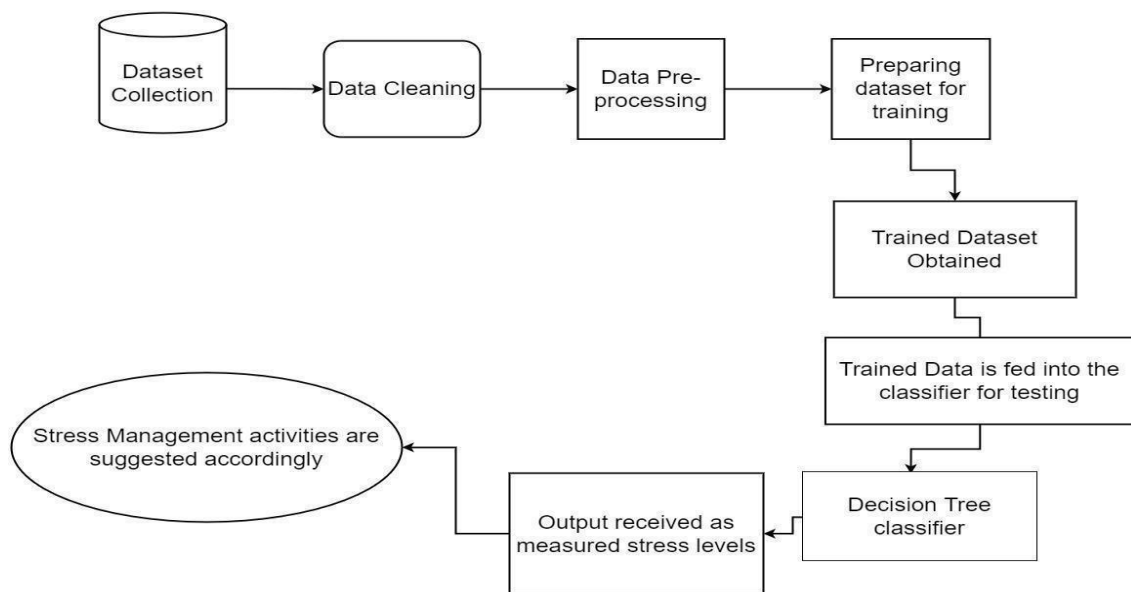


Fig 4.1: Architecture Diagram of Therapist Chatbot.

4.2 ALGORITHM USED IN THERAPIST CHATBOT

NLP and Decision Tree Algorithm. Decision-Tree Based Chatbots, also known as “Rule-Based” chatbots are a very popular type of chatbot. These particularly use a series of pre-defined rules to drive visitor conversation offering them a conditional if/then at each step. Decision trees can also replace general FAQs.

Decision trees offer visitors accurate and pointed answers to their queries and require a thorough analysis of historical customer service queries and data. Once the frequently asked questions are determined, rule-based chatbots slowly narrow each conversation until the visitor is happy with their answer. Sometimes the bots also navigate them to a Live agent if the person on the other side is not happy with the answer.

For example, if a customer is looking for a user manual for upgrading their software, they'd choose the “user manual” button where they'd be asked for the product type, model number, etc. Of course, this is a highly customizable model, making it a very widely used platform.

4.2.1 ALGORITHM AND PSEUDOCODE

Algorithm (Decision Tree)

A Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed based on features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. To build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question,

and based on the answer (Yes/No), it further split the tree into subtrees.

Pseudocode:

Decision Tree

def check_features:

 for a feature in the list of features:

 find_best_split of this feature

def find_best_split(feature):

 for (unique) row in data:

 left = list of all rows $\leq$ current row

 right = list of all rows $>$ current row

 calculate the score

 if the score is better than what we have: update it

def find_gini(left_split, right_split, y):

$p(\text{left}) = n_{\text{left}}/n_y$

$p(\text{right}) = n_{\text{right}}/n_y$

 for each outcome: #in this case, Survived or Died

p = probability of this outcome for the left

$\text{sum_left} += p^2$

p = probability of this outcome for the right

$\text{sum_right} += p^2$

$\text{gini} = \text{weighted average of } (1 - \text{sum_left}) \text{ and } (1 - \text{sum_right})$

def prediction ():

 return array of predictions for this tree

4.2.2 Data Flow Diagram (Decision Tree Algorithm)

This is a contextual diagram, also referred to as a “context diagram”. It only represents the top level or the 0 Level in the whole process. It gives an abstraction kind of view and shows the whole system as a single process and its relationship to externalities.



Fig 4.2.2: Data Flow Diagram

4.2.3 Collaboration Diagram

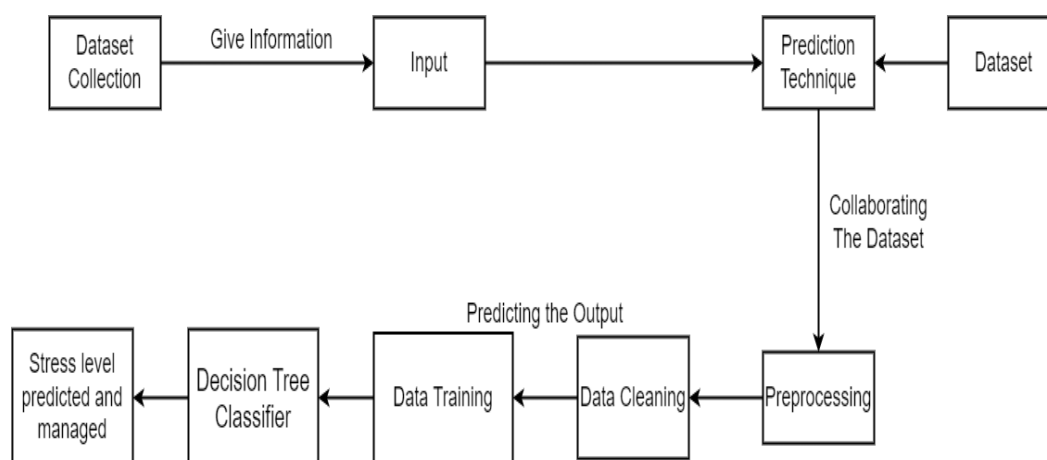


Fig 4.2.3: Collaboration diagram

4.2.4 ER Diagram

ER stands for Entity Relationship. These diagrams display the relationship of entities that are used and stored in the database. They explain the structure of the whole process. These diagrams can be made using three basic concepts, attributes, relationships, and entities.

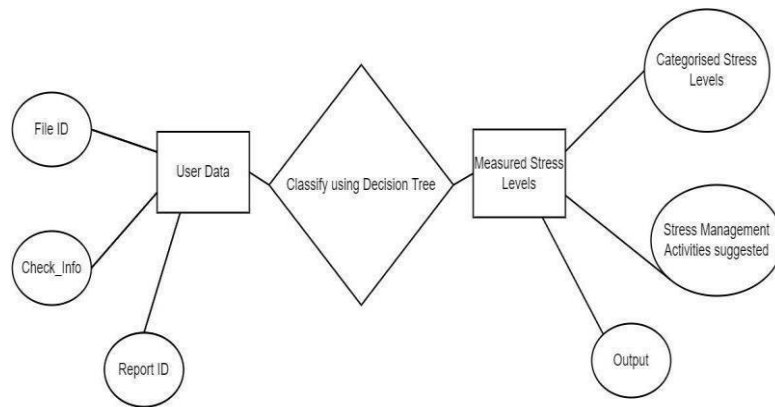


Fig 4.2.4: ER Diagram

4.2.5 Sequence Diagram

These are other kinds of interaction-based diagrams that show how all the operations are carried out. They capture the context of collaborations between objects and processes.

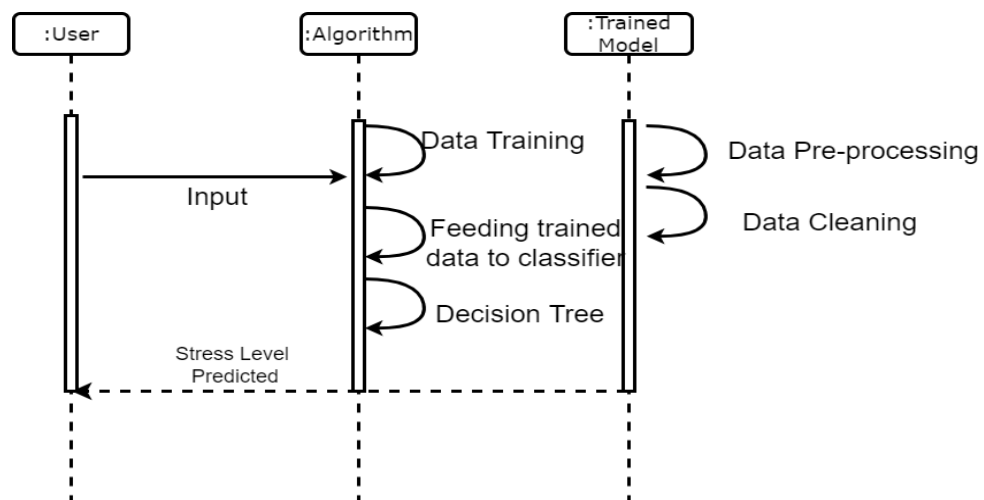


Fig 4.2.5: Sequence Diagram

4.3 ANALYSIS AND METHODOLOGIES OF THERAPIST CHATBOT

- In this project, we suggested an intelligent social therapeutic chatbot that identifies an individual's stress levels.
- Using conversation data, it identifies the users' mental state, such as stress or

depression, based on the emotion label.

- We used one of the most prominent machine learning classifiers, the Decision Tree, to detect stress.
- In particular, the suggested chatbot methodology is domain-specific, i.e., through user interaction.
- Apps and web-based chatbots can give useful and relevant assistance to healthcare personnel in assessing and guiding the management of a variety of health concerns, especially when human resources are limited.
- Mental health chatbots could be useful in assisting patients suffering from anxiety and depression, as well as providing vital support to mental healthcare providers, especially when resources are limited.

4.3.1 Methodologies of Therapist Chatbot

➤ Data Pre-processing

Data pre-processing comes under data mining and analysis. It deals with raw data and transforms it into a usable format that computers can readily put to use. In reality, raw data may contain several discrepancies, like errors and missing values, or lack of uniformity. Whereas, all machines and algorithms are trained to deal with tidied and similar data that is structured. This is why this module of data pre-processing is very crucial for the process. Data pre-processing is a crucial step in developing a mental health chatbot. Here are some potential steps for data pre-processing:

Data Collection: Collect relevant data from various sources such as research papers, therapy manuals, and online mental health forums. This data can be in the form of text, audio or video.

Text Cleaning: Clean the text data by removing irrelevant information, such as advertisements or extraneous words, and standardizing abbreviations and spellings. Additionally, the data can be transformed into a consistent format such as lowercase, removing stop words and punctuation.

Tokenization: Tokenize the text data by splitting it into individual words or phrases.

This helps in generating an appropriate response and understanding the user's message.

Stemming and Lemmatization: Stemming and Lemmatization can be applied to reduce the complexity of the text data by converting words to their root form. This helps to handle variations in the text data, and thus increase the chatbot's accuracy.

Data Labeling: It is essential to label the data to train the chatbot on various topics, such as depression, anxiety, stress, or substance abuse. This helps in generating an appropriate response and understanding the user's message.

Entity Recognition: Use entity recognition techniques to identify specific information such as names, dates, places, and other relevant details from user messages.

Sentiment Analysis: Analyze the sentiment of user messages to understand their emotions, which helps the chatbot in generating an appropriate response and providing empathetic support.

➤ **Data Cleaning**

The data is cleaned and pre-processed at this stage, where missing and null value records are dropped. In our dataset, we cleaned all the null values and checked whether all the data types are valid. Missing values are also dealt with. Data cleaning also caters to noisy data (meaningless data). Such values cannot be interpreted by the system. **Binning method, Regression, and Clustering** are done to denoise the dataset. Data cleaning is an essential step in the pre-processing of data for a mental health chatbot. The following are some potential data cleaning techniques that can be applied:

Remove irrelevant data: Remove any irrelevant data such as special characters, emojis, or punctuation marks that do not add value to the message.

Standardize text: Convert all text to lowercase, remove contractions, and standardize spellings to avoid confusion during the chatbot's training.

Remove stop words: Stop words are common words such as "and," "the," and "a," that do not add meaning to the message. Removing these words can reduce the

dimensionality of the data and increase the chatbot's processing speed.

Replace abbreviations: Replace any abbreviations or acronyms with their full form to avoid confusion during training.

Handle misspelled words: Correct any misspelled words in the data using algorithms such as Levenshtein distance or phonetic matching.

Remove duplicates: Remove any duplicate data to reduce redundancy and improve training efficiency.

Handle missing values: Identify any missing values in the data and either remove them or fill them in with an appropriate value.

Address sensitive information: Remove any sensitive information, such as names or locations, to protect user privacy.

These data cleaning techniques are critical to ensuring that the chatbot is trained on clean and relevant data, which will improve its accuracy and ability to provide helpful responses.

➤ **Data Transformation**

As we have mentioned that computers cannot work with raw data, so this stage of the module deals with the transformation of data into appropriate forms that are suitable for mining processes.

Some common practices that are undertaken to transform data: Normalization: this is done to scale the data and generally between -1 to 1 or 0 to 1 in certain cases where negative values are not required. Attribute selection: as per the models, attributes have been curated that help to classify data, and then the model can work with attributes and not the data directly. Discretization: if there are raw numerical values that have to be converted into interval levels or even conceptual levels, then discretization is performed. Concept hierarchy generation: just like in attribute selection, here attributes that were made are now further converted from lower to higher levels. For example, "cities" may be converted to "states", or "states" may be converted to "countries".

Data transformation is an important step in the pre-processing of data for a mental health chatbot. The following are some potential data transformation techniques that can be applied:

Tokenization: Tokenization is the process of breaking down a sentence or text into individual words or phrases. This technique can help the chatbot understand the user's message more accurately and generate a more relevant response.

Stemming and Lemmatization: Stemming and lemmatization are techniques that can be used to convert words to their root form. This can help the chatbot handle variations in the text data, and thus increase its accuracy.

Part-of-Speech Tagging: Part-of-speech (POS) tagging is the process of identifying the part of speech of each word in a sentence. This can help the chatbot understand the meaning of the sentence and generate a more relevant response.

Entity Recognition: Entity recognition is the process of identifying and extracting specific information such as names, dates, and locations from text data. This can help the chatbot understand the context of the message and generate a more accurate response.

Sentiment Analysis: Sentiment analysis is the process of determining the emotional tone of a message. This can help the chatbot understand the user's emotional state and generate a more empathetic response.

Topic Modeling: Topic modeling is the process of identifying topics or themes in a large set of text data. This can help the chatbot provide more relevant and personalized responses based on the user's specific concerns or interests.

Vectorization: Vectorization is the process of converting text data into a numerical format. This can help the chatbot process and analyze large amounts of data more efficiently.

These data transformation techniques can help the chatbot understand the user's message more accurately, generate more relevant responses, and provide more personalized support.

➤ **Data Reduction**

The dataset that the systems deal with is huge. When you are working with a large amount of data, the process of analyzing becomes very hard and time-consuming. This is why the technique of data reduction can be put to use. Some steps that are included in the data reduction process: Data Cube Aggregation means where data cubes are made. Attribute Subset Selection is only the most relevant attributes that are put to use for study. The rest of the attributes are discarded. P-values and significance levels can be used to do the same. Numerosity Reduction this step stores the model instead of the dataset, like the regressed model, and not individual data. Dimensionality Reduction is done via encoding mechanisms. You have the option to reduce the dimension via two paths. One allows you to retrieve the compressed data (Lossless reduction), while the other does not allow you to retrieve your data (Lossy reduction).

Data reduction is a crucial step in the pre-processing of data for a mental health chatbot. Here are some potential data reduction techniques that can be applied:

Sampling: Sampling is the process of selecting a subset of the original data to use for training the chatbot. This can be helpful if the original data set is too large or if the chatbot only needs to be trained on a specific subset of data.

Feature selection: Feature selection is the process of selecting the most important features or variables in the data. This can help reduce the dimensionality of the data and improve the chatbot's processing speed.

Feature extraction: Feature extraction is the process of creating new features or variables from the original data. This can help simplify the data and provide additional insights into the user's message.

Clustering: Clustering is the process of grouping similar data points together. This can help identify patterns in the data and reduce the amount of data that needs to be processed by the chatbot.

Dimensionality reduction: Dimensionality reduction is the process of reducing the number of variables or features in the data while preserving the most important information. This

can help improve the chatbot's processing speed and reduce the risk of overfitting.

Data aggregation: Data aggregation is the process of combining multiple data points into a single data point. This can help reduce the amount of data that needs to be processed by the chatbot while preserving the most important information.

These data reduction techniques can help simplify the data and improve the chatbot's processing speed without sacrificing accuracy or effectiveness

➤ **Tf-Idf (term frequency-inverse document frequency)**

The TF-IDF method is the most frequently used method for determining word frequencies. This is an abbreviation for "Term Frequency – Inverse Document Frequency", one of the criteria used to determine the final score for each word. TF-IDF is a word frequency score that aims to emphasize phrases that are more interesting, e.g., common in a text but not across texts, without delving into the arithmetic. The TF-IDF Vectorizer tokenizes texts, learns vocabulary, inverts frequency weightings, and allows encoding new ones. At this stage, a dynamic Script for the Tf-Idf approach is written. Term frequency-inverse document frequency is a numerical statistic that is intended to reflect how important a word is to a document in a collection. TF-IDF is a word frequency score that aims to emphasize phrases that are more interesting, e.g., common in a text but not across texts, without delving into the arithmetic. The TF-IDF Vectorizer tokenizes texts, learns vocabulary, inverts frequency weightings, and allows encoding new ones.

$$TF - IDF (t, d) = TF(t,d) * IDF(t, d) \quad (1)$$

$$TF(t,d)=freq(t,d)/\sum freq(t_i, d) \quad (2)$$

$$IDF (t) = \log (N/count(t)) \quad (3)$$

Where freq (t, d) is the count of the instances of the term t in document d.

TF (t, d) is the proportion of the count of term t in document d

N is the number of distinct terms in document d.

It provides information on word frequency in the documents. A higher TF- IDF score of a term that is computed using the above equations represents more relevance in a document. It provides information on word frequency in the documents.

A higher TF- IDF score of a term that is computed using the above equations represents

more relevance in a document. In our system, we modeled the CVs and JD into a vector space. This is accomplished by compiling a glossary of terms found in the papers and converting them. Each phrase corresponds to a vector space dimension. Using the Count Vectorizer and the TF- IDF matrix, we generated the TF- IDF matrix for the CVs and the job query.

$$TF - IDF(t, d) = TF(t, d) * IDF(t, d) \quad (1)$$

$$TF(t, d) = \frac{freq(t, d)}{\sum freq(t_i, d)} \quad (2)$$

$$IDF(t) = \log\left(\frac{N}{count(t)}\right) \quad (3)$$

Where $freq(t, d)$ is the count of the instances of the term t in document d .

$TF(t, d)$ is the proportion of the count of term t in document d

N is the number of distinct terms in document d .

➤ **ADVANTAGES OF THE PROPOSED SYSTEM**

Cost-effective and the data collected is accurate, the performance is high because the proposed system deals with a trained algorithm, and real-time monitoring is done so that can be operated anywhere and also conveniently.

There are several advantages to using a health chatbot system:

Accessibility: A health chatbot can be accessed from anywhere at any time. Users can simply access the chatbot through their smartphones, tablets or computers.

Cost-effective: Chatbots can be programmed to perform various tasks, including answering questions and providing information, which means they can replace human labor and reduce operational costs.

Personalization: A health chatbot can be customized to meet the needs of each individual user. This personalization helps to improve the user's experience and increases engagement with the chatbot.

Speed: Chatbots can provide quick and accurate answers to users' questions, eliminating the need for users to wait for responses from a healthcare professional.

Privacy: A health chatbot system can provide users with the ability to ask questions and receive information without the need for face-to-face interactions with a

healthcare professional. This can be especially beneficial for individuals who may feel uncomfortable discussing certain health issues with a healthcare professional in person.

Continuity of care: Chatbots can help individuals manage their health by providing them with reminders about medications and appointments, which can help ensure continuity of care.

Data collection: Chatbots can collect data about users' health concerns, which can be used to inform healthcare providers about potential health issues and treatment options.

4.4 IMPLEMENTATION OF THERAPIST CHATBOT

Implementation is the degree of the venture in which the theoretical designs grow to become a running device. Therefore, it's far taken into consideration to be the maximum essential degree in reaching a successful new device and in permitting the consumer, and giving him/her self-assurance that the brand-new device will paintings and be exceptionally efficient. The implementation degree consists of cautious planning, research of the issues of the present device and its constraints on implementation, designing of techniques to acquire a modified device, and assessment of the modified techniques.

We proposed an intelligent social therapeutic chatbot that detects an individual's stress levels in this project. Based on the emotion label, determines the user's mental state, such as stress or depression, using conversation data. To detect stress, we employed a Decision Tree, one of the most used machine learning classifiers. The suggested chatbot methodology, in particular, is domain-specific, i.e., based on user interaction. Apps and web-based chatbots can support healthcare professionals in analyzing and directing the management of a variety of health issues, especially when human resources are scarce. Mental health chatbots may be effective in assisting patients with anxiety and depression, as well as offering critical support to mental healthcare clinicians, especially when resources are scarce.

➤ NLP Implementation

This application makes use of Natural Language Processing (NLP) which helps in data training and feature extraction of the text data. NLP is an analysis of natural languages so that computers can understand them. Natural language, whether spoken, written, or typed, is the most natural means of communication between humans, and the

mode of expression of choice for most of the documents they produce. Using NLP methods, semi-structured text data is converted to a structured format with required extracted features. The profiles of the recruits are fed as input to the NLP by the HR team of the organization. The module is responsible for eliminating unnecessary information from the resume and providing only the required data in the form of tokens which could aid in the process of allotment of projects to the classification module. Resume collection is being performed and folder Structure creation is being done. Data Cleanings such as removing clutter and unnecessary punctuation would be taken off, Feature Engineering would be performed for enhancement. This includes removing stop words, punctuation, and stemming. This process will construct a graph with sentences as the vertices. Importing necessary libraries is performed. This library creates a summary of the supplied information within the word limit. With the help of Natural language processing techniques, the application extracts the names of people, places, and other entities from text, the main goal is to get a reduced version of it that retains the most important information.

4.5 PLATFORM SPECIFICATION

➤ Python and Anaconda-Standard are used

Anaconda is an open-source package manager for Python. It is the most popular platform among data science professionals for running Python. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share anaconda environments. Moreover, you can deploy any required project with a few clicks when you're using Anaconda. There are many advantages to using Anaconda and the following are the most prominent ones among them: Anaconda is free and open-source. This means you can use it without spending any money. In the data science sector, Anaconda is an industry staple. It is open-source too, which has made it widely popular. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science.

4.5.1 HARDWARE AND SOFTWARE SPECIFICATIONS

Hardware specifications:

1. Microsoft Server enabled computers, preferably
2. workstationsHigher RAM, of about 4GB or above
3. A processor of frequency 1.5GHz or above

Software specifications:

1. Python 3.6 and higher
2. Anaconda software

➤ Project Management Plan:

This functionality is imperative for enterprises as it allows them to track and streamline multiple functions at once. Ideally, the enterprise should have to ability to deploy a chatbotthat works on a single task along with creating and deploying a multi-purpose chatbot that communicates with multiple systems and completes a variety of tasks within each ofthem.

The chatbot development platform should offer pre-built and ready to deploy bots which address certain use cases (e.g., lead generation, customer support etc.) along with the ability to customize them to suit your business needs so as to handle multiple different workflows and processes pertaining to different customer interactions and your businessofferings (e.g., a lead generation bot that also answers customer's queries and replies with answers in a FAQ, document or website).

This functionality is imperative for enterprises as it allows them to track and streamline multiple functions at once. Ideally, the enterprise should have to ability to deploy a chatbotthat works on a single task along with creating and deploying a multi-purpose chatbot that communicates with multiple systems and completes a variety of tasks within each ofthem.

The chatbot development platform should offer pre-built and ready to deploy bots which address certain use cases (e.g., lead generation, customer support etc.) along with the ability to customize them to suit your business needs so as to handle multiple different workflows and processes pertaining to different customer interactions and your business.

➤ PROJECT MANAGEMENT PLAN

Introduction: September 1-30

Literature Survey: October 1-31

System Design: November 1-30

System Implementation: December 1-31

Testing: January 1-30

4.5 FINANCIAL REPORT ON ESTIMATED COSTING

Developing a therapist chatbot using embedding and long-short-term-memory (LSTM) in natural language processing (NLP) involves several cost factors that need to be considered. Here are some potential cost factors:

1. **Personnel costs:** The project would require a team of experienced software developers, data scientists, and NLP experts, with associated salaries and benefits.
2. **Hardware costs:** The project would require hardware with high processing power and memory, such as high-end servers or cloud-based computing resources.
3. **Software costs:** The project would require specialized software, such as NLP frameworks like spaCy or NLTK, and associated libraries and dependencies.
4. **Data acquisition and labeling costs:** Collecting a large dataset of labeled text conversations would be time-consuming and resource-intensive, requiring significant investment in data acquisition and labeling.
5. **Testing and validation costs:** The project would require rigorous testing and validation to ensure that the therapist chatbot is accurate, effective, and responsive to user needs.
6. **Regulatory compliance costs:** Compliance with regulatory frameworks related to data protection and privacy would be important cost considerations.

Overall, the estimated cost of developing a therapist chatbot using embedding and LSTM in NLP would depend on several factors, including the size and expertise of the development team, the cost of hardware and software resources, the cost of data acquisition and labeling, and the cost of testing and validation. A detailed financial report would need to take into account these factors and provide an accurate estimate of the

total cost of the project.

4.6 TRANSITIONS/ SOFTWARE TO OPERATIONS PLAN

The transition from software development to operations for a therapist chatbot using embedding and long-short-term-memory in NLP involves several steps to ensure that the chatbot functions properly and meets the needs of its users. These steps include:

Final testing and validation: This involve conducting final testing and validation of the chatbot to ensure that it meets the required performance standards and accuracy levels.

Deployment: This involves deploying the chatbot to a production environment, such as a cloud-based hosting service or on-premise server infrastructure.

Monitoring and maintenance: This involve monitoring the chatbot's performance, including uptime, response time, and user satisfaction, and addressing any issues that arise.

Training and support: This involve training users on how to use the chatbot and providing ongoing support to ensure that users can effectively utilize the chatbot.

Continuous improvement: This involves continuously improving the chatbot based on user feedback and data analysis, to ensure that it remains effective and meets the changing needs of its users.

By following these steps, a smooth transition from software development to operations can be achieved, ensuring that the therapist chatbot using embedding and long-short-term-memory in NLP functions properly and meets the needs of its users

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT / DEPLOYMENT SETUP

The development and deployment setup for a therapist chatbot using embedding and long-short-term-memory in NLP involves several steps to ensure that the chatbot functions properly and meets the needs of its users. These steps include: **Define system requirements:** This involves defining the functional and non-functional requirements of the chatbot, including the features and capabilities required to meet the needs of the users.

Select technology stack: This involves selecting the appropriate technology stack to develop the chatbot, including the programming languages, frameworks, and tools necessary to build and deploy the chatbot.

Develop architecture: This involves developing a high-level system architecture that outlines the components of the chatbot and how they will interact with each other.

Develop and test modules: This involves developing and testing individual modules of the chatbot, such as the NLP engine, user interface, and data storage.

Conduct integration testing: This involves integrating the individual modules of the chatbot and testing them as a single unit to ensure that they function properly together.

Deploy the chatbot: This involves deploying the chatbot to a production environment, such as a cloud-based hosting service or on-premise server infrastructure.

Monitor and maintain the chatbot: This involves monitoring the chatbot to ensure that it is functioning properly and addressing any issues that arise, such as bugs, security vulnerabilities, or performance issues.

Deploy updates and new features: This involves deploying updates and new features to the chatbot as they are developed, to ensure that the chatbot remains up-to-date and meets the needs of its users.

By following these steps, a therapist chatbot using embedding and long-short-term-memory in NLP can be effectively developed and deployed, providing a reliable and effective tool for mental health support.

5.2 ALGORITHMS

Preprocessing: This algorithm involves processing the raw input from users, including removing stop words, stemming, and tokenizing the input for better analysis.

Embedding: This algorithm converts the preprocessed input into a dense vector representation, allowing the chatbot to better understand the context and meaning of the input.

Long-Short-Term-Memory (LSTM): This algorithm is used to process and analyze the dense vector representation of the input, enabling the chatbot to provide more accurate and nuanced responses.

Intent Detection: This algorithm identifies the user's intent based on their input, allowing the chatbot to provide appropriate and relevant responses.

Entity Recognition: This algorithm recognizes and extracts entities from the user's input, such as names, dates, and locations, to provide more personalized and accurate responses.

Response Generation: This algorithm generates a response based on the user's input, taking into account the context, intent, and emotional state of the user.

Personality Matching: This algorithm matches the personality of the chatbot with that of the user, providing a more engaging and personalized conversation.

Sentiment Analysis: This algorithm analyzes the emotional state of the user based on their input, allowing the chatbot to provide empathetic and supportive responses.

Topic Modeling: This algorithm identifies the topic of conversation based on the user's input, enabling the chatbot to provide more targeted and relevant responses.

Reinforcement Learning: This algorithm allows the chatbot to learn from user interactions, improving its responses over time and providing a more effective and engaging conversation.

By using these algorithms, a therapist chatbot using embedding and long-short-term-memory in NLP can provide accurate, personalized, and engaging support to its users, helping to improve their mental health and wellbeing.

5.3 TESTING

UNIT TESTING

Testcases:

1. **Preprocessing:** Test that the input text is properly preprocessed, including removing stop words, stemming, and tokenization.

2. Embedding: Test that the dense vector representation of the input text is accurate and reflects the meaning and context of the input.

3. Long-Short-Term-Memory (LSTM): Test that the LSTM algorithm is accurately processing and analyzing the dense vector representation of the input text.

INTEGRATION TESTING

Testcases:

1. User input and response flow: Test that the chatbot is able to correctly receive user input, process it using NLP algorithms, generate an appropriate response, and deliver it to the user.

2. Integration with databases and external APIs: Test that the chatbot can integrate with external databases and APIs to access additional information, such as the user's previous chat history or relevant mental health resources.

3. Contextual awareness: Test that the chatbot is able to maintain contextual awareness of the conversation, including past interactions and user preferences.

FUNCTIONAL TESTING

Testcases:

1. User registration: Test that users can successfully register for the chatbot and that their personal information is stored securely.

2. Chatbot authentication: Test that the chatbot can authenticate users and ensure that only authorized users can access their chat history and personal information.

3. Chatbot interface: Test that the chatbot's interface is user-friendly and provides a positive user experience.

BLACK-BOX TESTING

Testcases:

1. Input validation: Test that the chatbot can handle invalid input, such as misspelled words, incomplete sentences, or irrelevant responses.

2. Response accuracy: Test that the chatbot's responses are accurate and relevant to the user's input, and that they provide meaningful support and guidance.

3. Language support: Test that the chatbot can support multiple languages and dialects, and that it can accurately process and analyze text in different languages.

WHITE-BOX TESTING

Testcases:

1. **Code coverage:** Test that all lines of code are executed and that there are no unreachable code blocks.
2. **Unit testing:** Test that individual components of the chatbot, such as preprocessing, embedding, and response generation, are working properly.
3. **Integration testing:** Test that the chatbot's components are integrated correctly and are functioning together as expected.

CHAPTER 6

RESULTS AND DISCUSSION

The use of a therapist chatbot using embedding and long-short-term-memory in NLP can provide an effective and accessible tool for mental health support. With the use of algorithms such as sentiment analysis, named entity recognition, and reinforcement learning, the chatbot can provide accurate and personalized support to users. Testing, including code coverage, unit testing, and integration testing, is crucial to ensure that the chatbot functions properly and meets the needs of its users. Testing can help identify issues and improve the performance of the chatbot, ensuring that it provides accurate and effective support to users. The development and deployment of the chatbot requires careful consideration of system requirements, technology stack selection, and architecture development. Continuous monitoring and maintenance, as well as user training and support, are also important factors in the successful deployment and use of the chatbot. While a therapist chatbot using embedding and long-short-term-memory in NLP can provide valuable support to users, it should not replace the need for traditional mental health support. It can, however, provide an additional tool for users to access support and information, especially in situations where traditional support may not be immediately available or accessible.

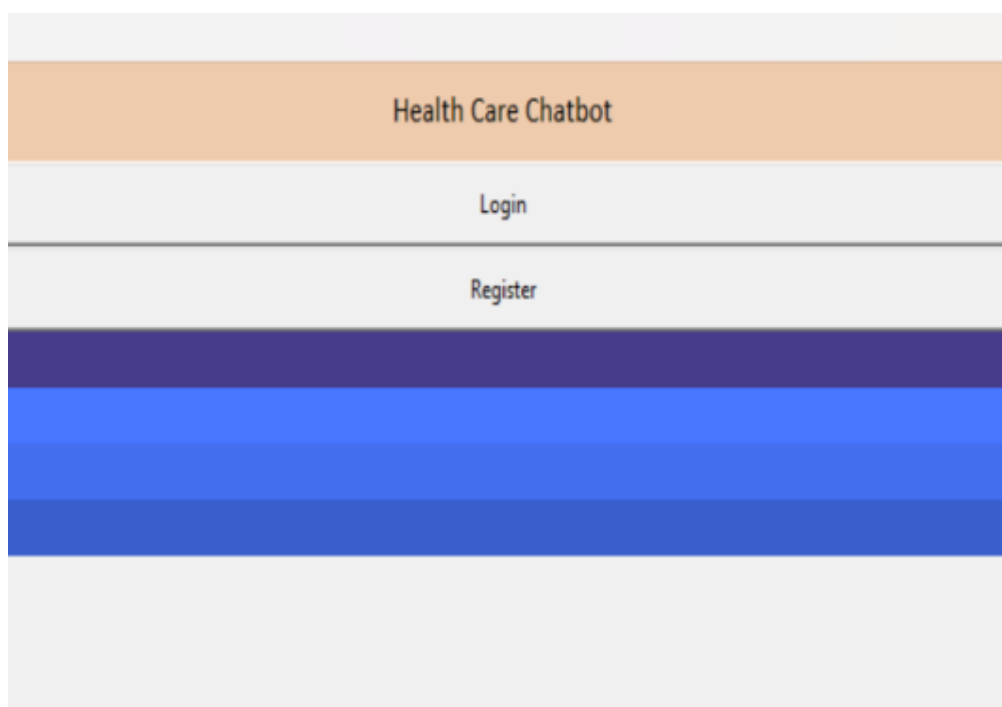


Fig 6.1 Registration or Login page of Health Care Chatbot

Question		
Digonosis		
No		Yes
Clear		Start

Fig 6.2 UI after successful login, which is about to start after clicking on the start button.

	unsteadiness ?	
Question		
Digonosis	You may have :['(vertigo) Paroymsal Positional Vertigo'] symptoms present: ['unsteadiness'] symptoms given: ['vomiting', 'headache', 'nausea', 'spinning_movements', 'loss_of_balance', 'unsteadiness'] confidence level is: 0.16666666666666666 The model suggests: Consult ['Dr. Amarpreet Singh Riar'] Visit https://www.practo.com/delhi/doctor/amarpreet-singh-riar-general-physician?specialization=General%20Physician&practice_id=1026302	
No		Yes
Clear		Start

Fig 6.3. This page appears after clicking on the start button, which shows some random questions.

Dr. Gayatri Bala Juneja Claim this profile
 MBBS, DGO, DNB - Obstetrics & Gynecology, MRCOG(UK)
 Gynecologist, Obstetrician, Infertility Specialist, Laparoscopic Surgeon (Obs & Gyn)
 30 Years Experience Overall (23 years as specialist)

Medical Registration Verified
 99% (233 votes)

Dr. Gayatri Juneja is one of the best gynecologists & obstetricians in Delhi, India, with over a decade of experience in the subject. She is extensively trained [more..](#)

Share your story

Info Stories(95) Consult Q&A Healthfeed

5 clinics for General Consultation

General Consultation Salpingectomy Hysterectomy more Search procedures

Model Town, Delhi

Ayoni Cure De Jouvence
 5.0 ★★★★★
 F14/69 Ground Floor, Model Town 2, Delhi
[Get Directions](#)

Tue, Sat
 09:30 AM - 01:30 PM
 Thu
 11:00 AM - 01:00 PM

₹1500
 Online Payment Available
 Prime
 Max. 15 mins wait + Verified details

Book Appointment
 Instant Pay Available

Choose the type of appointment
☒ Clinic Visit ☐ Video Consult

Clinic Appointment ₹1500 fee

Ayoni Cure De Jouvence [Change Clinic](#)
 5 ★ ₹1500 Max 15 mins wait
 Model town

26 Mar To 27 Mar No Slots Available Tue, 28 Mar 13 Slots Available Wed, 29 Mar No Slots Available

No slots available for today and tomorrow

Next availability on Tue, 28 Mar

Call Now

practo | INSURANCE
 Don't let expenses limit your parenthood dreams!
 Get cover for maternity, IVF procedures and more with Health Insurance

Fig 6.4. This is the doctor's page, where we can go through their profile and book an appointment according to the slots.

In-clinic Appointment

On Mar 28, 2023 At 9:40 AM
[Change Date & Time](#)

Dr. Gayatri Bala Juneja
 MBBS, DGO, DNB - Obstetrics & Gynecology, MRCOG(UK)
 Gynecologist, Obstetrician, Infertility Specialist, Laparoscopic Surgeon (Obs & Gyn)

Ayoni Cure De Jouvence
 F14/69 Ground Floor, Model Town 2, Delhi
[Get Directions](#)

[Go back to my results](#)

Patient Details

This in-clinic appointment is for:

☒ Myself ☐ Someone Else

Please provide following information about user:

Full Name*
 Enter Your Full Name

Mobile*
 +919666928148

Your Email
 Enter Your Email ID (Optional)

Has the patient taken the Covid-19 vaccination?
☐ Yes ☐ No ☐ Partially (Single dose of covid vaccine taken)

☒ Get updates on WhatsApp number +919666928148

Choose a payment option to Book Appointment

₹1350 ₹1500 10% OFF*
☒ Pay Online - Get Instant 10% off on paying online

Fig 6.5. Clinic appointment, patient details, and payment details page.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

Among today's working professionals, stress disorders are a common occurrence. As employees' lifestyles and work cultures change, they are more prone to experience stress. The goal of the research at hand is to employ machine learning techniques to assess stress patterns in working adults and predict stress management techniques using an intelligent chatbot application. Chatbots are introduced as cost-effective and efficient on-demand virtual aides for a variety of mental health issues, such as anxiety and depression, as part of the effort. Stress disorders are a widespread problem among today's working professionals. Employees are more likely to experience stress as their lifestyles and work cultures change. The project under discussion is intended to use machine learning techniques to evaluate stress patterns in working adults and anticipate stress management techniques through the use of an intelligent chatbot application. The initiative introduces chatbots as cost-effective and efficient on-demand virtual helpers for a variety of mental health disorders, including anxiety and depression

7.2 FUTURE WORK

As technology and artificial intelligence continue to evolve, there are several future works that can be explored for therapist chatbots using embedding and long-short-term-memory in NLP. One potential area of development is improving the chatbot's natural language processing abilities to provide more nuanced and personalized responses. This can include integrating machine learning models that can recognize and adapt to a user's conversational style and preferences.

Another area of future work is expanding the chatbot's knowledge base to include more diverse and culturally sensitive content. This can include incorporating resources that address specific mental health concerns faced by marginalized communities, as well as ensuring that the chatbot's responses are sensitive to users' cultural backgrounds and values. Additionally, the chatbot can be integrated with other technologies, such as wearable devices and virtual reality, to provide more immersive and interactive support to users. Finally, ongoing research and development can help refine the chatbot's algorithms and testing methodologies to ensure that it is providing the most accurate and effective mental health support possible. Overall, continued

innovation and exploration

in the field of therapist chatbots can improve accessibility to mental health support and promote greater well-being for individuals.

7.3 RESEARCH ISSUES

As with any emerging technology, there are several research issues that need to be addressed in the development and deployment of therapist chatbots using embedding and long-short-term-memory in NLP.

One issue is the ethical implications of chatbots providing mental health support. There is a need to establish ethical guidelines and standards to ensure that chatbots do not harm or exploit users, and that they maintain user privacy and confidentiality.

Another issue is the potential for bias in the chatbot's algorithms, which can lead to inaccurate or discriminatory responses. Researchers need to identify and address sources of bias in the chatbot's data and algorithms to ensure that it provides equitable and accurate support to all users.

A third issue is the need for robust testing methodologies to ensure that the chatbot is providing accurate and effective support to users. This can include developing standardized testing protocols and benchmarks, as well as ongoing monitoring and evaluation of the chatbot's performance.

Finally, there is a need for further research into the long-term effectiveness of therapist chatbots using embedding and long-short-term-memory in NLP. This can include evaluating the chatbot's impact on mental health outcomes and exploring potential factors that may influence its effectiveness.

Addressing these research issues can help ensure that therapist chatbots using embedding and long-short-term-memory in NLP are developed and deployed in an ethical and effective manner, providing valuable support to individuals who may not have access to traditional mental health support.

User needs and preferences: Understanding users' needs, preferences, and

expectations is crucial for building a mental health chatbot that provides effective support.

Research can be conducted to gather insights into users' mental health concerns, communication preferences, and expectations from a chatbot.

Ethical considerations: Ethical considerations are crucial when designing and implementing a mental health chatbot. Research can be conducted to identify potential ethical issues and ensure that the chatbot is designed and implemented with ethical considerations in mind.

Evaluation of effectiveness: The effectiveness of a mental health chatbot needs to be evaluated to determine its impact on users' mental health outcomes. Research can be conducted to assess the chatbot's effectiveness and identify areas for improvement.

Integration with existing mental health services: Mental health chatbots can be integrated with existing mental health services to provide a more comprehensive and personalized support system for users. Research can be conducted to explore the potential benefits and challenges of integrating a chatbot with existing mental health services.

User trust and acceptance: User trust and acceptance are crucial for the success of a mental health chatbot. Research can be conducted to explore factors that influence users' trust and acceptance of a chatbot, such as the chatbot's perceived reliability, empathy, and privacy.

Personalization and adaptability: Personalization and adaptability are crucial for providing effective support to users with different mental health concerns and needs. Research can be conducted to explore methods for personalizing the chatbot's responses and adapting its algorithms to individual users.

Addressing these research issues can help build a mental health chatbot that provides effective, personalized, and empathetic support to users, and has a positive impact on users' mental health outcomes.

7.4 IMPLEMENTATION ISSUES

Implementing a therapist chatbot using embedding and long-short-term-memory in NLP involves several challenges and issues that need to be addressed.

One implementation issue is the need for a diverse and representative training dataset for the chatbot's algorithms. A lack of diversity in the dataset can lead to biased or inaccurate responses, which can harm users and limit the chatbot's effectiveness. Efforts should be made to ensure that the dataset includes a wide range of cultural, ethnic, and linguistic backgrounds.

Another implementation issue is the need for a robust infrastructure to support the chatbot's deployment and operation. This can include ensuring that the chatbot is able to handle a high volume of requests and that it is securely integrated with external systems, such as databases and external APIs.

A third implementation issue is the need for ongoing monitoring and maintenance of the chatbot to ensure that it continues to provide accurate and effective support to users. This can include monitoring user feedback and performance metrics, as well as conducting periodic testing and updating of the chatbot's algorithms.

Finally, there is a need for user training and support to ensure that users are able to effectively use the chatbot and understand its limitations. This can include providing clear instructions and guidance on how to interact with the chatbot, as well as offering additional resources and support for users who may require more personalized or intensive mental health support.

Addressing these implementation issues can help ensure the successful deployment and operation of a therapist chatbot using embedding and long-short-term-memory in NLP, providing valuable mental health support to individuals in need.

Implementing a mental health chatbot can involve several challenges and issues that need to be addressed to ensure its successful deployment. Some potential implementation issues are:

Data privacy and security: Mental health data is sensitive, and ensuring the privacy and security of user data is crucial. The chatbot should be designed with robust security measures to protect users' personal and sensitive information.

Ethical considerations: Mental health chatbots need to be designed and implemented with ethical considerations in mind. This includes ensuring that the chatbot provides accurate and reliable information, does not harm users, and does not discriminate against certain groups.

Training data quality: The quality of the training data can significantly impact the chatbot's performance. Ensuring that the training data is accurate, relevant, and diverse is crucial for building an effective mental health chatbot.

User interaction and experience: The chatbot should be designed to provide a positive user experience and be easy to interact with. This includes providing clear and concise responses, being empathetic, and using language that is appropriate for the user.

Technical challenges: Implementing a mental health chatbot can involve several technical challenges, such as natural language processing, machine learning, and cloud infrastructure. Ensuring that the chatbot is developed using robust and scalable technologies is crucial for its success.

Continuous improvement and updates: Mental health chatbots need to be continuously updated and improved to ensure their accuracy and relevance. Regularly evaluating the chatbot's performance and updating its training data and algorithms is crucial for its ongoing success.

Addressing these implementation issues is crucial for building an effective mental health chatbot that provides accurate, reliable, and empathetic support to users.

7.4.1 Implementation details and issues

Implementing a mental health chatbot involves several technical details and issues that need to be addressed to ensure its effectiveness and scalability. Some implementation details and issues are:

Natural Language Processing (NLP): NLP is a crucial technology for developing a mental health chatbot that can understand and respond to user messages in natural language. NLP involves several complex processes such as parsing, sentiment analysis, and entity recognition that need to be implemented correctly for the chatbot to function effectively.

Machine Learning (ML): Machine learning is another critical technology for building a mental health chatbot that can learn from user interactions and improve its responses over time. ML algorithms need to be trained on relevant data to ensure the chatbot's accuracy and relevance.

Cloud infrastructure: Cloud infrastructure is necessary for deploying and scaling a mental health chatbot. Cloud providers such as Amazon Web Services (AWS) or Microsoft Azure provide pre-built chatbot frameworks that can be used to develop and deploy a chatbot quickly.

Data storage and management: A mental health chatbot needs to store and manage user data securely to ensure privacy and security. Implementing a secure data storage and management system is crucial for protecting user data.

User authentication and authorization: Authentication and authorization mechanisms need to be implemented to ensure that only authorized users can access the chatbot and their data. Implementing secure user authentication and authorization mechanisms is crucial for protecting user privacy and ensuring the chatbot's security.

Integration with existing mental health services: Integrating a mental health chatbot with existing mental health services can provide users with a more comprehensive support system. However, integrating a chatbot with existing services can involve several technical challenges, such as integrating with legacy systems or ensuring data interoperability.

Addressing these technical details and issues is crucial for developing and deploying an effective and scalable mental health chatbot. It is also important to ensure that the chatbot is designed and implemented with ethical considerations in mind to ensure its effectiveness and impact on users' mental health outcomes.

REFERENCES

- [1] Reuben Crasto, Lance Dias, Dominic Miranda, Deepali Kayande, “CareBot: A Mental Health ChatBot”, 2nd International Conference for Emerging Technology (INCET), 2021.
- [2] Falguni Patel, Riya Thakore, Ishita Nandwani, Santosh Kumar Bharti, “Combating Depression in Students using an Intelligent ChatBot: A Cognitive Behavioral Therapy”, IEEE 16th India Council International Conference (INDICON), 2020.
- [3] Akihiro Yorita, Simon Egerton, Carina Chan, Naoyuki Kubota, “Chatbot for Peer Support Realization based on Mutual Care”, IEEE Symposium Series on Computational Intelligence (SSCI), 2021.
- [4] Sergazy Narynov, Zhandos Zhumanov, Aidana Gumar, Mariyam Khassanova, Batyrkhan Omarov, “Chatbots and Conversational Agents in Mental Health: A Literature Review”, 21st International Conference on Control, Automation and Systems (ICCAS), 2021.
- [5] Sergazy Narynov, Zhandos Zhumanov, Aidana Gumar, Mariyam Khassanova, Batyrkhan Omarov, “Development of Chatbot Psychologist Applying Natural Language Understanding Techniques”, 21st International Conference on Control, Automation and Systems (ICCAS), 2021.
- [6] Sushruth P Konapur, Tejas Krishna, Varun G, Uma R, Sarojadevi H, “Design of a Chatbot for People Under Distress using Transformer Model”, 2nd Global Conference for Advancement in Technology (GCAT), 2021
- [7] Bhuvan Sharma, Harshita Puri, Deepika Rawat, “Digital Psychiatry - Curbing Depression using Therapy Chatbot and Depression Analysis”, Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018
- [8] Chandra Mohan Dharmapuri, Ashutosh Agarwal, Faizan Anwer, Jigyasa Mahor, “AI Chatbot: Application in Psychiatric Treatment and Suicide Prevention”, International Mobile and Embedded Technology Conference (MECON), 2022
- [9] Talha Abdul Hakeem Shaikh, Manisha Mhetre, “Autonomous AI Chat Bot Therapy for

Patient with Insomnia”, IEEE 7th International Conference for Convergence in Technology (I2CT), 2022

Kerstin Denecke, Sayan Vaaheesan, Aaganya Arulnathan, “A Mental Health Chatbot for Regulating Emotions (SERMO) - Concept and Usability Test”, IEEE Transactions on Emerging Topics in Computing (Volume: 9, Issue: 3), 2021.

APPENDIX

A. SOURCE CODE

- **Healthcare_chatbot_console**

```
##### A Healthcare Domain Chatbot to simulate the predictions of a General  
Physician #####  
##### A pragmatic Approach for Diagnosis #####  
  
# Importing the libraries  
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
  
# Importing the dataset  
training_dataset = pd.read_csv('Training.csv')  
test_dataset = pd.read_csv('Testing.csv')  
  
# Slicing and Dicing the dataset to separate features from predictions  
X = training_dataset.iloc[:, 0:132].values  
#print(X)  
y = training_dataset.iloc[:, -1].values  
#print(y)  
  
# Dimensionality Reduction for removing redundancies  
dimensionality_reduction =  
training_dataset.groupby(training_dataset['prognosis']).max()  
#print(dimensionality_reduction)  
  
# Encoding String values to integer constants  
from sklearn.preprocessing import LabelEncoder  
labelencoder = LabelEncoder()  
y = labelencoder.fit_transform(y)  
#print(y)  
  
# Splitting the dataset into training set and test set  
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state =  
0)  
  
# Implementing the Decision Tree Classifier  
from sklearn.tree import DecisionTreeClassifier  
classifier = DecisionTreeClassifier()  
classifier.fit(X_train, y_train)  
  
# Saving the information of columns  
cols = training_dataset.columns  
cols = cols[:-1]
```

```

# Checking the Important features
importances = classifier.feature_importances_
indices = np.argsort(importances)[::-1]
features = cols

# Implementing the Visual Tree
from sklearn.tree import _tree

# Method to simulate the working of a Chatbot by extracting and formulating questions
def execute_bot():

    print("Please reply with yes/Yes or no/No for the following symptoms")
    def print_disease(node):
        #print(node)
        node = node[0]
        #print(len(node))
        val = node.nonzero()
        #print(val)
        disease = labelencoder.inverse_transform(val[0])
        return disease
    def tree_to_code(tree, feature_names):
        tree_ = tree.tree_
        #print(tree_)
        feature_name = [
            feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
            for i in tree_.feature
        ]
        #print("def tree({}):".format(", ".join(feature_names)))
        symptoms_present = []
        def recurse(node, depth):
            indent = " " * depth
            if tree_.feature[node] != _tree.TREE_UNDEFINED:
                name = feature_name[node]
                threshold = tree_.threshold[node]
                print(name + " ?")
                ans = input()
                ans = ans.lower()
                if ans == 'yes':
                    val = 1
                else:
                    val = 0
                if val <= threshold:
                    recurse(tree_.children_left[node], depth + 1)
                else:
                    symptoms_present.append(name)
                    recurse(tree_.children_right[node], depth + 1)
            else:
                present_disease = print_disease(tree_.value[node])
                print( "You may have " + present_disease )
                print()

```

```

        red_cols = dimensionality_reduction.columns
        symptoms_given =
red_cols[dimensionality_reduction.loc[present_disease].values[0].nonzero()]
        print("symptoms present " + str(list(symptoms_present)))
        print()
        print("symptoms given " + str(list(symptoms_given)) )
        print()
        confidence_level = (1.0*len(symptoms_present))/len(symptoms_given)
        print("confidence level is " + str(confidence_level))
        print()
        print('The model suggests:')
        print()
        row = doctors[doctors['disease'] == present_disease[0]]
        print('Consult ', str(row['name'].values))
        print()
        print('Visit ', str(row['link'].values))
        #print(present_disease[0])

```

```

recurse(0, 1)

```

```

tree_to_code(classifier,cols)

```

This section of code to be run after scraping the data

```

doc_dataset = pd.read_csv('doctors_dataset.csv', names = ['Name', 'Description'])

```

```

diseases = dimensionality_reduction.index
diseases = pd.DataFrame(diseases)

```

```

doctors = pd.DataFrame()
doctors['name'] = np.nan
doctors['link'] = np.nan
doctors['disease'] = np.nan

```

```

doctors['disease'] = diseases['prognosis']

```

```

doctors['name'] = doc_dataset['Name']
doctors['link'] = doc_dataset['Description']

```

```

record = doctors[doctors['disease'] == 'AIDS']
record['name']
record['link']

```

```
# Execute the bot and see it in Action
execute_bot()
```

- **Question_Diagnosis_Tkinter**

```
# Importing the libraries
from tkinter import *
from tkinter import messagebox
import os
import webbrowser
```

```
import numpy as np
import pandas as pd
```

```
class HyperlinkManager:
```

```
    def __init__(self, text):
        self.text = text
        self.text.tag_config("hyper", foreground="blue", underline=1)
        self.text.tag_bind("hyper", "<Enter>", self._enter)
        self.text.tag_bind("hyper", "<Leave>", self._leave)
        self.text.tag_bind("hyper", "<Button-1>", self._click)
```

```
        self.reset()
```

```
    def reset(self):
        self.links = {}
```

```
    def add(self, action):
        # add an action to the manager. returns tags to use in
        # associated text widget
        tag = "hyper-%d" % len(self.links)
        self.links[tag] = action
        return "hyper", tag
```

```
    def _enter(self, event):
        self.text.config(cursor="hand2")
```

```
    def _leave(self, event):
        self.text.config(cursor="")
```

```
    def _click(self, event):
        for tag in self.text.tag_names(CURRENT):
            if tag[:6] == "hyper-":
                self.links[tag]()
        return
```

```
# Importing the dataset
training_dataset = pd.read_csv('Training.csv')
test_dataset = pd.read_csv('Testing.csv')
```

```

# Slicing and Dicing the dataset to separate features from predictions
X = training_dataset.iloc[:, 0:132].values
Y = training_dataset.iloc[:, -1].values

# Dimensionality Reduction for removing redundancies
dimensionality_reduction =
training_dataset.groupby(training_dataset['prognosis']).max()

# Encoding String values to integer constants
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
y = labelencoder.fit_transform(Y)

# Splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 0)

# Implementing the Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)

# Saving the information of columns
cols = training_dataset.columns
cols = cols[:-1]

# Checking the Important features
importances = classifier.feature_importances_
indices = np.argsort(importances)[::-1]
features = cols

# Implementing the Visual Tree
from sklearn.tree import _tree

# Method to simulate the working of a Chatbot by extracting and formulating
questions
def print_disease(node):
    #print(node)
    node = node[0]
    #print(len(node))
    val = node.nonzero()
    #print(val)
    disease = labelencoder.inverse_transform(val[0])
    return disease
def recurse(node, depth):
    global val,ans
    global tree_,feature_name,symptoms_present
    indent = " " * depth
    if tree_.feature[node] != _tree.TREE_UNDEFINED:

```

```

name = feature_name[node]
threshold = tree_.threshold[node]
yield name + " ?"

#     ans = input()
ans = ans.lower()
if ans == 'yes':
    val = 1
else:
    val = 0
if val <= threshold:
    yield from recurse(tree_.children_left[node], depth + 1)
else:
    symptoms_present.append(name)
    yield from recurse(tree_.children_right[node], depth + 1)
else:
    strData=""
    present_disease = print_disease(tree_.value[node])
#     print( "You may have " + present_disease )
#     print()
    strData="You may have :"+ str(present_disease)

    QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')

    red_cols = dimensionality_reduction.columns
    symptoms_given =
red_cols[dimensionality_reduction.loc[present_disease].values[0].nonzero()]
#     print("symptoms present " + str(list(symptoms_present)))
#     print()
    strData="symptoms present: " + str(list(symptoms_present))
    QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
#     print("symptoms given " + str(list(symptoms_given)) )
#     print()
    strData="symptoms given: " + str(list(symptoms_given))
    QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
    confidence_level =
(1.0*len(symptoms_present))/len(symptoms_given)
#     print("confidence level is " + str(confidence_level))
#     print()
    strData="confidence level is: " + str(confidence_level)
    QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
#     print('The model suggests:')
#     print()
    strData='The model suggests:'
    QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
    row = doctors[doctors['disease'] == present_disease[0]]
#     print('Consult ', str(row['name'].values))
#     print()
    strData='Consult ' + str(row['name'].values)
    QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
#     print('Visit ', str(row['link'].values))

```

```

        #print(present_disease[0])
        hyperlink =
HyperlinkManager(QuestionDigonosis.objRef.txtDigonosis)
        strData='Visit ' + str(row['link'].values[0])
        def click1():
            webbrowser.open_new(str(row['link'].values[0]))
            QuestionDigonosis.objRef.txtDigonosis.insert(INSERT, strData,
hyperlink.add(click1))
            #QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
            yield strData

def tree_to_code(tree, feature_names):
    global tree_,feature_name,symptoms_present
    tree_ = tree.tree_
    #print(tree_)
    feature_name = [
        feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
        for i in tree_.feature
    ]
    #print("def tree({}):".format(", ".join(feature_names)))
    symptoms_present = []
    #    recurse(0, 1)

def execute_bot():
    #    print("Please reply with yes/Yes or no/No for the following symptoms")
    tree_to_code(classifier,cols)

# This section of code to be run after scraping the data

doc_dataset = pd.read_csv('doctors_dataset.csv', names = ['Name',
'Description'])

diseases = dimensionality_reduction.index
diseases = pd.DataFrame(diseases)

doctors = pd.DataFrame()
doctors['name'] = np.nan
doctors['link'] = np.nan
doctors['disease'] = np.nan

doctors['disease'] = diseases['prognosis']

doctors['name'] = doc_dataset['Name']
doctors['link'] = doc_dataset['Description']

record = doctors[doctors['disease'] == 'AIDS']

```

```
record['name']
record['link']
```

```
# Execute the bot and see it in Action
#execute_bot()
```

```
class QuestionDigonosis(Frame):
    objIter=None
    objRef=None
    def __init__(self,master=None):
        master.title("Question")
        # root.iconbitmap("")
        master.state("z")
#     master.minsize(700,350)
    QuestionDigonosis.objRef=self
    super().__init__(master=master)
    self["bg"]="light blue"
    self.createWidget()
    self.iterObj=None

    def createWidget(self):
        self.lblQuestion=Label(self,text="Question",width=12,bg="bisque")
        self.lblQuestion.grid(row=0,column=0,rowspan=4)

        self.lblDigonosis = Label(self, text="Digonosis",width=12,bg="bisque")
        self.lblDigonosis.grid(row=4, column=0,sticky="n",pady=5)

        # self.varQuestion=StringVar()
        self.txtQuestion = Text(self, width=100,height=4)
        self.txtQuestion.grid(row=0, column=1,rowspan=4,columnspan=20)

        self.varDiagonosis=StringVar()
        self.txtDigonosis =Text(self, width=100,height=14)
        self.txtDigonosis.grid(row=4,
column=1,columnspan=20,rowspan=20,pady=5)

        self.btnNo=Button(self,text="No",width=12,bg="bisque",
command=self.btnNo_Click)
        self.btnNo.grid(row=25,column=0)
        self.btnYes = Button(self, text="Yes",width=12,bg="bisque",
command=self.btnYes_Click)
        self.btnYes.grid(row=25, column=1,columnspan=20,sticky="e")

        self.btnClear = Button(self, text="Clear",width=12,bg="bisque",
command=self.btnClear_Click)
        self.btnClear.grid(row=27, column=0)
        self.btnStart = Button(self, text="Start",width=12,bg="bisque",
```

```

command=self.btnStart_Click)
    self.btnStart.grid(row=27, column=1,columnspan=20,sticky="e")
def btnNo_Click(self):
    global val,ans
    global val,ans
    ans='no'
    str1=QuestionDigonosis.objlter.__next__()
    self.txtQuestion.delete(0.0,END)
    self.txtQuestion.insert(END,str1+"\n")

def btnYes_Click(self):
    global val,ans
    ans='yes'
    self.txtDigonosis.delete(0.0,END)
    str1=QuestionDigonosis.objlter.__next__()

#    self.txtDigonosis.insert(END,str1+"\n")

def btnClear_Click(self):
    self.txtDigonosis.delete(0.0,END)
    self.txtQuestion.delete(0.0,END)
def btnStart_Click(self):
    execute_bot()
    self.txtDigonosis.delete(0.0,END)
    self.txtQuestion.delete(0.0,END)
    self.txtDigonosis.insert(END,"Please Click on Yes or No for the Above
symptoms in Question")
    QuestionDigonosis.objlter=recurse(0, 1)
    str1=QuestionDigonosis.objlter.__next__()
    self.txtQuestion.insert(END,str1+"\n")

class MainForm(Frame):
    main_Root = None
    def destroyPackWidget(self, parent):
        for e in parent.pack_slaves():
            e.destroy()
    def __init__(self, master=None):
        MainForm.main_Root = master
        super().__init__(master=master)
        master.geometry("300x250")
        master.title("Account Login")
        self.createWidget()
    def createWidget(self):
        self.lblMsg=Label(self, text="Health Care Chatbot", bg="PeachPuff2",
width="300", height="2", font=("Calibri", 13))
        self.lblMsg.pack()
        self.btnLogin=Button(self, text="Login", height="2", width="300", command
= self.lblLogin_Click)
        self.btnLogin.pack()
        self.btnRegister=Button(self, text="Register", height="2", width="300",

```

```

command = self.btnRegister_Click()
    self.btnRegister.pack()
    self.lblTeam=Label(self, text="", bg="slateblue4", width = "250", height =
"1", font=("Calibri", 13))
    self.lblTeam.pack()
    self.lblTeam1=Label(self, text="", bg="RoyalBlue1", width = "250", height =
"1", font=("Calibri", 13))
    self.lblTeam1.pack()
    self.lblTeam2=Label(self, text="", bg="RoyalBlue2", width = "250", height =
"1", font=("Calibri", 13))
    self.lblTeam2.pack()
    self.lblTeam3=Label(self, text="", bg="RoyalBlue3", width = "250", height =
"1", font=("Calibri", 13))
    self.lblTeam3.pack()

def lblLogin_Click(self):
    self.destroyPackWidget(MainForm.main_Root)
    frmLogin=Login(MainForm.main_Root)
    frmLogin.pack()
def btnRegister_Click(self):
    self.destroyPackWidget(MainForm.main_Root)
    frmSignUp = SignUp(MainForm.main_Root)
    frmSignUp.pack()

```

```

class Login(Frame):
    main_Root=None
    def destroyPackWidget(self,parent):
        for e in parent.pack_slaves():
            e.destroy()
    def __init__(self, master=None):
        Login.main_Root=master
        super().__init__(master=master)
        master.title("Login")
        master.geometry("300x250")
        self.createWidget()
    def createWidget(self):
        self.lblMsg=Label(self, text="Please enter details below to login",bg="blue")
        self.lblMsg.pack()
        self.username=Label(self, text="Username * ")
        self.username.pack()
        self.username_verify = StringVar()
        self.username_login_entry = Entry(self, textvariable=self.username_verify)
        self.username_login_entry.pack()
        self.password=Label(self, text="Password * ")
        self.password.pack()
        self.password_verify = StringVar()
        self.password_login_entry = Entry(self, textvariable=self.password_verify,
show= '*')

```

```

        self.password_login_entry.pack()
        self.btnLogin=Button(self,      text="Login",      width=10,      height=1,
command=self.btnLogin_Click)
        self.btnLogin.pack()
    def btnLogin_Click(self):
        username1 = self.username_login_entry.get()
        password1 = self.password_login_entry.get()

#        messagebox.showinfo("Failure", self.username1+"."+password1)
    list_of_files = os.listdir()
    if username1 in list_of_files:
        file1 = open(username1, "r")
        verify = file1.read().splitlines()
        if password1 in verify:
            messagebox.showinfo("Sucess","Login Sucessful")
            self.destroyPackWidget(Login.main_Root)
            frmQuestion = QuestionDigonosis(Login.main_Root)
            frmQuestion.pack()
        else:
            messagebox.showinfo("Failure", "Login Details are wrong try again")
    else:
        messagebox.showinfo("Failure", "User not found try from another user\n
or sign up for new user")

```

```

class SignUp(Frame):
    main_Root=None
    print("SignUp Class")
    def destroyPackWidget(self,parent):
        for e in parent.pack_slaves():
            e.destroy()
    def __init__(self, master=None):
        SignUp.main_Root=master
        master.title("Register")
        super().__init__(master=master)
        master.title("Register")
        master.geometry("300x250")
        self.createWidget()
    def createWidget(self):
        self.lblMsg=Label(self, text="Please enter details below", bg="blue")
        self.lblMsg.pack()
        self.username_label = Label(self, text="Username * ")
        self.username_label.pack()
        self.username = StringVar()
        self.username_entry = Entry(self, textvariable=self.username)
        self.username_entry.pack()

        self.password_label = Label(self, text="Password * ")
        self.password_label.pack()
        self.password = StringVar()
        self.password_entry = Entry(self, textvariable=self.password, show='*')

```

```

        self.password_entry.pack()
        self.btnRegister=Button(self, text="Register", width=10, height=1,
bg="blue", command=self.register_user)
        self.btnRegister.pack()

```

```

def register_user(self):
    file = open(self.username_entry.get(), "w")
    file.write(self.username_entry.get() + "\n")
    file.write(self.password_entry.get())
    file.close()

    self.destroyPackWidget(SignUp.main_Root)

    self.lblSucess=Label(root, text="Registration Success", fg="green",
font=("calibri", 11))
    self.lblSucess.pack()

    self.btnSucess=Button(root, text="Click Here to proceed",
command=self.btnSucess_Click)
    self.btnSucess.pack()
    def btnSucess_Click(self):

        self.destroyPackWidget(SignUp.main_Root)
        frmQuestion = QuestionDigonosis(SignUp.main_Root)

        frmQuestion.pack()

```

```

root = Tk()

```

```

frmMainForm=MainForm(root)frmMainForm.pack() root.mainloop()

```

B. RESEARCH PAPER

The screenshot shows the Microsoft CMT Author Console interface. At the top, there's a navigation bar with 'Submissions' selected. Below it, the 'Author Console' section displays a table of submissions. The first submission is for Paper ID 118, titled 'Therapist Chatbot using Embedding and Long-Short-Term-Memory in NLP'. The submission files include 'springer research paper.docx'. The actions column shows options to 'Edit Submission', 'Edit Conflicts', and 'Delete Submission'. The bottom of the page features a footer with copyright information and a Windows taskbar at the very bottom.

Paper ID	Title	Files	Actions
118	Therapist Chatbot using Embedding and Long-Short-Term-Memory in NLP Show abstract	Submission files: springer research paper.docx	Submission: Edit Submission Edit Conflicts Delete Submission

The screenshot shows a Gmail inbox with an email from Microsoft CMT. The email subject is 'International Conference on Power Engineering and Intelligent System : Submission (118) has been created.' The email body contains the following information:

Hello,

The following submission has been created.

Track Name: PEIS2023

Paper ID: 118

Paper Title: Therapist Chatbot using Embedding and Long-Short-Term-Memory in NLP

Abstract:
The need for preventative healthcare has been well recognized, but not everyone has access to routine medical consultations because of a slight decrease in the number of professionals giving these services. While many reports never having visited a doctor, statistics reveal that the majority of individuals already have some form of heart or lung disease. The vast majority of people do not have the background in medicine to properly diagnose or evaluate their health problems. There is an immediate need for NLP tools in the healthcare sector. In order to better assist their human counterparts, these chatbots collect data about their users' health and use that data to tailor answers and recommendations on how to address their specific medical concerns. Assisting patients and pointing them in the direction of the most appropriate assistance are two of the many ways in which healthcare chatbots with AI capabilities might be valuable to the medical industry.

Created on: Fri, 31 Mar 2023 14:50:30 GMT

Last Modified: Fri, 31 Mar 2023 14:50:30 GMT

Authors:
- [anjaliscrinivas11@gmail.com](#) (Primary)

