

Dayananda Sagar College of Engineering, Bangalore

An Institution Affiliated to
Visvesvaraya Technological University (VTU), Belgaum



Therabot Your Pocket Therapist

Thesis submitted in partial fulfillment of the curriculum prescribed
for the award of the degree of Bachelor of Engineering in
Computer Science & Engineering by

Submitted by

1DS15CS417 Rajan Rajan A.
1DS14CS118 Varsha Jambunathan
1DS14CS136 Laasya R.

Under the Guidance of

Dr. Ramesh Babu D. R.

Vice-Principal & HoD
Department of CSE,

Dayananda Sagar College of Engineering, Bangalore



Department of Computer Science & Engineering
June, 2018

Dayananda Sagar College of Engineering, Bangalore

An Institution Affiliated to
Visvesvaraya Technological University (VTU), Belgaum



Certificate

This is to certify that the work entitled "**Therabot**" is a bonafide work carried out by Raja Rajan A., Varsha Jambunathan and Laasya R. in partial fulfillment of the award of the degree of Bachelor of Engineering in Computer Science & Engineering of Visvesvaraya Technological University, Belgaum, during the year 2018. It is certified that all corrections / suggestions indicated during continuous internal assesments have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Guide

Dr. Ramesh Babu D. R.

Vice-Principal & HoD
Department of CSE,
DSCE, Bangalore

Head of the Department

Dr. Ramesh Babu D. R.

Vice-Principal & HoD
Department of CSE,
DSCE, Bangalore

Examiners: 1. _____

2. _____

Date:

Place: Bangalore

3. _____

Abstract

This study aims to highlight the use cases of a variety of technologies in the field of Artificial Intelligence, Speech Synthesis, Personalization, User Character Segmentation and Automated Text/Speech-based Chatbots. To illustrate the uses, we have chosen a specific field of expertise, mental wellness.

We have built an artificially chatbot that is designed to understand the complexities of it's user, help the user feel more self-aware and spontaneously check back on the user from time to time. We have also devised a method to extract key likes and dislikes from the user's messages using NLP techniques.

The goal of Therabot as a product to the consumer is not to overthrow the need for real life therapists, but to simply allow the user to feel more comfortable talking about his/her experiences, thus decreasing the stigma that perpetuates around mental health in our country and around the world. We want to make mental wellness more accessible and educate users on the true meaning of depression and anxiety.

The objective of our research is to show how chatbots powered by artificial intelligence can be not only used for service based QA systems, but also in unique niche markets such as medical health, distress systems, education etc. Using our unique methods of generating the user's responses and creating a schema for each individual user, we are able to bring more personalization and unique feeling to every user.

Acknowledgement

The satisfaction that accompanies the successful completion of our project would be incomplete without mentioning the people who made it possible, whose constant guidance and encouragement crowns all the efforts with success.

We would greatly mention the enthusiastic influence for provided by Dr. Krishnan R., Professor, Department of Computer of Science and Engineering of Dayananda Sagar College of Engineering for his ideas and co-operation during our venture and making this project a great success.

We are thankful to Dr. D. R. Ramesh Babu, Vice-Principal & HoD of Department of Computer Science and Engineering of Dayananda Sagar College of Engineering for his co-operation during all phases of our project.

We are also thankful to Dr. C. P. S. Prakash, Principal of Dayananda Sagar College of Engineering, Bangalore for being kind enough to provide us an opportunity to work on a project in this institution.

It is a pleasure to thank the friendly co-operation showed by all the staff members of the Computer Science Department, Dayananda Sagar College of Engineering, and finally, not to forget our friends & family who have supported us through our academic career.

Raja Rajan A.
Varsha Jambunathan
Laasya R.

Table of Contents

Table of Contents	iii
List of Figures	iv
1 Introduction	1
1.1 The State of Mental Health	2
1.2 The Cognitive Behavioural Therapy	3
1.3 The Rise of Chatbots	4
1.4 The Abstract of Artificial Intelligence	6
2 Problem Statement and Proposed Solution	8
2.1 Problem Statement	9
2.2 Existing State of the Art	10
2.3 Proposed Solution	11
2.4 System Requirements	13
3 Literature Survey	14
3.1 Base Papers	15
3.2 Additional Papers found through in-depth research	16
4 Architecture and Design	18
4.1 System Overview	19
4.2 Software Architecture	20
4.3 Sentiment Analysis Anatomy	23
4.4 Word2Vec (NLP) Structure	24
5 Implementation	26
5.1 Hardware Applications	27
5.2 Software Applications	27
5.3 Procedure	28
5.4 Algorithms	32
5.5 Diagrams	37

6 Experimentation & Results	39
6.1 Tensorflow	40
6.2 Screenshots	49
6.3 Testing	58
7 Conclusion	59
7.1 Conclusive Statement	60
8 Appendix: Code	61
8.1 Website	62
8.2 Chatbot	89
8.3 Machine Learning	98
References	107

List of Figures

1.1 Roots of Mental Health Issues	2
1.2 Sequence of Cognitive Behavioural Therapy	3
1.3 Usecases of Chatbots	5
1.4 Dumb Chatbots	5
1.5 Overview of Artificial Intelligence	6
2.1 Comparitive Study of Competitors	10
4.1 End-to-End System Block Diagram	20
4.2 Inventive Steps	21
4.3 Conversational Trees of the Chatbot	22
4.4 Steps of the VADER Model	23
4.5 Syntax Parsing and Breakdown to Parts of Speech	24
4.6 Sentiment Analysis Response	25
5.1 Project File Structure	28
5.2 Therabot Questionnaire	29
5.3 Jupyter Notebook - Tokenizing	31
5.4 Jupyter Notebook - Stemming	31

5.5	Gradient Descent with Different Learning Rates	33
5.6	Typical Logistic Regression Model Plot	35
5.7	Softmax Function	36
5.8	Data Flow Diagram	37
5.9	Sequence Diagram - Onboarding Phase	38
5.10	Sequence Diagram - Regular Conversations	38
6.1	Jupyter Notebook - Sample Response	45
6.2	Jupyter Notebook - Number of Epochs = 500	45
6.3	Jupyter Notebook - Number of Epochs = 1000	45
6.4	Jupyter Notebook - Number of Epochs = 1500	46
6.5	Jupyter Notebook - Number of Epochs = 2000	46
6.6	Jupyter Notebook - Batch Size = 4	47
6.7	Jupyter Notebook - Batch Size = 8	47
6.8	Jupyter Notebook - Batch Size = 16	47
6.9	Jupyter Notebook - Batch Size = 24	47
6.10	Jupyter Notebook - Batch Size = 36	48
6.11	Introduction	49
6.12	Cognitive Behavioral Therapy	50
6.13	Welcome - User Dashboard	50
6.14	Modules on the Website	51
6.15	Module #1 - Dear Diary	51
6.16	Module #2 - Photo Album	52
6.17	Module #3 - Emote your Feelings	52
6.18	Facebook Messenger	53
6.19	User Onboarding	53
6.20	Record Journal	54
6.21	Saving Journal Entry	54
6.22	Meditation	55
6.23	Google Assistant	55
6.24	Meditation w/ SSML	56
6.25	PHQ9 Test	56
6.26	Tablet Screenshot	57
6.27	Desktop Screenshot	57
6.28	Chatbot Testcases	58

Chapter 1

Introduction

1.1 The State of Mental Health

Burden of mental disorders had risen over last few decades. Mental health is a state of well-being in which the individual realizes his or her own abilities, can cope with the normal stresses of life, can work productively and is able to make a contribution to his or her community. WHO estimated that globally over 450 million people suffer from mental disorders. Currently mental and behavioural disorders account for about 12 percent of the global burden of diseases. Major proportions of mental disorders come from low and middle income countries.

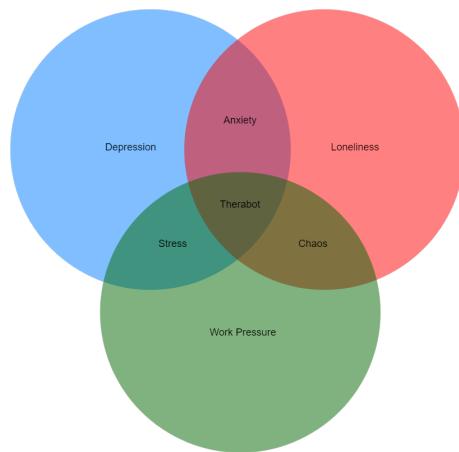


Figure 1.1: Roots of Mental Health Issues

Progress in mental health service delivery has been slow in most low- and middle-income countries. Barriers include the existing public-health priorities and its influence on funding; challenges to delivery of mental health care in primary-care settings; the low numbers of those trained in mental health care; and the lack of mental health perspective in public-health leadership. There have been numerous calls for invoking political will, for enhancing advocacy and for galvanizing community participation; all with scant improvement in outcomes.

Thus, it becomes now opportune to explore the paradigm of mental health awareness as a means of combating stigma, enhancing prevention, ensuring early recognition, and also stimulating simple and practical interventions within the community. Today there are opportunities in terms of growing acknowledgement of mental disorders as key targets of global health action, as well as of leveraging new technologies particularly internet, big data and cell phones in amplifying simple field interventions found successful in primary care and other echelons.

1.2 The Cognitive Behavioural Therapy

Cognitive behavioral therapy (CBT) is a short-term therapy technique used by counselors and therapists to teach individuals to change their unwanted behaviors by changing their thought patterns. The premise of cognitive behavioral therapy is that our thought patterns (cognition) and interpretations of life events greatly influence how we behave and, ultimately, how we feel.

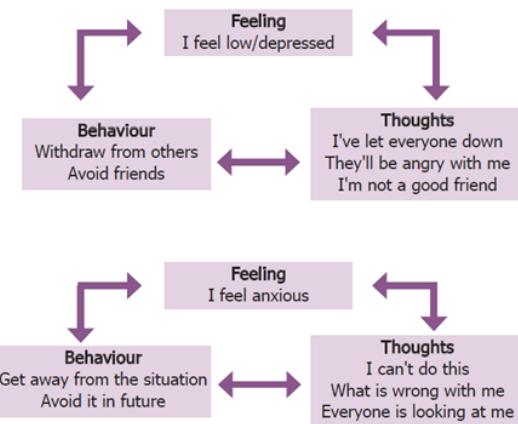


Figure 1.2: Sequence of Cognitive Behavioural Therapy

CBT is a form of psychotherapy that focuses on how your thoughts, beliefs and attitudes affect your feelings and behavior. It aims to teach you effective coping strategies for dealing with different problems throughout life. CBT can help you make sense of overwhelming problems by breaking them down into smaller parts.

One of the key tenets of CBT is that distorted thinking leads to distress and problematic behaviors, whereas thinking realistically with less negativity allows individuals to respond to challenging life circumstances in an effective way. Research shows this technique is an effective therapy for not only depression and panic disorder, but many illnesses and dysfunctional behaviors.

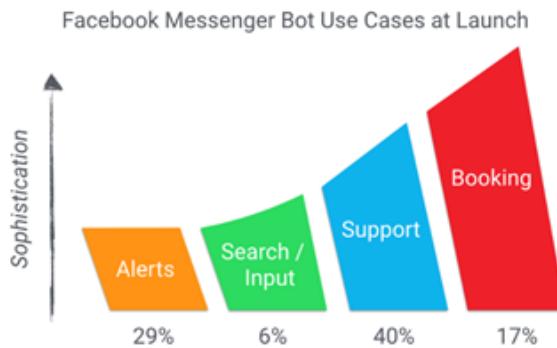
1.3 The Rise of Chatbots

A chatbot is one of the simple ways to transport data from a computer without having to think for proper keywords to look up in a search or browse several web pages to collect information; users can easily type their query in natural language and retrieve information. In this paper, information about the design, implementation of the chatbot has been presented.

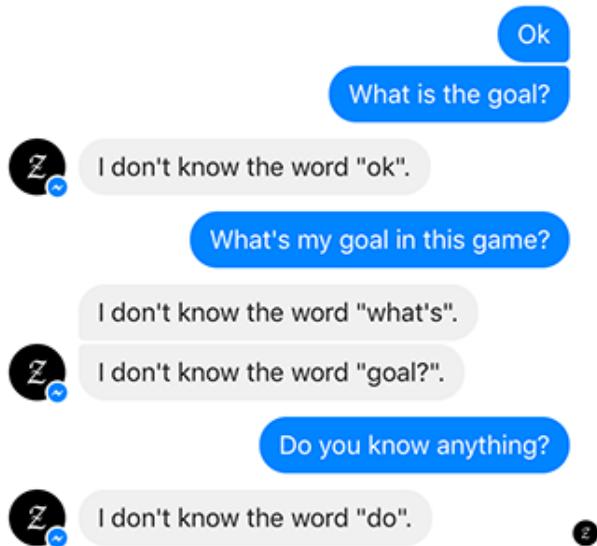
From the survey above, it can be said that the development and improvement of chatbot design grow at an unpredictable rate due to variety of methods and approaches used to design a chatbot. Chatbot is a great tool for quick interaction with the user. They help us by providing entertainment, saving time and answering the questions that are hard to find. The Chatbot must be simple and conversational. Since there are many designs and approaches for creating a chatbot, it can be at odds with commercial considerations. Researchers need to interact and must agree on a common approach for designing a Chatbot.

In this project, we looked into how chatbots are developed and the applications of chatbots in various fields. In addition, comparison has been made with other chatbots in the same field of interest. A general purpose chatbot must be simple, user friendly, must be easily understood and the knowledge base must be compact. Although some of the commercial products have recently emerged, improvements must be made to find a common approach for designing a chatbot.

Chatbots have seen a huge rise in the recent markets and have primarily been used in the fields of service, question and answering and quite recently, home automation. They have proven to be very useful in the field of automation and have successfully replaced menial jobs that could've cost a lot of money for major corporations. Current commercial chatbots are capable of understanding simple sentences through pattern matching techniques, and search for questions asked in a repository of answers. This works out well in a closed scenario where the kind of questions the user may ask are limited, but not in a real life scenario.

**Figure 1.3:** Usecases of Chatbots

Chatbots were later trained on neural networks to become much smarter, using a larger dataset. They were trained to learn on a wider spectrum and can understand sentences in a more humane form due to the advancements in NLP. But with all the new things that chatbots now have to learn, they've never really been personal to an individual.

**Figure 1.4:** Dumb Chatbots

The objective of our method is to create a personalized user model that is unique to every user but keeps them anonymous at the same time. This helps the chatbot response generator understand who the user is and push relatable content and context when necessary.

1.4 The Abstract of Artificial Intelligence

Artificial Intelligence (AI) is defined as intelligence exhibited by an artificial entity to solve complex problems and such a system is generally assumed to be a computer or machine. Artificial Intelligence is an integration of computer science and physiology. Intelligence in simple language is the computational part of the ability to achieve goals in the world. Intelligence is the ability to think to imagine creating memorizing and understanding, recognizing patterns, making choices adapting to change and learn from experience.

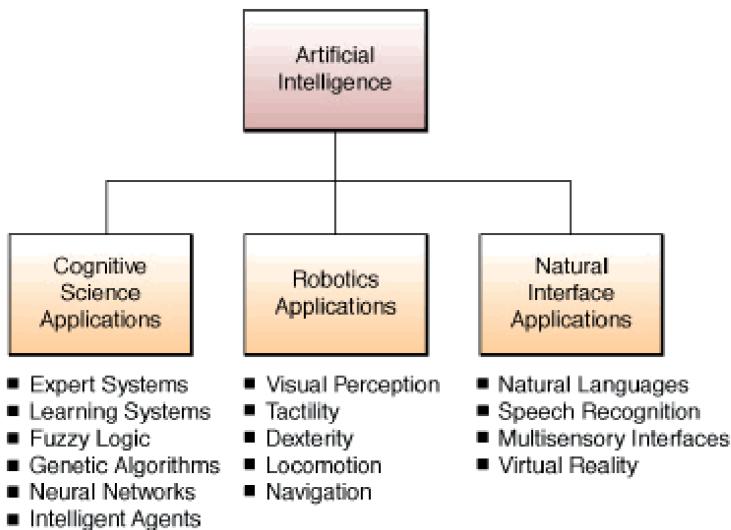


Figure 1.5: Overview of Artificial Intelligence

Artificial Intelligence was created with the sole aim of mimicking or even outperforming human minds. Thus it is very important we question the fact whether it has actually been able to do so.

It cannot be ignored that the fact of AI is being used all around us especially in the fields of medicine, robotics, law, stock trading etc. It is being used in homes and big establishments. such as military bases and the NASA space station. NASA has sent out artificially intelligent robots to planets so as to learn more about their habitat and atmosphere, with the intention of investigating if there is a possibility of humans living on these planets.

Expert systems have been used by Mercedes Benz and other auto manufacturers in the design of vehicle components, subway systems in Washington, D.C. use expert system software controllers to cause subway trains to stop within 3 inches of the right spot on the platform. These trains have motormen primarily to reassure passengers.

AI has filtered into general applications in these fields and has become so common that it is not referred to as Artificial Intelligence anymore.

Blind supporters of AI would point to the time when AI Deep Blue II defeated chess master Garry Kasparov to prove that Artificial Intelligence can in fact be smarter than humans. Though there is no doubt that the AI Deep Blue II won that game, it is still probably one of the dumbest software alive. The operators were programming the AI in every round depending on the opposition's last move. Also, the Deep Blue II had studied all of Kasparov's previous games while the latter wasn't given the same benefit. One can safely say that even though the Deep Blue II AI defeated Kasparov, it was never a fair fight to begin with. Latest technologies like Xbox 360's Kinect and iPhone's Siri use algorithms based on Artificial Intelligence, but it is a well-known fact that these technologies are a long way from being perfect.

Thus we can safely conclude that though Artificial Intelligence has made a lot of progress in the past few decades, it is not at a level where one can confidently state that it is now ready to completely replace the human mind. That being said, large scale research is now being conducted into the field of proper simulation of the human brain.

Chapter 2

Problem Statement and Proposed Solution

2.1 Problem Statement

Narrative therapy is embracing into the second decade on the arena of psychotherapy. Yet, how can these therapeutic qualities best be inferred? How persuasive are the arguments relating to its effectiveness? What type of work is presides over with patients that portray that such practices are indeed beneficial to people? And lastly, how solid is the narrative-inspired research(often called co-research) that forge the use of externalizing practices? A distinctive quality of the therapeutic stance, if narrative therapy lies in the manner in which it pledges from the postmodern philosophical tradition. Similarly, narrative-based research pop up to be unique from other genre of research in mental health professions.

Conventional forms of research generally search for empirical conclusions (as with quantitative designs) and consistently rely on the researcher's expert role to collect, codify, and interpret data (as is the case with many forms of qualitative methodology). Narrative-inspired research, conversely, limelight's on the subjective nature of experience and inquire to de-emphasize the therapist's role as an expert.

The relevant elements necessary for a basic psychotherapy reading, is:

- What the main problem is (for example depression or anxiety).
- What triggers it (what makes the patient feel that way on a day to day basis).
- Examples of patient's symptoms, changes in patients behaviours and thinking.
- What consequence the problem has on patient's life.

An example with all the relevant elements needed, is: "My main problem is feeling depressed most of the day every day. I feel tired, my Appetite is affected and I struggle to concentrate. I have stopped seeing friends, goingto work or doing things around the home and I am spending more time in bed. I have thoughts that I am letting my family down and that I can't be bothered. As a Consequence, I am isolated and being off sick means I have used up most of my savings and money is tight".

2.2 Existing State of the Art

Features	Therabot	Woebot	Ruuh	ELIZA	Psychiatrist
Designed By	Students of Dayananda Sagar College of Engineering	Scientists	Microsoft	Scientists	God, and a costly medical degree
Release Month and Year	March 2018	October 2017	March 2017	February 1966	Since the beginning of mankind
Focused on Mental Health	✓	✓		✓	✓
Uses Machine Learning	✓	✓	✓		
Machine Learning Algorithms	Ensemble ML (NLP, KNN, DTC, RF, NB)	NLP, Decision Tree Classifier	NLP, Artificial Intelligence		
Cross Platform	✓	✓			
Cost per Session	FREE	FREE	FREE	FREE	\$100 to \$500 per hour
Security and Data Encryption	✓		✓		✓
Human Like Interaction	✓		✓		✓ (duh!)
Stress Reduction Methods	Cognitive Behaviour Therapy, Yogic Breathing, Journalling, Personality Shaping	Cognitive Behavior Therapy, Journalling	Small Talk Conversation	Increases Stress	Empties Wallet & Increases Stress

Figure 2.1: Comparative Study of Competitors

Currently, people who do feel sad have only a handful of options available to them, including talking to friends, talking to family or seeing a therapist. The two are subjective, and doesn't really apply when someone is clinically depressed and not willing to talk it out to his/her nearest and dearest.

Then, there are those who are hesitant to talk to strangers as well, considering they have the time and money to book an appointment with a therapist. Even if they do so, they have trouble opening up to someone who they have never met before.

When it comes to the existing Chatbot ecosystem, it has boomed in the market just a few days ago and has been dominant amongst companies to work as a front for their customer services portals, replacing everyday queries etc. We wanted to extend this functionality into something even more helpful and personal.

(why therapy, why chatbots) = (how good have they been : they trick the user into thinking that they are communicating with an actual person) chatbot which automatically gives immediate responses to the users based on the data set of Frequently Answered Questions (FAQs), using Artificial Intelligence Markup Language (AIML) and Latent Semantic Analysis (LSA). Template based questions like greetings and general questions will be answered using AIML and other service related questions use LSA to give responses.

2.3 Proposed Solution

The purpose of this study is to take an in-depth critical look into the narrative practice of externalizing; focusing intimately on its various uses and purported therapeutic qualities. The oneness of this deliberation is in the singular idiosyncrasy in which it locus on the various shot in which externalizing praxis are employed within narrative therapy and the leeway to which they are evaluated through research.

Narrative therapy proffer vast innovative essence and therapeutic practices. Key betwixt them is the praxis of engaging people in externalizing gab. Intermittently viewed as matter-of-factly a therapy technique, externalizing practices address wider inference about how clinicians view the world, understand clients quandary, and by association, how clients view their self and their potentiality to make turnover in their lives.

An evolving frame of narrative-inspired literature and research is being ordain that may assist in repose narrative therapy alongside other catalog and researched therapeutic entrance. An nonpartisan of this study is to further that literary process, concretely as it relates to the practice of externalizing. This study's oneness lies in the comportment in which it targets solely on the issues and therapeutic entanglement surrounding externalizing practices. By doing this, it represents novel work in the field of narrative therapy literature, and will assist in explaining a pivotal narrative practice to the preponderant clinical psychology community.

The intent of this abstraction is to take an in-depth critical look into the narrative practice of externalizing; focusing intimately on its diversified uses and purported therapeutic qualities.

A considerable measure of literature abide on the multiple distinctive ways in which externalizing practices function. Much of the literature is sprinkled with various tantalizing therapeutic qualities ascribing to externalizing practices, often not beyond the context of how externalizing serves other narrative practices; such as deconstructive listening, reliant influence questions, discovering unique sequel, etc.

Accordingly, the oneness of this study is in the singular manner in which it locates on the numerous ways in which externalizing practices are employed within narrative therapy and the leeway to which they are evaluated through research.

Narrative therapy affirms many innovative ideas and therapeutic conveniences. Key among them is the process of engaging people in externalizing interactions. Consistently it is viewed as simply, a therapy technique, externalizing practices address wider implications about how clinicians picture the world, understand clients' problems, and by association, how clients view themselves and their ability to make a new revision to their lives.

While narrative-inspired texts discept externalizing practices to varying degrees, this study is distinguished by the manner in which it locates singularly on the use of and therapeutic value concord with externalizing practices. By doing so, this study provides a richer forbearing of the theory and convention of externalizing approaches to clinicians versed in narrative work, as well as to those new to the narrative advent. It also intends an opportunity to explore in-depth the sequence of research behind narrative therapy and externalizing practices.

2.3.1 Unique Features

- Build a simple and interactive real time chat system.
- Extraction of preferences and interests of the user.
- Dedicated system which is able be a bolster of support and understanding.
- Designed in such a way that it should work in cross platform devices.
- Effective hybrid models built on ensemble methods for ready prediction.
- Can be easily integrated and upgradable.

2.4 System Requirements

- Android/iOS Mobile Device to test/run the chatbot
- Facebook Messenger or Google Assistant
- Adequately fast Internet connection to access
 - Dialogflow
 - Actions on Google Dashboard
 - Firebase Firestore Database
 - Google Cloud Platform Dashbaord
- Node and NPM to run the webhook server
- Python 3.6+ to run the NLP and Analytics module
- iPython Notebook to test/run the scripts
- Flask to run the API server to access Python scripts

Chapter 3

Literature Survey

3.1 Base Papers

3.1.1 Engineering Base Paper

Psychology Predictive Model Research based on Artificial Neural Network

Author: Cai Zhongxi

Understanding the psychological state of mind in college students has become crucial due to the increasing mental health issues. The scenario is such that students succumb to emotional stress, learning pressure, relationship issues and inability to express thoughts, and are trapped in depression. This paper focuses on analyzing the incidents that affect the mental status of students by predicting the correlation between the implicating factors. The neural network is trained using a multilayer perceptron to obtain a back propagated output which needs optimization. The authors have proposed a mathematical model for the use of genetic algorithms to improve the mutation rate of the result, thereby, increasing the learning rate.

3.1.2 Medical Base Paper

Delivering Cognitive Behavior Therapy to Young Adults With Symptoms of Depression and Anxiety Using a Fully Automated Conversational Agent (Woebot):A Randomized Controlled Trial *Authors: Kathleen Kara Fitzpatrick, Alison Darcy and Molly Vierhile*

This paper outlines the role of Cognitive Behavior Therapy (CBT) in the treatment of individuals facing anxiety, depression, obsessive compulsive disorder (OCD), post-traumatic stress disorder (PTSD) and anger problems. The authors have created a chatterbot, Woebot, whose sole purpose is to talk to the user and get inputs about his mood. Patient Health Questionnaire-9 and Generalized Anxiety Disorder-7 are some of the standardized questionnaires that every therapist uses to determine the depression severity and frequency. Woebot uses these measures to better understand the individual.

3.2 Additional Papers found through in-depth research

Content-Oriented User Modeling for Personalized Response Ranking in Chatbots *Authors:* *Bingquan Liu , Zhen Xu , Chengjie Sun, Baoxun Wang , Xiaolong Wang, Derek F. Wong , Senior Member and Min Zhang*

Automatic chatbots (also known as chat-agents) have attracted much attention from both researching and industrial fields. Generally, the semantic relevance between users' queries and the corresponding responses is considered as the essential element for conversation modeling in both generation and ranking based chat systems. This paper aims to address the personalized response ranking task by incorporating user profiles into the conversation model.

On the Construction of more Human-like Chatbots: Affect and Emotion Analysis of Movie Dialogue Data *Author:* *Rafael E. Banchs*

Affect and emotion are inherent properties of human-human communication and interaction. Recent research interest in chatbots and conversational agents aims at making human-machine interaction more human-like in both behavioral and attitudinal terms. This paper intends to present some baby steps in this direction by analyzing a large dialogue dataset in terms of tonal, affective and emotional bias, with the objective of providing a valuable resource for developing and training data-driven conversational agents with discriminative power across such dimensions.

NLAST: A natural language assistant for students *Authors:* *Fernando A. Mikic Fonte, Martín Llamas Nistal, Juan C. Burguillo Rial, and Manuel Caeiro Rodríguez*

This paper presents a system that works as an assistant for students in their learning process. The assistant system has two main parts: an Android application and a server platform. The Android application is a chatterbot (i.e., an agent intended to conduct a conversation in natural language with a human being) based on AIML, one of the more successful languages for developing conversational agents. The chatterbot acts as an intermediation agent between a student and the server platform.

Sentiment Analysis of Text using Deep Convolution Neural Networks

Authors: Anmol Chachra, Pukkit Mehndiratta and Mohit Gupta

Sentiment analysis has been one of the most researched topics in Machine learning. The roots of sentiment analysis are in studies on public opinion analysis at the start of 20th century, but the outbreak of computer-based sentiment analysis only occurred with the availability of subjective text in Web. The task of generating effective sentence model that captures both syntactic and semantic relations has been the primary goal to make better sentiment analyzers. In this paper, we harness the power of deep convolutional neural networks (DCNN) to model sentences and perform sentiment analysis.

Big data in psychology: using word embeddings to study theory-of-mind

Authors: Michel Généreux, Bryor Snelfella and Marta Maslej

This paper uses a computational approach to estimate the concreteness of words. After examining its reliability and validity, we apply this approach to study a claim within Psychology, namely, that reading a literary fiction story improves the ability to attribute mental states to others.

Automated Text Messaging as an Adjunct to Cognitive Behavioral

Therapy for Depression:A Clinical Trial

Authors: Adrian Aguilera, Emma Bruehlman-Senecal, Orianna Demasi and Patricia Avila

Cognitive Behavioral Therapy (CBT) for depression is efficacious, but effectiveness is limited when implemented in low-income settings due to engagement difficulties including nonadherence with skill-building homework and early discontinuation of treatment. Automated messaging can be used in clinical settings to increase dosage of depression treatment and encourage sustained engagement with psychotherapy.

Weighted Word2Vec Based on the Distance of Words

Authors: Chia-Yang Chang, Shie-Jue Lee and Chih-Chin Lai

Word2vec is a novel technique for the study and application of natural language processing(NLP). It trains a word embedding neural network model with a large training corpus. After the model is trained, each word is represented by a vector in the specified vector space. The vectors obtained possess many interesting and useful characteristics that are implicitly embedded with the original words. The idea of word2vec is that there are relations between the words if they appear in the neighborhood.

Chapter 4

Architecture and Design

4.1 System Overview

The system is designed in such a way that it contains three main modules:

- User Interface
- Natural Language Processing
- Sentiment & Keyword Extraction

4.1.1 User Interface

We've integrated the Facebook Messenger and Google Assistant interface to be able to get into an array of devices right away instead of marketing to install a new application. The benefits to this is that, we would not take up any extra space on the user's phone and also that there are zero to none chances of being hacked or reverse engineered since everything is on the server side and not on the client side which is under user control.

4.1.2 Natural Language Processing

This happens in realtime as we are conversing with the chatbot. The sentences being sent to the NLP module are broken down and tokenized into various parts of speech tags. These help the model understand what the significance of each word means. We will then attempt to match the key word to one of the intents in our database through means of similarity. If there exists no such match, we will simply call the Default Fallback Intent and display to the user that it doesn't know what to do.

4.1.3 Sentiment & Keyword Extraction

In this phase, we attempt to summarize the journals written by the user and extract the sentiment of the message. This will explain a lot about the user's intent and interests, not to mention the dislikes. This information can be stored for each individual user and later used in casual conversations.

4.2 Software Architecture

First, architecture defines software elements. The architecture embodies information about how the elements relate to each other. This means that it specifically omits certain information about elements that does not pertain to their interaction. Thus, an architecture is foremost an abstraction of a system that suppresses details of elements that do not affect how they use, are used by, relate to, or interact with other elements. In nearly all modern systems, elements interact with each other by means of interfaces that partition details about an element into public and private parts. Architecture is concerned with the public side of this division; private details—those having to do solely with internal implementation—are not architectural.

4.2.1 System Block Diagram

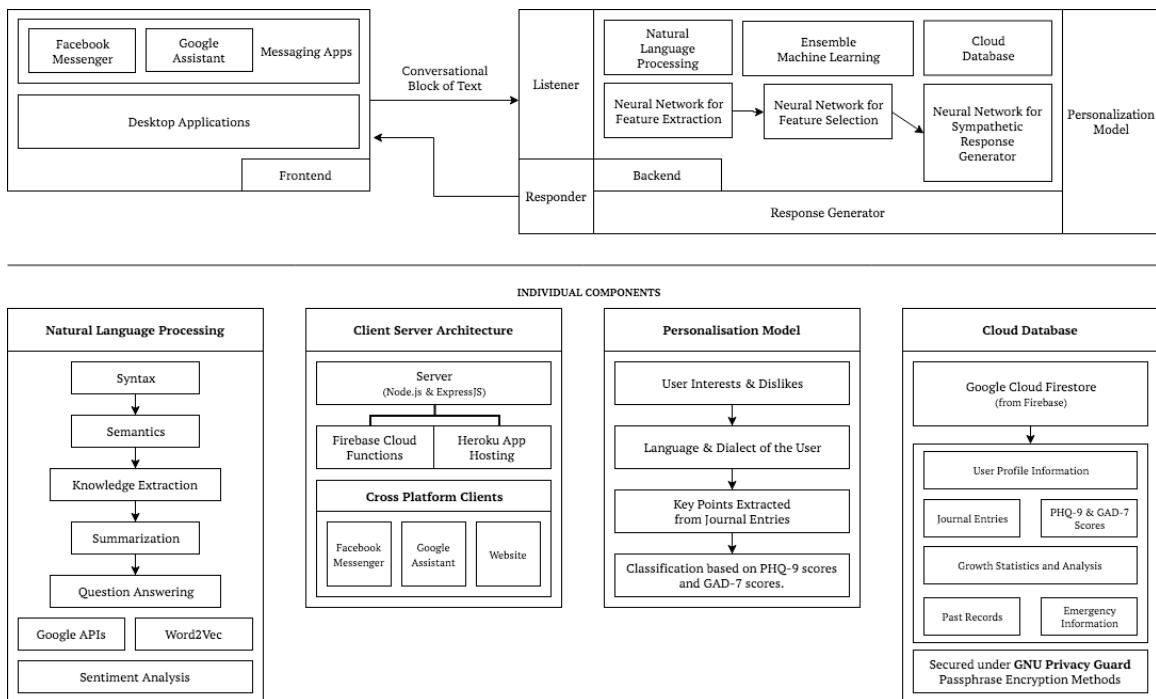


Figure 4.1: End-to-End System Block Diagram

Second, the definition makes clear that systems can and do comprise more than one structure and that no one structure can irrefutably claim to be the architecture. For example, all nontrivial projects are partitioned into implementation units; these units are given specific responsibilities and are frequently the basis of work assignments for programming teams. This type of element comprises programs and data that software in other implementation units can call or access, and programs and data that are private. In large projects, these elements are almost certainly subdivided for assignment to subteams. This is one kind of structure often used to describe a system. It is very static in that it focuses on the way the system's functionality is divided up and assigned to implementation teams.

4.2.2 Inventive Steps

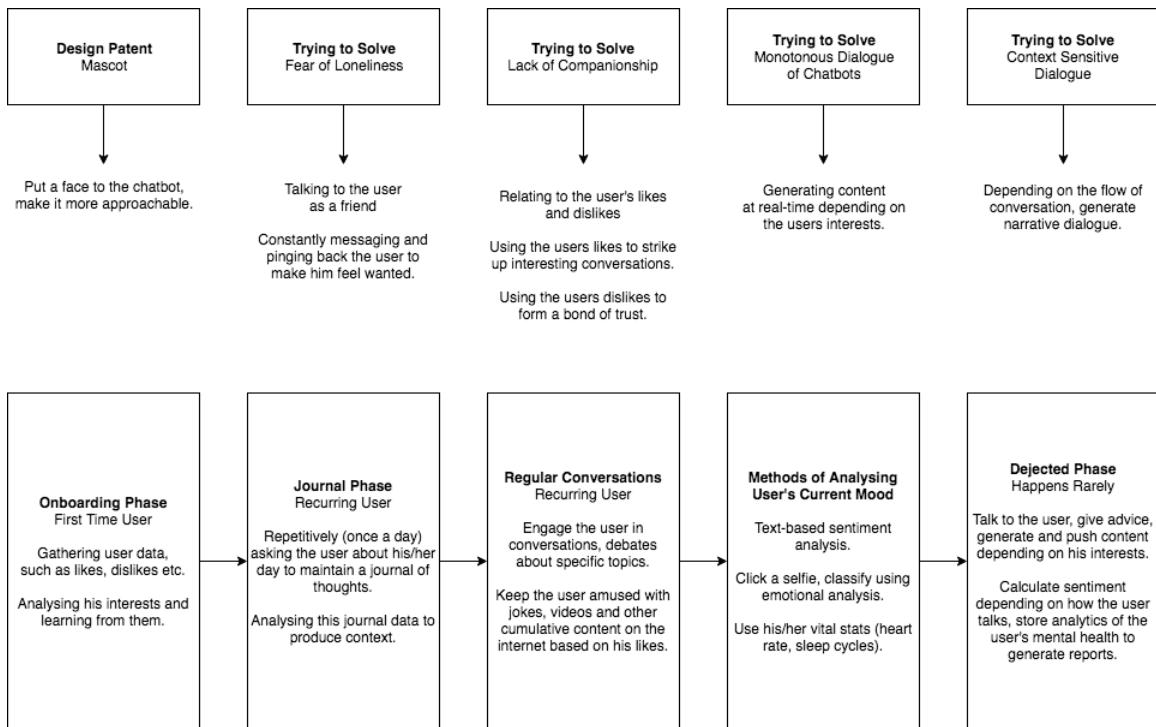


Figure 4.2: Inventive Steps

4.2.3 Conversational Trees

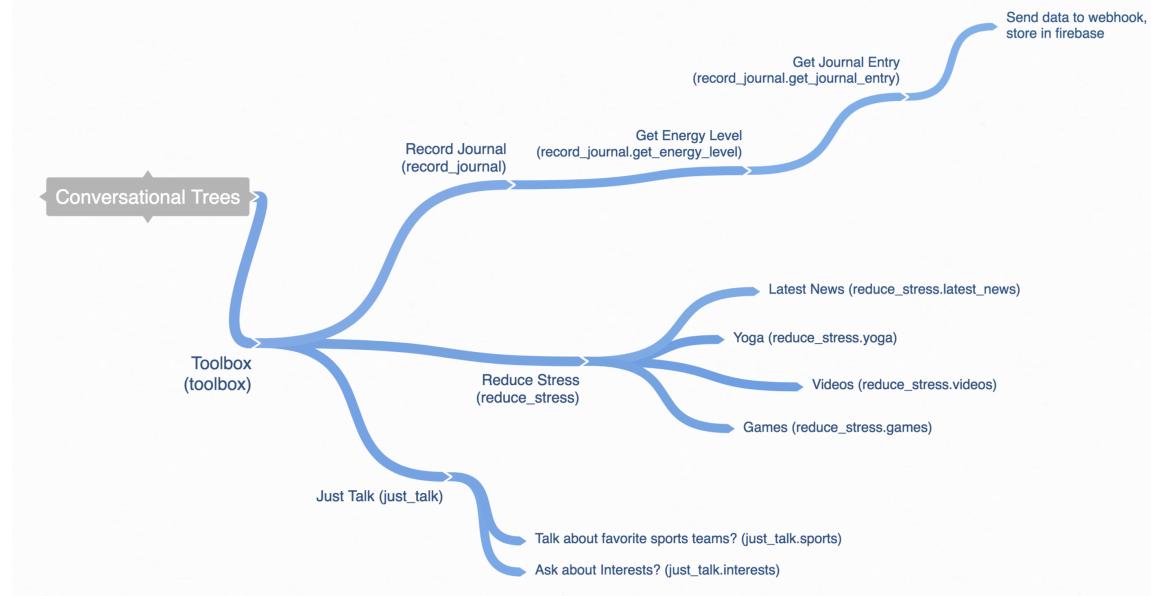


Figure 4.3: Conversational Trees of the Chatbot

4.3 Sentiment Analysis Anatomy

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a model that is used to predict the sentiment type and sentiment score of a particular sentence or a word input. It gives out an accurate score of each of the three sentiments - positive, negative and neutral. The reason for the preference of this tool compared to any other machine learning approach is the surpassing the complexity of a voluminous dataset. VADER has a Dictionary of lexicons that are used to predict the score of the sentences.

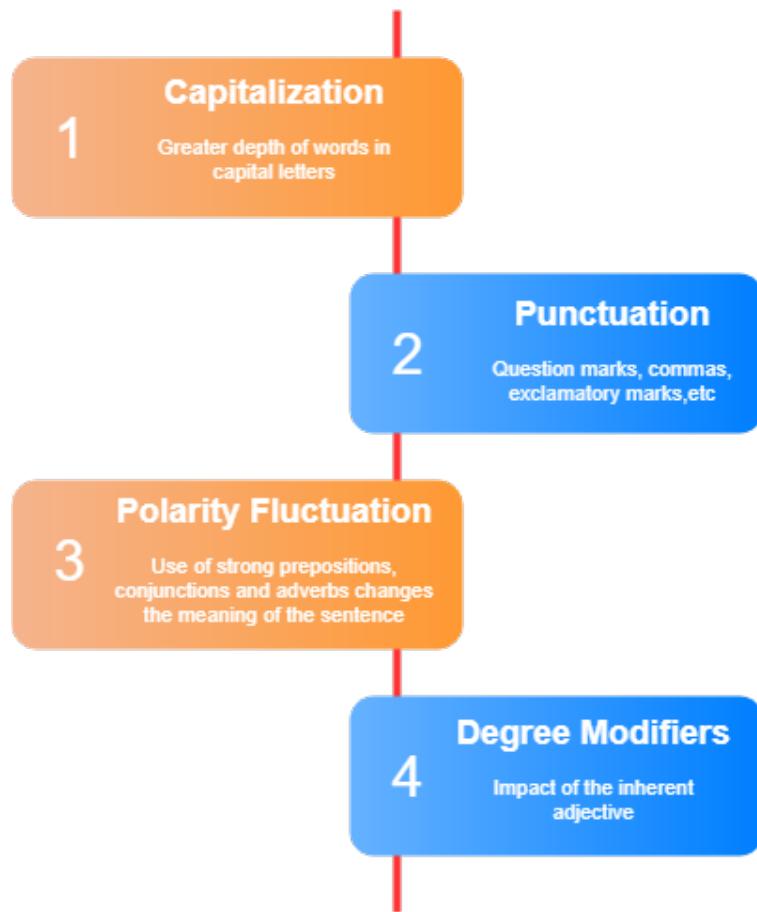


Figure 4.4: Steps of the VADER Model

4.4 Word2Vec (NLP) Structure

Word2vec is a two-layer neural net that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. While Word2vec is not a deep neural network, it turns text into a numerical form that deep nets can understand. Deeplearning4j implements a distributed form of Word2vec for Java and Scala, which works on Spark with GPUs. Word2vec's applications extend beyond parsing sentences in the wild. It can be applied just as well to genes, code, likes, playlists, social media graphs and other verbal or symbolic series in which patterns may be discerned.

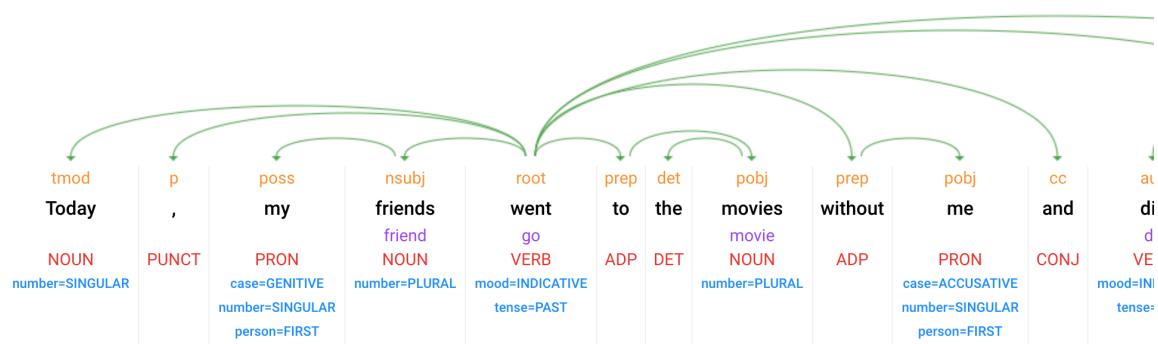


Figure 4.5: Syntax Parsing and Breakdown to Parts of Speech

Why? Because words are simply discrete states like the other data mentioned above, and we are simply looking for the transitional probabilities between those states: the likelihood that they will co-occur. So gene2vec, like2vec and follower2vec are all possible. With that in mind, the tutorial below will help you understand how to create neural embeddings for any group of discrete and co-occurring states. The purpose and usefulness of Word2vec is to group the vectors of similar words together in vectorspace. That is, it detects similarities mathematically. Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words. It does so without human intervention.

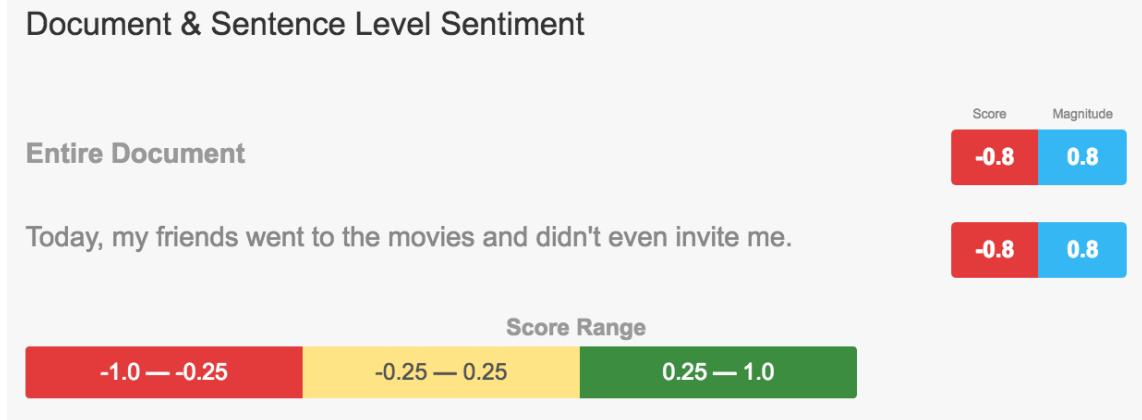


Figure 4.6: Sentiment Analysis Response

Given enough data, usage and contexts, Word2vec can make highly accurate guesses about a word's meaning based on past appearances. Those guesses can be used to establish a word's association with other words (e.g. "man" is to "boy" what "woman" is to "girl"), or cluster documents and classify them by topic. Those clusters can form the basis of search, sentiment analysis and recommendations in such diverse fields as scientific research, legal discovery, e-commerce and customer relationship management. The output of the Word2vec neural net is a vocabulary in which each item has a vector attached to it, which can be fed into a deep-learning net or simply queried to detect relationships between words.

Chapter 5

Implementation

5.1 Hardware Applications

Since this project and study was purely an algorithmic implementation and product engineering showcase, the hardware required is extremely minimal and found in abundance with most consumers. All the project requires is a Windows/Mac/Linux PC that has minimum system hardware requirements to build the source, and an Android/iOS device with Google Assistant or Facebook Messenger installed to be able to test and run the application.

5.2 Software Applications

- Operating System: Mac, Linux, Windows 8/10
- Programming Languages: Python, Node.js
- Chatbot Framework: DialogFlow
- Deep Learning Framework: Tensorflow
- Editors: Visual Studio Code, Jupyter Notebook
- Server: Heroku

5.3 Procedure

This section extensively talks about the artificial intelligence involved in our project.

5.3.1 Project File Structure

Organizing files on your computer is just like organizing anything else. Say you want to organize your clothes. You might sort each type of clothes into separate stacks. Then you might pair the socks or group all the shirts by color. Or, you could throw everything into one drawer and hope you can find the right pair of socks when you need it. And that's how we typically treat our files: we save files randomly to our Desktop and Documents folders, then waste time searching for files every day.

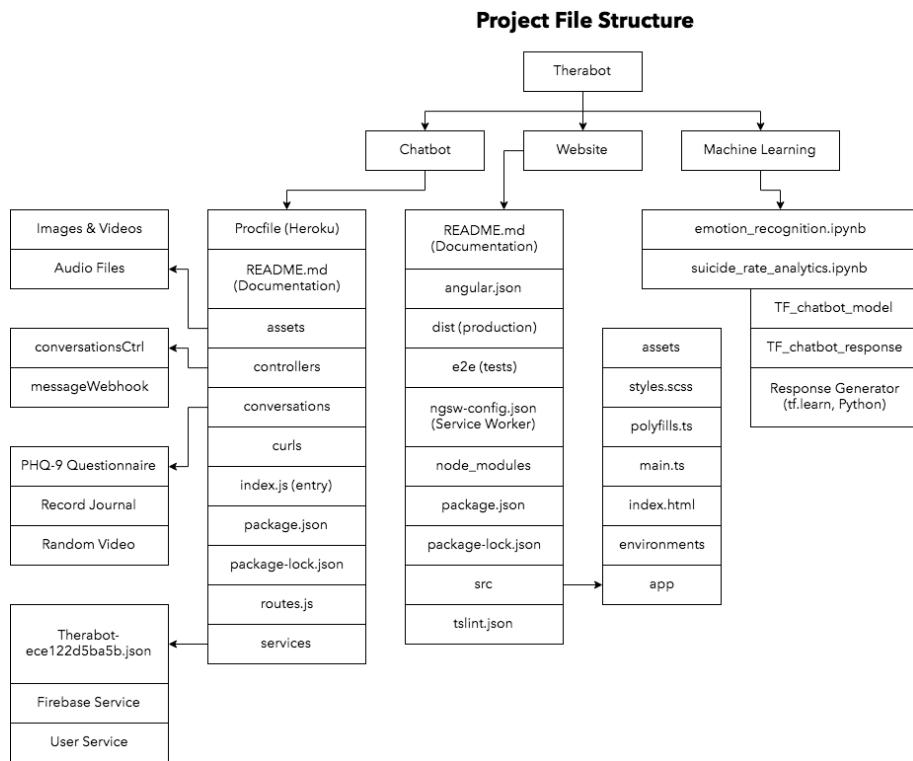


Figure 5.1: Project File Structure

Folder structures can help, just like drawers and dividers can keep your clothes organized. A folder structure is the way folders are organized on your computer. As folders are added over time, you can either keep them at the same level—like Folders 1, 2, and 3 in the chart below—or nest them within each other for a hierarchy—like Subfolders 1B and 1B-1 below. Nested folders generally make it easier to find specific files later, because you don't have to sift through all your files at once.

5.3.2 Dataset Collection

Our project requires an enormous amount of data in the form of conversational corpus. We have created a simple dataset in the form of questions and answers that would enable our tensorflow model to train with the generated responses. In order to incorporate the power of crowdsourcing, we have created a Google Form that would enable the community to facilitate wide range of responses.

Therabot Questionnaire

Hey there!

We are creating a dataset for our project, Therabot, and we need help. We need questions pertaining to anxiety, depression, work pressure, relationship trouble, etc.

Please write down questions you feel appropriate to ask someone who's feeling low. Mention the questions first and respond to them. You can even write phrases/keywords that would be appropriate to help them.

There can be many answers to a single question and many questions to the same responses. In such cases, write them in the next line. You can fill the form as many times as you want. We are trying to help out those people who are introverts currently suffering mental problem. This step of yours can help them.

For example,

Q: I am not getting much attention in my office. They don't appreciate my work
A: Time to start looking for a better job.
A: Why do you think they don't care.
A: Are they threatening you?
A: You should resign.
A: Have you clearly analysed your position.

NOTE: Your responses are *anonymous*. So please feel free to be open and submit as many questions and answers as you can. This could help a lot of people with serious mental health issues, and you could be a part of this change.

* Required

Figure 5.2: Therabot Questionnaire

We've also referred the Facebook Data Corpus and Cornell Movie Corpus for understanding the structure of data and scraped various datasets on Kaggle cherrypicking the from various conversational models to statistical data.

5.3.3 Setting up the Environment

We'll need to download the Natural Language Processing Library as well as Tensorflow for higher order numerical computation.

Installing NLTK: For Windows, Linux and Mac users we recommend installing Anaconda over which the NLP library can be set up. The following command will automatically fetch all the resources and install the Natural Language Toolkit.

```
sudo pip install -U nltk
```

Installing TensorFlow: An environment must be created in which TensorFlow will be installed. All the necessary pre-requisites and a step-by-step guide to a successful installation are available at <https://www.tensorflow.org/install/>.

5.3.4 Data Preprocessing

The accumulated dataset is subjected to a lot of functions which optimize its use during training. This cleaning process is what constitutes the preprocessing.

5.3.4.1 Tokenizing

The dataset contains paragraphs of sentences and well as individual sentences. In some cases, it is necessary to retrieve the sentence as a whole and sometimes as individual words. To facilitate this, the NLTK library has two predefined functions: `sent_tokenize` and `word_tokenize`.

```
In [13]: import nltk
from nltk import sent_tokenize
from nltk.tokenize import word_tokenize
print(nltk.__version__)
3.2.5

In [14]: paragraphs = 'Albert Einstein discovered the Theory of Relativity. Hendrik Lorentz also contributed to it.'
sentences = sent_tokenize(paragraphs)
print(sentences)
['Albert Einstein discovered the Theory of Relativity.', 'Hendrik Lorentz also contributed to it.']

In [15]: tokens = word_tokenize(paragraphs)
print(tokens)
['Albert', 'Einstein', 'discovered', 'the', 'Theory', 'of', 'Relativity', '.', 'Hendrik', 'Lorentz', 'also', 'contributed', 'to', 'it', '.']
```

Figure 5.3: Jupyter Notebook - Tokenizing

Similarly, punctuation and stop words can also be filtered out.

5.3.4.2 Stemming

While storing the user's data into Firebase, we need to peddle down the words in these sentences to their purest form or root. This process is called stemming.

```
In [15]: tokens = word_tokenize(paragraphs)
print(tokens)
['Albert', 'Einstein', 'discovered', 'the', 'Theory', 'of', 'Relativity', '.', 'Hendrik', 'Lorentz', 'also', 'contributed', 'to', 'it', '.']

In [16]: # stemming of words
from nltk.stem.porter import PorterStemmer
porter = PorterStemmer()
stemmed = [porter.stem(word) for word in tokens]
print(stemmed[:100])
['albert', 'einstein', 'discov', 'the', 'theori', 'of', 'rel', '.', 'hendrik', 'lorentz', 'also', 'contribut', 'to', 'it', '.']
```

Figure 5.4: Jupyter Notebook - Stemming

5.4 Algorithms

5.4.1 Linear Regression

When we have a single input attribute (x) and we want to use linear regression, this is called **Simple Linear Regression**.

With simple linear regression we want to model our data as follows:

$$y = B_0 + B_1 * x$$

This is a line where y is the output variable we want to predict, x is the input variable we know and B_0 and B_1 are coefficients we need to estimate.

B_0 is called the intercept because it determines where the line intercepts the y axis. In machine learning we can call this the bias, because it is added to offset all predictions that we make. The B_1 term is called the slope because it defines the slope of the line or how x translates into a y value before we add our bias.

The model is called Simple Linear Regression because there is only one input variable (x). If there were more input variables (e.g. x_1, x_2 , etc.) then this would be called Multiple Regression.

5.4.2 Gradient Descent

Optimization refers to the task of minimizing/maximizing an objective function $f(x)$ parameterized by x . In machine/deep learning terminology, it's the task of minimizing the cost/loss function $J(w)$ parameterized by the model's parameters $w \in R^d$.

Optimization algorithms (in case of minimization) have one of the following goals:

- Find the global minimum of the objective function. This is feasible if the objective function is convex, i.e. any local minimum is a global minimum.
- Find the lowest possible value of the objective function within its neighborhood. That's usually the case if the objective function is not convex as the case in most deep learning problems.

Gradient Descent is the most common optimization algorithm in machine learning and deep learning. It is a first-order optimization algorithm. This means it only takes into account the first derivative when performing the updates on the parameters. On each iteration, we update the parameters in the opposite direction of the gradient of the objective function $J(w)$ w.r.t the parameters where the gradient gives the direction of the steepest ascent. The size of the step we take on each iteration to reach the local minimum is determined by the learning rate α . Therefore, we follow the direction of the slope downhill until we reach a local minimum.

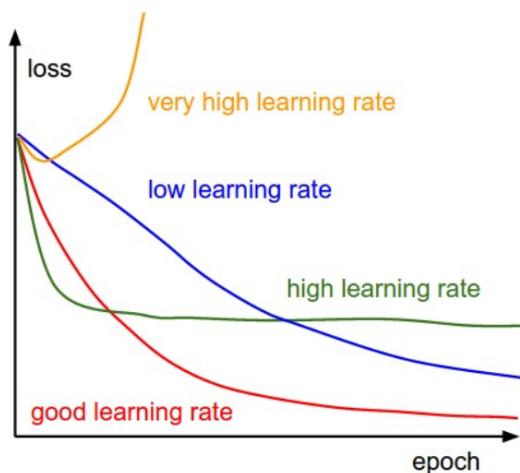


Figure 5.5: Gradient Descent with Different Learning Rates

$$\frac{\delta}{\delta m} = \frac{2}{N} \sum_{i=1}^N -x_i(y_i - (mx_i + b))$$

The coefficients used in simple linear regression can be found using stochastic gradient descent. Linear regression is a linear system and the coefficients can be calculated analytically using linear algebra. Stochastic gradient descent is not used to calculate the coefficients for linear regression in practice (in most cases). Linear regression does provide a useful exercise for learning stochastic gradient descent which is an important algorithm used for minimizing cost functions by machine learning algorithms.

B_0	B_1
0.01	0.01
0.0397	0.0694
0.066527	0.176708
0.08056049	0.21880847
0.1188144616	0.410078328
0.1235255337	0.4147894001
0.1439944904	0.4557273134
0.1543254529	0.4970511637
0.1578706635	0.5076867953
0.1809076171	0.6228715633

About 10 iterations or 2 epochs is a nice round number and a good place to stop. You could keep going if you wanted. Your values should match closely, but may have minor differences. You can plug each pair of coefficients back into the simple linear regression equation. This is useful because we can calculate a prediction for each training instance and in turn calculate the error.

5.4.3 Logistic Regression

Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

The fundamental equation of generalized linear model is:

$$g(E(y)) = \alpha + \beta x_1 + \gamma x_2$$

Here, $g()$ is the link function, $E(y)$ is the expectation of target variable and $\alpha + \beta_{x_1} + \gamma_{x_2}$ is the linear predictor (α, β, γ to be predicted). The role of link function is to ‘link’ the expectation of y to linear predictor.

On further deriving the equation, we get this:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta(Age)$$

This is the equation used in Logistic Regression. Here $(\frac{p}{1-p})$ is the odd ratio. Whenever the log of odd ratio is found to be positive, the probability of success is always more than 50%. A typical logistic model plot is shown below. You can see probability never goes below 0 and above 1.

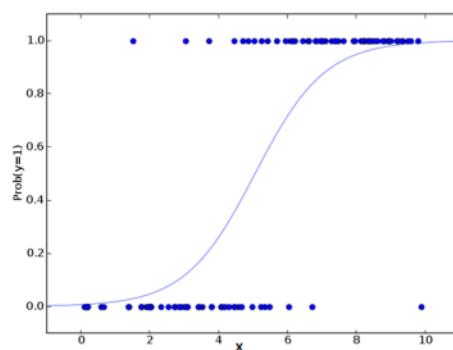


Figure 5.6: Typical Logistic Regression Model Plot

5.4.4 Softmax Function

Softmax function calculates the probabilities distribution of the event over ‘ n ’ different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

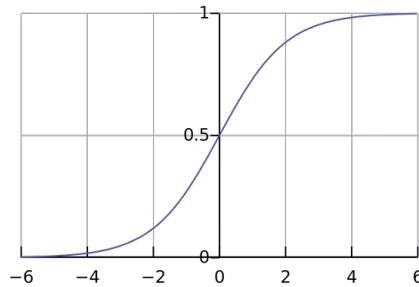


Figure 5.7: Softmax Function

The main advantage of using Softmax is the output probabilities range. The range will 0 to 1, and the sum of all the probabilities will be equal to one. If the softmax function used for multi-classification model it returns the probabilities of each class and the target class will have the high probability.

The formula computes the exponential (e -power) of the given input value and the sum of exponential values of all the values in the inputs. Then the ratio of the exponential of the input value and the sum of exponential values is the output of the softmax function.

$$P(y = j|z^{(i)}) = \phi_{softmax}(z^{(i)}) = \frac{e^z}{\sum_{j=1}^k e^z}$$

$$z = w_0x_0 + w_1x_1 + \dots + w_mx_m = \sum_{i=0}^m w_i x_i = w^T x$$

The softmax function is often used in the final layer of a neural network-based classifier. Such networks are commonly trained under a log loss (or cross-entropy) regime, giving a non-linear variant of multinomial logistic regression.

5.5 Diagrams

5.5.1 Data Flow Diagram

A Data Flow Diagram (DFD) is traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole.

There are four major components or external agents involved in the system:

- User
- DialogFlow Framework
- Machine Learning Model
- Firebase Database

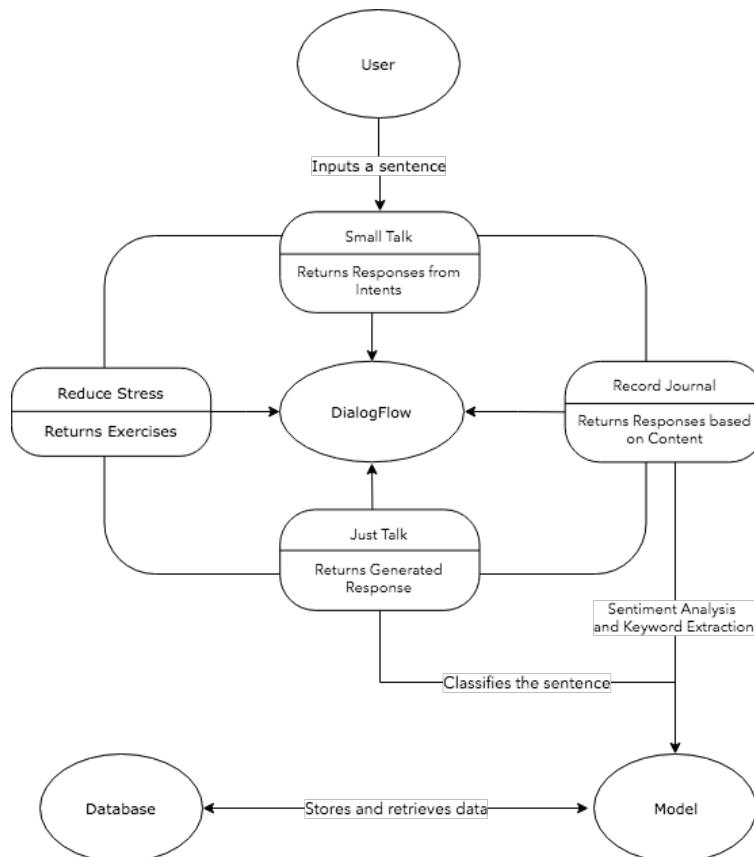


Figure 5.8: Data Flow Diagram

5.5.2 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

We have two different sequence diagrams here:

- Onboarding Phase
- Regular Conversations

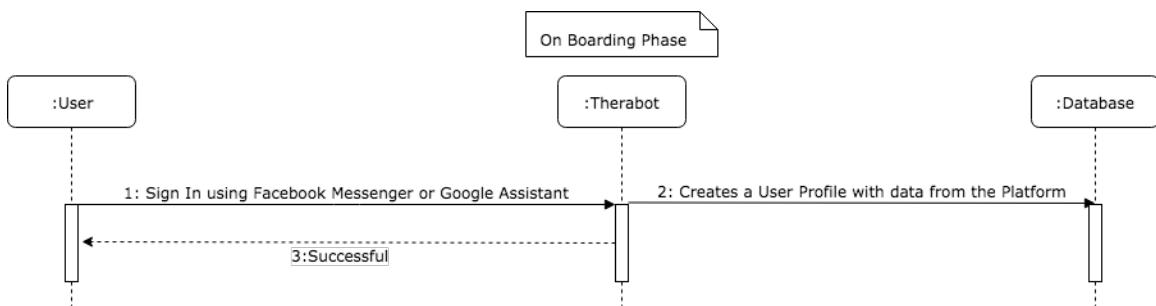


Figure 5.9: Sequence Diagram - Onboarding Phase

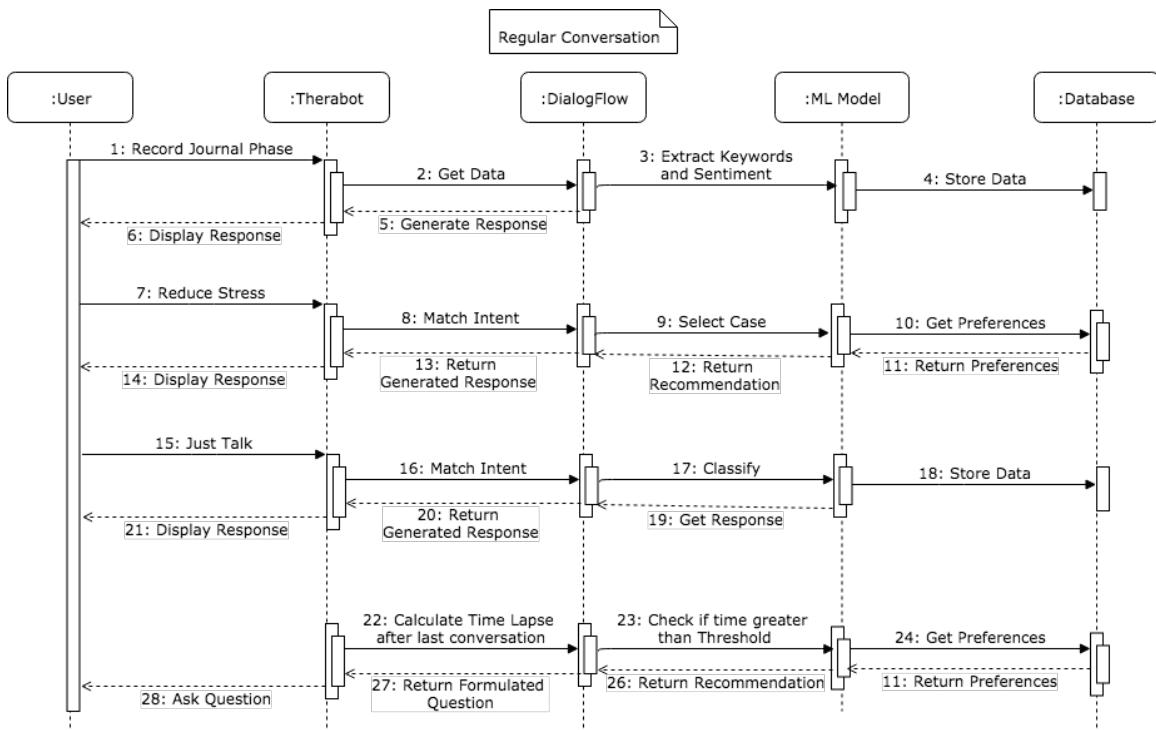


Figure 5.10: Sequence Diagram - Regular Conversations

Chapter 6

Experimentation & Results

6.1 Tensorflow

We are using Tensorflow as our Deep Learning Framework to train the train the model with our dataset. TFLearn is a high-level API for fast neural network building and training, and then showing how TFLearn layers, built-in ops and helpers can directly benefit any model implementation with Tensorflow.

6.1.1 Benefits of TFLearn

- Compact Code
- Swift construction of the neural network regardless of the underlying parameters
- Demystifies the customary training
- Works like a charm with any the model
- Provides many optimization and loss functions for custom models

6.1.2 TFLearn Employment

Our experiments are limited to the core and estimator layers of this vast API. We are using three functions from the `tflearn.core` layer. They are:

- Input Data
- Fully Connected
- Regression Layer

6.1.2.1 Input Data

The `tflearn.input_data` is an input layer to the network. It's major function is to provide a glance of input data. The arguments such as shape and placeholder that specify the shape, size of inputs, batch size, etc.

6.1.2.1.1 Arguments

- **shape:** list of int. An array or tuple representing input data shape. It is required if no placeholder is provided. First element should be 'None' (representing batch size), if not provided, it will be added automatically.
- **placeholder:** A Placeholder to use for feeding this layer (optional). If not specified, a placeholder will be automatically created. You can retrieve that placeholder through graph key: 'INPUTS', or the 'placeholder' attribute of this function's returned tensor.
- **dtype:** tf.type, Placeholder data type (optional). Default: float32.
- **data_preprocessing:** A DataPreprocessing subclass object to manage real-time data pre-processing when training and predicting (such as zero center data, std normalization...).
- **data_augmentation:** DataAugmentation. A DataAugmentation subclass object to manage real-time data augmentation while training (such as random image crop, random image flip, random sequence reverse...).
- **name:** str. A name for this layer (optional).

6.1.2.2 Fully Connected

The `tflearn.fully_connected` layer is responsible for the creation of neurons in the hidden layer. It takes many arguments out of which the first two are of much importance. The `ata` parameter is the incoming 2D Tensor and the second parameter is the number of units (neurons) to create.

6.1.2.2.1 Arguments

- **incoming:** Tensor. Incoming (2+)D Tensor.
- **n_units:** int, number of units for this layer.
- **activation:** str (name) or function (returning a Tensor). Activation applied to this layer (see `tflearn.activations`). Default: 'linear'.
- **bias:** bool. If True, a bias is used.
- **weights_init:** str (name) or Tensor. Weights initialization. (see `tflearn.initializations`) Default: 'truncated_normal'.
- **bias_init:** str (name) or Tensor. Bias initialization. (see `tflearn.initializations`) Default: 'zeros'.
- **regularizer:** str (name) or Tensor. Add a regularizer to this layer weights (see `tflearn.regularizers`). Default: None.
- **weight_decay:** float. Regularizer decay parameter. Default: 0.001.
- **trainable:** bool. If True, weights will be trainable.
- **restore:** bool. If True, this layer weights will be restored when loading a model.
- **reuse:** bool. If True and 'scope' is provided, this layer variables will be reused (shared).
- **scope:** str. Define this layer scope (optional). A scope can be used to share variables between layers. Note that scope will override name.
- **name:** A name for this layer (optional). Default: 'FullyConnected'.

6.1.2.3 Regression Layer

The `tflearn.regression` is a part of the estimator layer. It focuses on the optimization of the incurred losses due to the gradient descent algorithm. Both linear and logistic regression can be performed.

6.1.2.3.1 Arguments

- **incoming:** Tensor. Incoming 2-D Tensor.
- **placeholder:** Tensor. This regression target (label) placeholder. If 'default', a placeholder will be added automatically. You can retrieve that placeholder through graph key: 'TARGETS', or the 'placeholder' attribute of this function's returned tensor. If you do not want to use any target, set placeholder to 'None'.
- **optimizer:** str (name), Optimizer or function. Optimizer to use. Default: 'adam' (Adaptive Moment Estimation).
- **loss:** str (name) or function. Loss function used by this layer optimizer. Default: 'categorical_crossentropy'.
- **metric:** str, Metric or function. The metric to be used. Default: 'default' metric is 'accuracy'. To disable metric calculation, set it to 'None'.
- **learning_rate:** float. This layer optimizer's learning rate.
- **dtype:** tf.types. This layer placeholder type. Default: tf.float32.
- **batch_size:** int. Batch size of data to use for training. tflearn supports different batch size for every optimizers. Default: 64.
- **shuffle_batches:** bool. Shuffle or not this optimizer batches at every epoch. Default: True.
- **to_one_hot:** bool. If True, labels will be encoded to one hot vectors. 'n_classes' must then be specified.
- **n_classes:** int. The total number of classes. Only required when using 'to_one_hot' option.
- **trainable_vars:** list of Variable. If specified, this regression will only update given variable weights. Else, all trainable variable are going to be updated.
- **restore:** bool. If False, variables related to optimizers such as moving averages will not be restored when loading a pre-trained model.
- **op_name:** A name for this layer optimizer (optional). Default: optimizer op name.

- **validation_monitors:** list of Tensor objects. List of variables to compute during validation, which are also used to produce summaries for output to TensorBoard. For example, this can be used to periodically record a confusion matrix or AUC metric, during training. Each variable should have rank 1, i.e. shape [None].
- **validation_batch_size:** int or None. Specifies the batch size to be used for the validation data feed.
- **name:** A name for this layer's placeholder scope.

6.1.3 Model

- **Defining the shape:** The shape of the input data is provided by length of first vector and the batch size for our data is initially unspecified.
- **Creating the Input Hidden Layers:** Two hidden layers each have 8 neurons in them.
- **Creating the Output Layer:** One Fully Connected Output Layer with 8 neurons and a Softmax Activation Function
- **Estimator Layer:** Linear or Logistic Regression

6.1.4 Responses

The neural network is built and trained for the data. The response phase works such that the given sentence is classified into a category that best describes the type. Based on this context, the responses are delivered to the user.

```
st = "Nobody appreciates my work"
classify(st)

Out[19]: [('work', 0.7141276)]

In [22]: response(st)
Time to start looking for a better job.
```

Figure 6.1: Jupyter Notebook - Sample Response

Greater the expanse and variety of dataset, better the training and furnishing of responses.

6.1.4.1 Proposed Headway

The current training scenario has two parameters:

- Number of Epochs
- Batch Size

6.1.4.1.1 Increasing the Number of Epochs By maintaining a constant Batch Size of 8, we gradually increment the Number of Epochs.

```
In [35]: # Define model and setup tensorflow
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=500, batch_size=8, show_metric=True)
model.save('model.tflearn')

Training Step: 2499 | total loss: 0.23919 | time: 0.016s
| Adam | epoch: 500 | loss: 0.23919 - acc: 0.9571 -- iter: 32/33
Training Step: 2500 | total loss: 0.23467 | time: 0.016s
| Adam | epoch: 500 | loss: 0.23467 - acc: 0.9614 -- iter: 33/33
--
INFO:tensorflow:C:\Users\Varsha\Documents\Python_Scripts\Therabot\Books\TF Model Response\model.tflearn is not in all_model_checkpoint_paths. Manually adding it.
```

Figure 6.2: Jupyter Notebook - Number of Epochs = 500

```
In [10]: # Define model and setup tensorflow
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=1000, batch_size=8, show_metric=True)
model.save('model.tflearn')

Training Step: 4999 | total loss: 0.08910 | time: 0.014s
| Adam | epoch: 1000 | loss: 0.08910 - acc: 0.9780 -- iter: 32/33
Training Step: 5000 | total loss: 0.08742 | time: 0.021s
| Adam | epoch: 1000 | loss: 0.08742 - acc: 0.9802 -- iter: 33/33
--
INFO:tensorflow:C:\Users\Varsha\Documents\Python_Scripts\Therabot\Books\TF Model Response\model.tflearn is not in all_model_checkpoint_paths. Manually adding it.
```

Figure 6.3: Jupyter Notebook - Number of Epochs = 1000

```
In [29]: # Define model and setup tensorflow
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=1500, batch_size=8, show_metric=True)
model.save('model.tflearn')

Training Step: 7499 | total loss: 0.08547 | time: 0.019s
| Adam | epoch: 1500 | loss: 0.08547 - acc: 0.9711 -- iter: 32/33
Training Step: 7500 | total loss: 0.08140 | time: 0.025s
| Adam | epoch: 1500 | loss: 0.08140 - acc: 0.9740 -- iter: 33/33
--
INFO:tensorflow:C:\Users\Varsha\Documents\Python_Scripts\Therabot\Books\TF Model Response\model.tflearn is not in all_model_checkpoint_paths. Manually adding it.
```

Figure 6.4: Jupyter Notebook - Number of Epochs = 1500

```
In [31]: # Define model and setup tensorflow
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=2000, batch_size=8, show_metric=True)
model.save('model.tflearn')

Training Step: 9999 | total loss: 0.16538 | time: 0.021s
| Adam | epoch: 2000 | loss: 0.16538 - acc: 0.9651 -- iter: 32/33
Training Step: 10000 | total loss: 0.15286 | time: 0.024s
| Adam | epoch: 2000 | loss: 0.15286 - acc: 0.9686 -- iter: 33/33
--
INFO:tensorflow:C:\Users\Varsha\Documents\Python_Scripts\Therabot\Books\TF Model Response\model.tflearn is not in all_model_checkpoint_paths. Manually adding it.
```

Figure 6.5: Jupyter Notebook - Number of Epochs = 2000

Number of Epochs	Loss Incurred	Accuracy
500	0.23467	0.9614
1000	0.08742	0.9802
1500	0.08140	0.9740
2000	0.15286	0.9686
2250	0.20507	0.9544
2456(2500)	0.27993	0.9525

The outcome of any DNN model is two things. First, it should return the desired response precisely. Second, the loss at each iteration during training must be optimized. From the above table, it is evident that by increasing the batch size the losses are reduced.

But we do observe that when the `n_epoch = 2000`, the losses increase. This is the case of overfitting the data. It means that your model does not learn the data, it memorizes the data. You have to find the accuracy of validation data for each epoch or maybe iteration to investigate whether it over-fits or not. To deal with this problem, another approaches are used for avoiding the problem. Adding noise to different parts of models, like drop out or somehow batch normalization with a moderated batch size, help these learning algorithms not to over-fit even after so many epochs.

Since, accuracy is the prime factor, our model works at **`n_epoch = 1000`** with a **very high accuracy of 0.98**.

6.1.4.1.2 Increasing the Batch Size By maintaining the number of epochs at 1000, we increase the batch size.

```
In [59]: # Define model and setup tensorflow
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=1000, batch_size=4, show_metric=True)
model.save('model.tflearn')

Training Step: 8773 | total loss: 0.02439 | time: 0.030s
| Adam | epoch: 975 | loss: 0.02439 - acc: 0.9692 -- iter: 28/33
```

Figure 6.6: Jupyter Notebook - Batch Size = 4

```
In [61]: # Define model and setup tensorflow
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=1000, batch_size=8, show_metric=True)
model.save('model.tflearn')

Training Step: 4999 | total loss: 0.51368 | time: 0.018s
| Adam | epoch: 1000 | loss: 0.51368 - acc: 0.9155 -- iter: 32/33
Training Step: 5000 | total loss: 0.46362 | time: 0.021s
| Adam | epoch: 1000 | loss: 0.46362 - acc: 0.9239 -- iter: 33/33
--
INFO:tensorflow:C:\Users\Varsha\Documents\Python_Scripts\Therabot\Books\TF Model Response\model.tflearn is not in all_model_checkpoint_paths. Manually adding it.
```

Figure 6.7: Jupyter Notebook - Batch Size = 8

```
In [63]: # Define model and setup tensorflow
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=1000, batch_size=16, show_metric=True)
model.save('model.tflearn')

Training Step: 2999 | total loss: 0.20305 | time: 0.014s
| Adam | epoch: 1000 | loss: 0.20305 - acc: 0.9489 -- iter: 32/33
Training Step: 3000 | total loss: 0.18486 | time: 0.018s
| Adam | epoch: 1000 | loss: 0.18486 - acc: 0.9540 -- iter: 33/33
--
INFO:tensorflow:C:\Users\Varsha\Documents\Python_Scripts\Therabot\Books\TF Model Response\model.tflearn is not in all_model_checkpoint_paths. Manually adding it.
```

Figure 6.8: Jupyter Notebook - Batch Size = 16

```
In [65]: # Define model and setup tensorflow
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=1000, batch_size=24, show_metric=True)
model.save('model.tflearn')

Training Step: 1999 | total loss: 0.81430 | time: 0.003s
| Adam | epoch: 1000 | loss: 0.81430 - acc: 0.8657 -- iter: 24/33
Training Step: 2000 | total loss: 0.74673 | time: 0.007s
| Adam | epoch: 1000 | loss: 0.74673 - acc: 0.8791 -- iter: 33/33
--
INFO:tensorflow:C:\Users\Varsha\Documents\Python_Scripts\Therabot\Books\TF Model Response\model.tflearn is not in all_model_checkpoint_paths. Manually adding it.
```

Figure 6.9: Jupyter Notebook - Batch Size = 24

```
In [67]: # Define model and setup tensorflow
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=1000, batch_size=36, show_metric=True)
model.save('model.tflearn')

...
Training Step: 995 | total loss: 0.51059 | time: 0.004s
| Adam | epoch: 995 | loss: 0.51059 - acc: 0.9548 -- iter: 33/33
...
Training Step: 996 | total loss: 0.50349 | time: 0.004s
| Adam | epoch: 996 | loss: 0.50349 - acc: 0.9563 -- iter: 33/33
...
Training Step: 997 | total loss: 0.49693 | time: 0.004s
| Adam | epoch: 997 | loss: 0.49693 - acc: 0.9576 -- iter: 33/33
...
Training Step: 998 | total loss: 0.49087 | time: 0.003s
| Adam | epoch: 998 | loss: 0.49087 - acc: 0.9589 -- iter: 33/33
...
Training Step: 999 | total loss: 0.48524 | time: 0.003s
| Adam | epoch: 999 | loss: 0.48524 - acc: 0.9599 -- iter: 33/33
...
Training Step: 1000 | total loss: 0.48001 | time: 0.004s
| Adam | epoch: 1000 | loss: 0.48001 - acc: 0.9609 -- iter: 33/33
...
INFO:tensorflow:C:\Users\Varsha\Documents\Python_Scripts\Therabot\Books\TF Model Response\model.tflearn is not in all_model_checkpoint_paths. Manually adding it.
```

Figure 6.10: Jupyter Notebook - Batch Size = 36

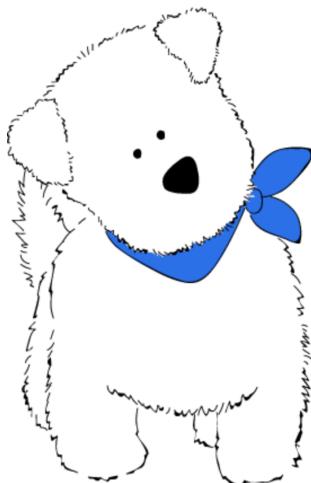
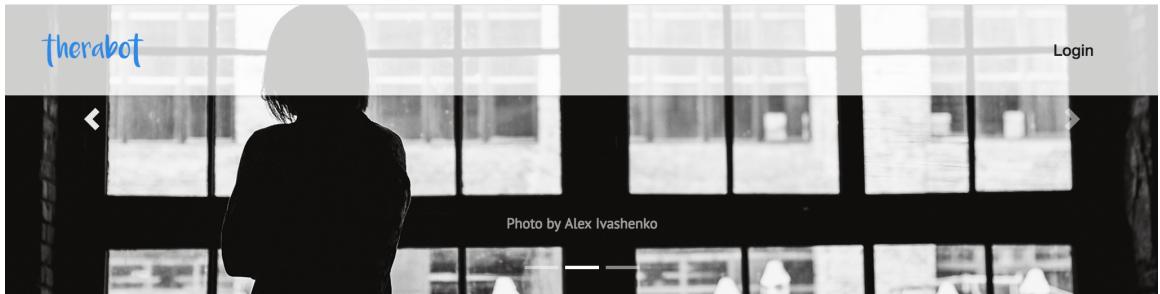
Batch Size	Loss Incurred	Accuracy
4	0.02439	0.9692
8	0.46362	0.9239
16	0.18486	0.9540
24	0.74673	0.8791
36	0.48001	0.9609
48	0.83720	0.8264

On deeper analysis of the results, we understand that by increasing the batch_size we approach the case of overfitting the data. Hence, best accuracy is obtained when the batch size is 8(or 16) for most of the given kinds of dataset.

6.2 Screenshots

6.2.1 Website

Below are some screenshots of the website, which is hosted on the cloud. It is a Progressive Web App (PWA), meaning it can function as a functioning native Android/iOS application or a desktop web application on the same code.



Introduction

This study aims to highlight the use cases of a variety of technologies in the field of Artificial Intelligence, Speech Synthesis, Personalization, User Character Segmentation and Automated Text/Speech-based Chatbots. To illustrate the uses, we have chosen a specific field of expertise, mental wellness. We have built an artificially chatbot that is designed to understand the complexities of it's user, help the user feel more self-aware and spontaneously check back on the user from time to time. We have also devised a method to extract key likes and dislikes from the user's messages using NLP techniques.

The goal of Therabot as a product to the consumer is not to overthrow the need for real life therapists, but to simply allow the user to feel more comfortable talking about his/her experiences, thus decreasing the stigma that perpetuates around mental health in our country and around the world. We want to make mental wellness more accessible and educate users on the true meaning of depression and anxiety. The objective of our research is to show how chatbots powered by artificial intelligence can be not only used for service based QA systems, but also in unique niche markets such as medical health, distress systems, education etc. Using our unique methods of generating the user's responses and creating a schema for each individual user, we are able to bring more personalization and unique feeling to every user.

Figure 6.11: Introduction

Modules

 Journal Writing

Write about your day, store your deepest thoughts and feelings and be assured that no one else can ever read your mind.

 Photo Albums

Store your most memorable photos and memories along with a story that makes you feel better and you can always get back to it.

 Just Talk

Powered by our intelligent AI chatbot, you can just have a simple conversation with Therabot if you need someone to talk to.

Cognitive Behavioral Therapy

Cognitive behavioral therapy (CBT) is a short-term therapy technique used by counselors and therapists to teach individuals to change their unwanted behaviors by changing their thought patterns. The premise of cognitive behavioral therapy is that our thought patterns (cognition) and interpretations of life events greatly influence how we behave and, ultimately, how we feel.

CBT is a form of psychotherapy that focuses on how your thoughts, beliefs and attitudes affect your feelings and behavior. It aims to teach you effective coping strategies for dealing with different problems throughout life. CBT can help you make sense of overwhelming problems by breaking them down into smaller parts.

One of the key tenets of CBT is that distorted thinking leads to distress and problematic behaviors, whereas thinking realistically with less negativity allows individuals to respond to challenging life circumstances in an effective way. Research shows this technique is an effective therapy for not only depression and panic disorder, but many illnesses and dysfunctional behaviors.

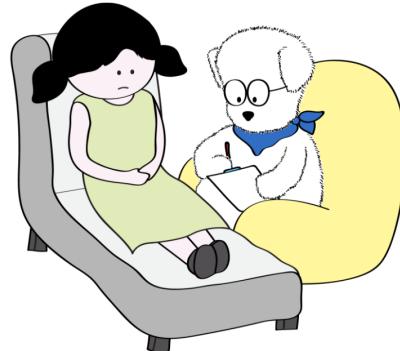


Figure 6.12: Cognitive Behavioral Therapy

therabot

[Dashboard](#) [Logout](#)



Welcome, Amruth!

Figure 6.13: Welcome - User Dashboard

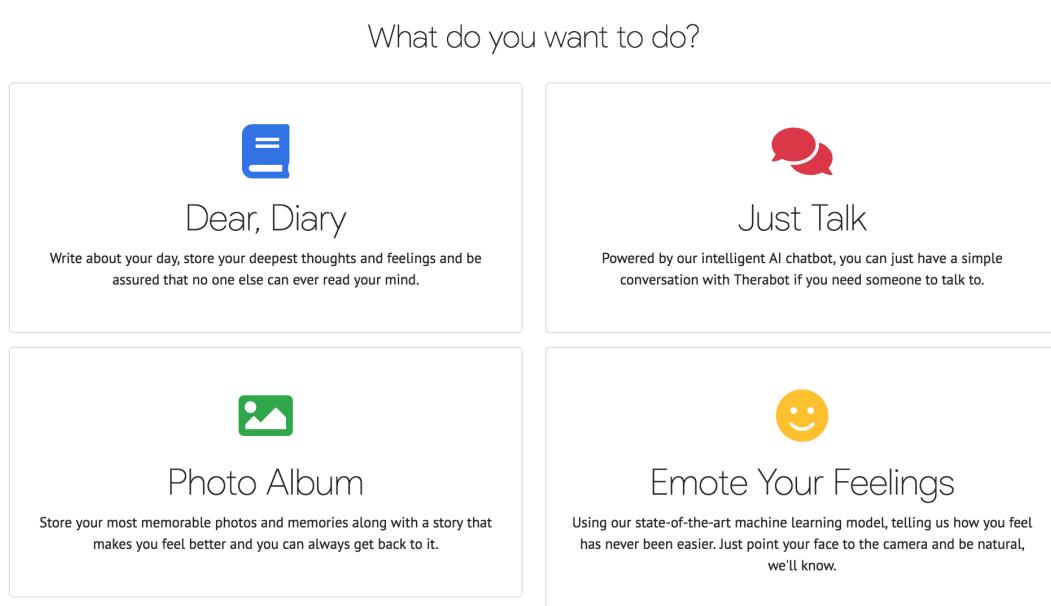
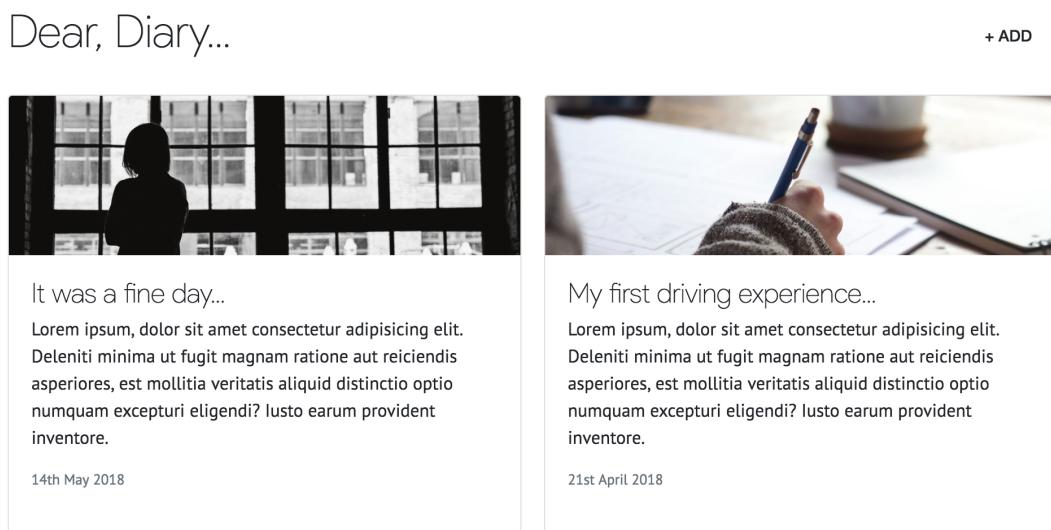
**Figure 6.14:** Modules on the Website**Figure 6.15:** Module #1 - Dear Diary

Photo Album

+ ADD

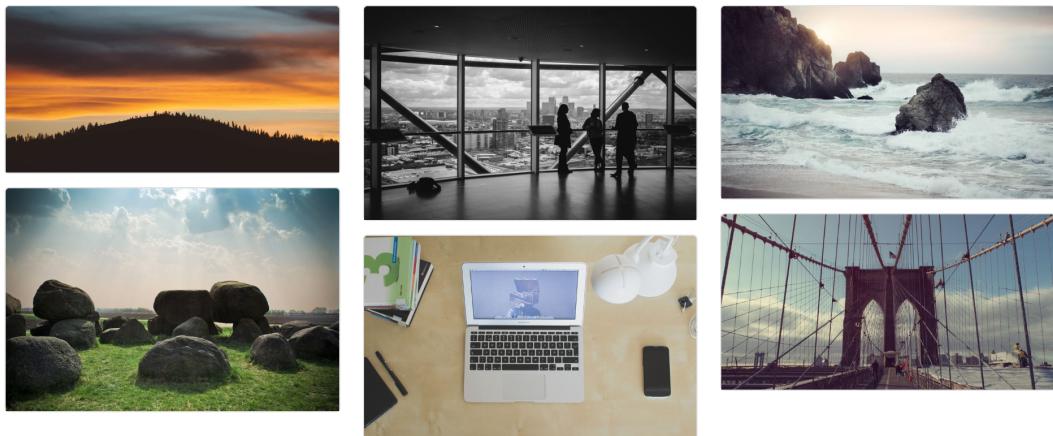


Figure 6.16: Module #2 - Photo Album

Emote Your Feelings

Using our state of the art machine learning model, we can easily detect faces and their emotions in a given photo, of upto 25 faces. Use this service for the times when you want to tell Therabot how you're feeling, but you just can't put it to words...

Don't worry, we don't store any of your photos uploaded here. We work on a pure predict and forget policy.

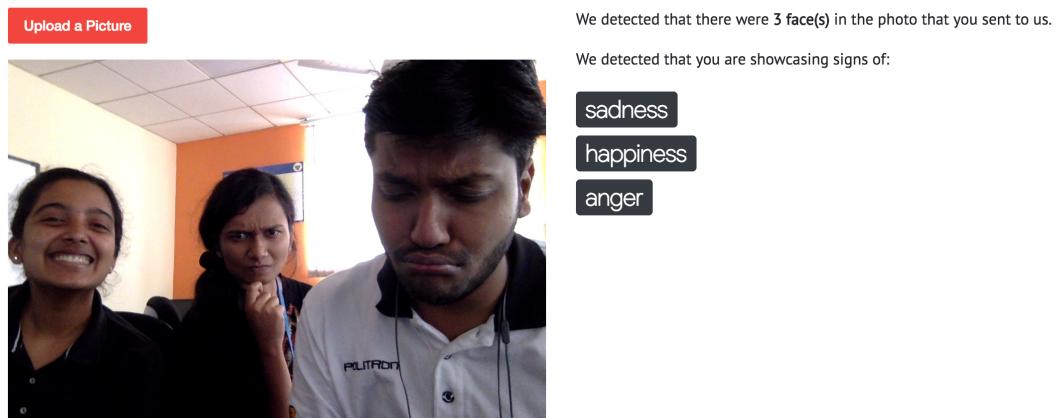


Figure 6.17: Module #3 - Emote your Feelings

6.2.2 Chatbot

Below are some screenshots of the chatbot, and it's various conversational trees showcased in action:

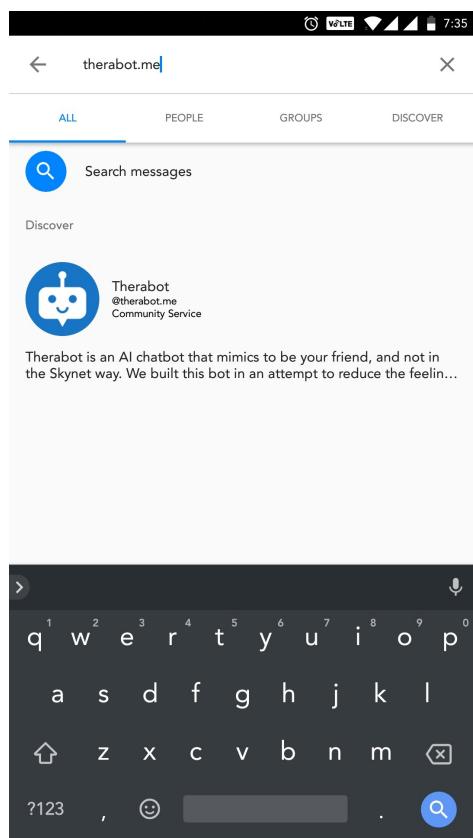


Figure 6.18: Facebook Messenger

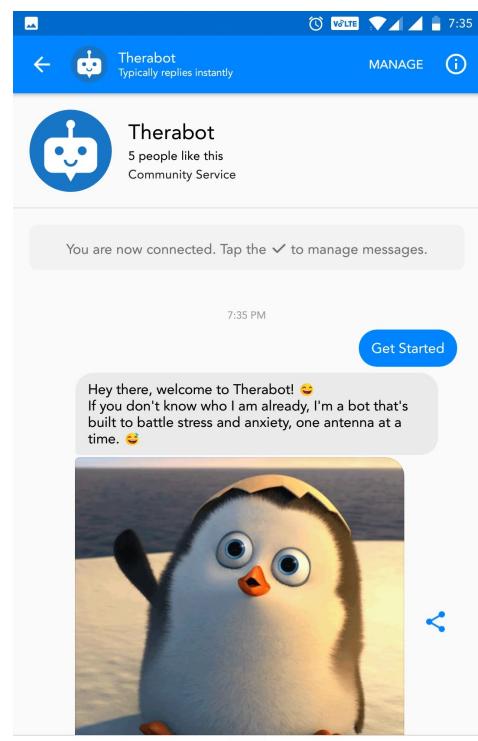
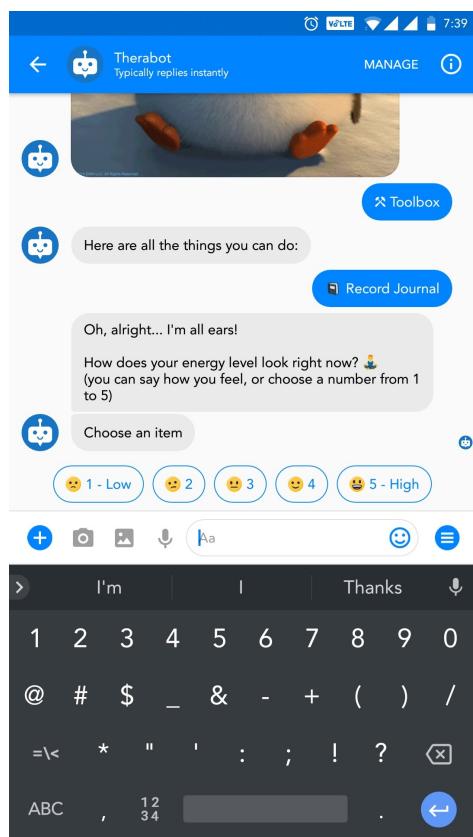
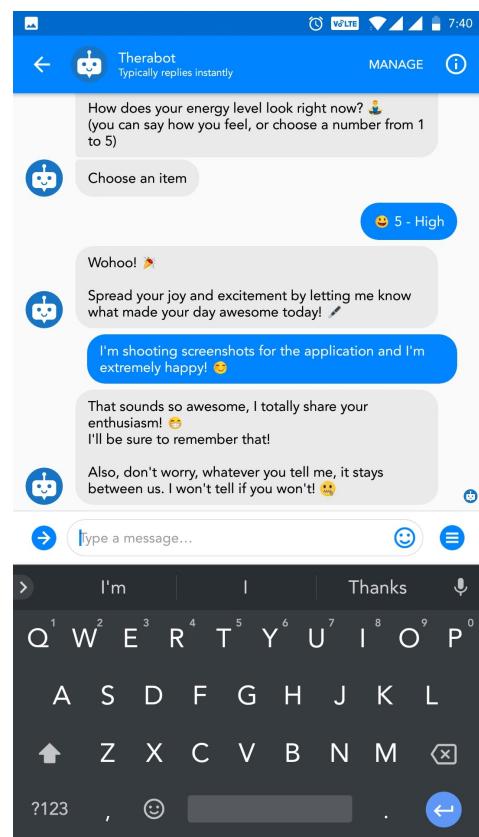
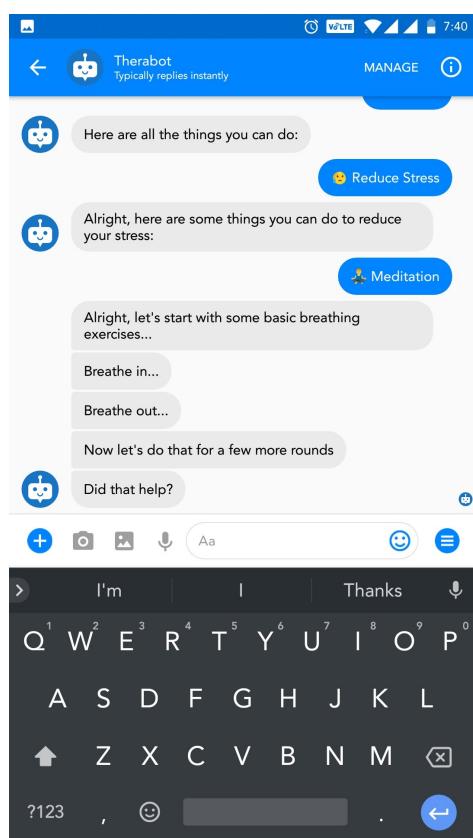
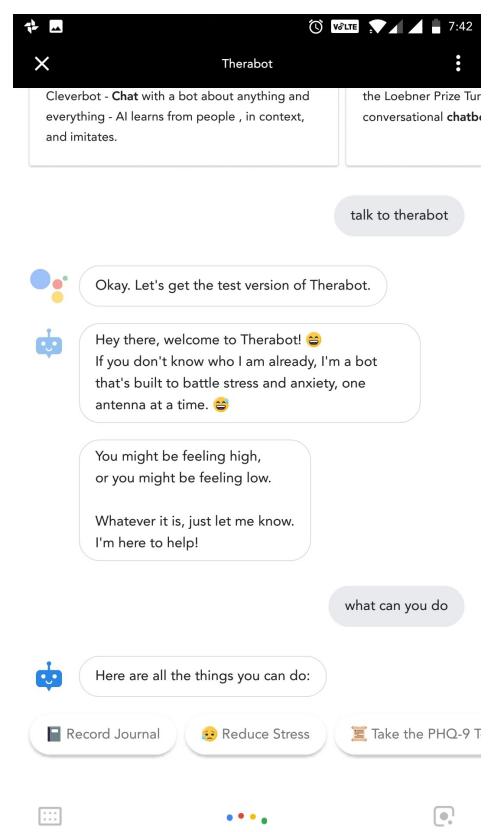
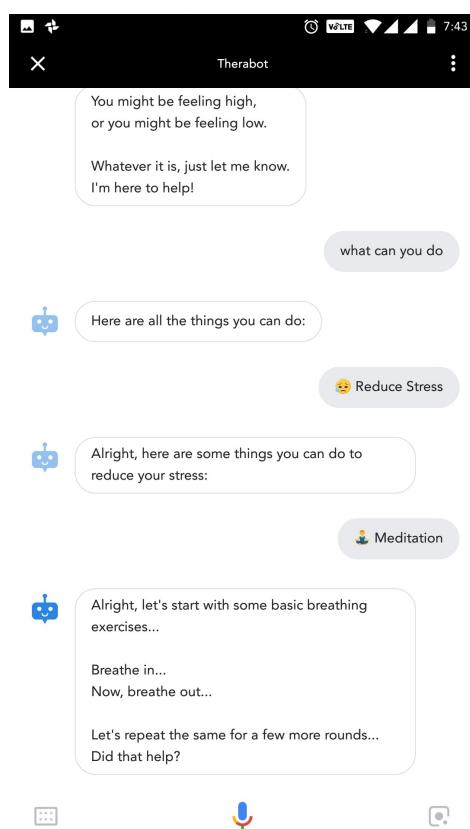
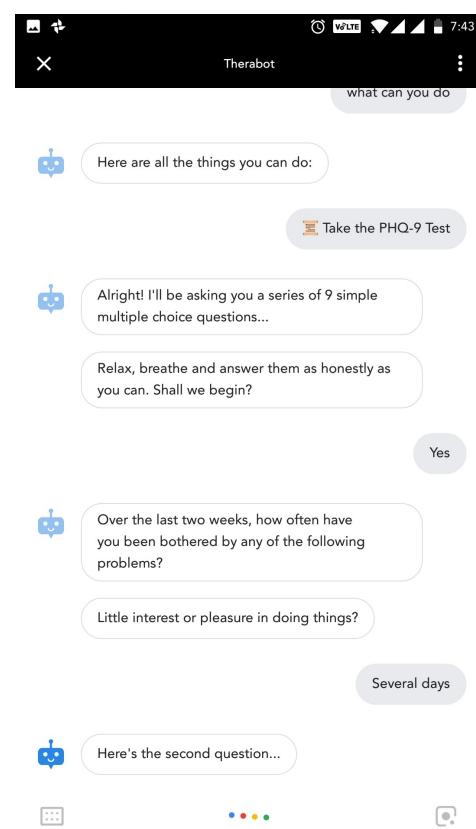


Figure 6.19: User Onboarding

**Figure 6.20:** Record Journal**Figure 6.21:** Saving Journal Entry

**Figure 6.22:** Meditation**Figure 6.23:** Google Assistant

**Figure 6.24:** Meditation w/ SSML**Figure 6.25:** PHQ9 Test

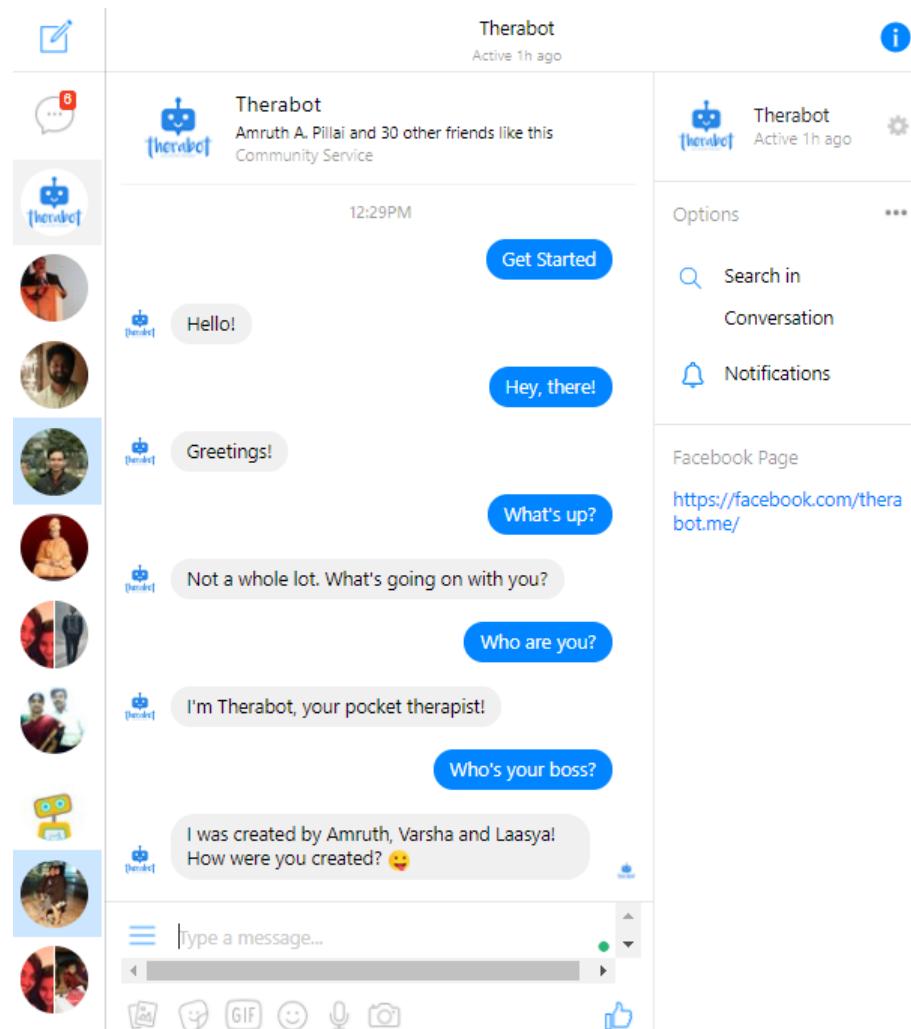


Figure 6.26: Tablet Screenshot

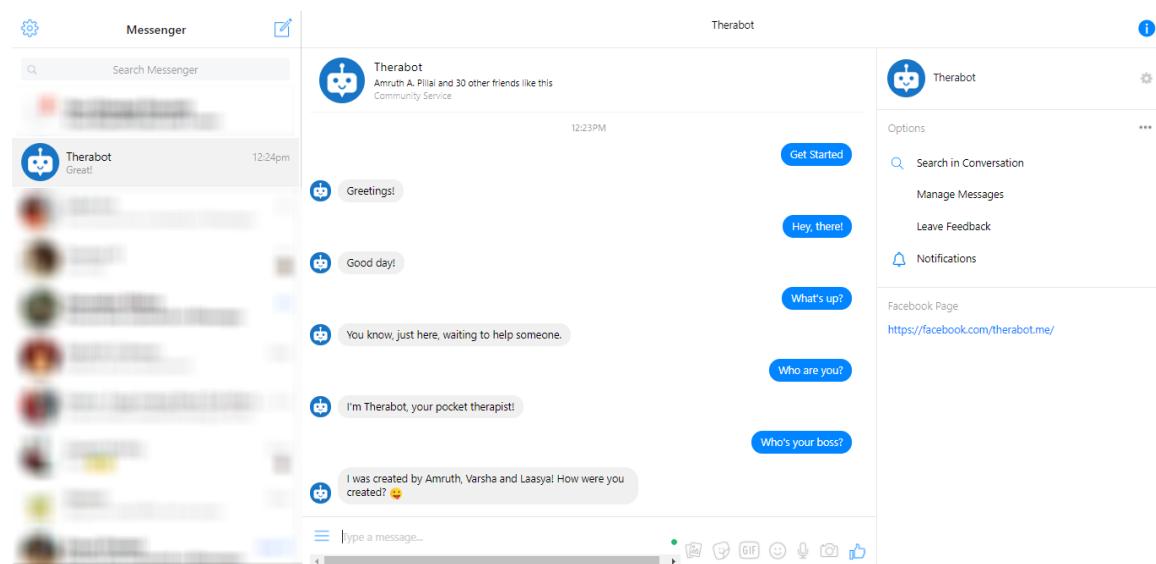


Figure 6.27: Desktop Screenshot

6.3 Testing

6.3.1 Chatbot Testcases

We brought up our dataset of questions and answers and validated it across our testcases, it passed on all counts and can be verified by talking to the chatbot with the specified input parameters.

Input	Classification	Expected Output
I think I don't look good.	Personal Appearances	Maybe you're looking into a funny mirror. You look awesome!
My style is out of date.		I think you look just great, don't read too much into it!
I think I dress like an old person.		
Nobody appreciates my work	Work Pressure	Why do you feel so? I'm sure they appreciate your work.
nobody likes my work		
My boss doesn't like my work.		Why do you think so? Maybe they do, but they're silent about it.
I'm not able to handle all the extra work.	Work Pressure	You should hire an assistant if you have too much work.
I want an assistant at work.		Talk to your superiors. I'm sure they'll understand.
There's just so much work left.		Deeply analyse your necessity and go forward. You could also ask your boss to provide you a trainee.
My boss is having unrealistic expectations on me	Work Pressure	All bosses can be that way, it happens.
My boss is an asshole		
My boss is being mean		Don't worry too much about work, just focus on your own happiness.
I got dumped.	Relationship Troubles	Oh, that's too bad. But don't worry, there are plenty of other fish in the sea... or soya, if you're vegan ;)
I just broke up with my boyfriend/girlfriend.		Damn... do you want to talk about him/her? About what happened?
I just broke up.		
Why should anyone be in a relationship?	Relationship Troubles	It's always nice to have a person who cares for you, remembers your likes and dislikes and is always there for you.
why do i need a boyfriend/girlfriend?		
Why should i be in a relationship?		Because everyone needs a person to shout, scream, fight, annoy, abuse and finally eat their muffin.

Figure 6.28: Chatbot Testcases

Chapter 7

Conclusion

7.1 Conclusive Statement

From the product being built to the math behind the magic to how Therabot works, we can conclude successfully that we have stepped into a new field with respect to the marriage of mental wellness and artificial intelligence, where we can now use robotic speech and personalized speech generation to cater to the user and create a more lasting bond.

The stigma around depression and getting help from a medical professional is still very high, and we believe that with the ease of access to a personalized robotic therapist in your smartphones/laptops, it should make it much more easier to approach the concept of getting help. Through the course of marketing the idea, we noticed that a lot of people were open about the idea of talking to a non-human person about their feelings as opposed to a stranger.

We believe that with the right amount of question-answer datasets and user input, along with the prolonged reinforcement learning in place, we will be able to build a self-sustaining chatbot that can help at least the majority of people who are in need of a person to talk to.

Chapter 8

Appendix: Code

8.1 Website

./code/website/README.md

```

1 # Therabot - PWA
2
3 A progressive web application written in Angular 6 for Therabot, a
4 stress and anxiety reducing AI agent and application that aims to
5 lessen the impact around depression and related mental illnesses.
6
7 This project was generated with [Angular CLI](https://github.com/angular/angular-cli) version 6.0.1.
8
9 ## Development server
10
11 Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`.
12 The app will automatically reload if you change any of the source
13 files.
14
15 Run `ng generate component component-name` to generate a new component.
16 You can also use `ng generate directive|pipe|service|class|guard|
17 interface|enum|module`.
18
19 Run `ng build` to build the project. The build artifacts will be stored
20 in the `dist/` directory. Use the `--prod` flag for a production
21 build.
22
23 Run `ng test` to execute the unit tests via [Karma](https://karma-runner.github.io).
24
25 ## Running end-to-end tests
26
27 Run `ng e2e` to execute the end-to-end tests via [Protractor](http://www.protractortest.org/).
28
29 ## Further help
30
31 To get more help on the Angular CLI use `ng help` or go check out the [
32 Angular CLI README\]\(https://github.com/angular/angular-cli/blob/master/README.md\).
```

./code/website/package.json

```

1 {
2   "name": "therabot-pwa",
3   "version": "0.0.0",
4   "scripts": {
5     "ng": "ng",
6     "start": "ng serve",
```

```
7      "build": "ng build",
8      "test": "ng test",
9      "lint": "ng lint",
10     "e2e": "ng e2e"
11   },
12   "private": true,
13   "dependencies": {
14     "@angular/animations": "^6.0.2",
15     "@angular/cdk": "^6.0.2",
16     "@angular/common": "^6.0.2",
17     "@angular/compiler": "^6.0.2",
18     "@angular/core": "^6.0.2",
19     "@angular/forms": "^6.0.2",
20     "@angular/http": "^6.0.2",
21     "@angular/material": "^6.0.2",
22     "@angular/platform-browser": "^6.0.2",
23     "@angular/platform-browser-dynamic": "^6.0.2",
24     "@angular/pwa": "^0.6.3",
25     "@angular/router": "^6.0.2",
26     "@angular/service-worker": "^6.0.2",
27     "angularfire2": "^5.0.0-rc.9",
28     "auth0-js": "^9.5.1",
29     "bootstrap": "^4.1.1",
30     "core-js": "^2.5.4",
31     "firebase": "^5.0.2",
32     "font-awesome": "^4.7.0",
33     "hammerjs": "^2.0.8",
34     "jquery": "^3.3.1",
35     "ngx-bootstrap": "^2.0.5",
36     "popper.js": "^1.14.3",
37     "rxjs": "^6.0.0",
38     "rxjs-compat": "^6.1.0",
39     "zone.js": "^0.8.26"
40   },
41   "devDependencies": {
42     "@angular-devkit/build-angular": "^0.6.3",
43     "@angular/cli": "~6.0.3",
44     "@angular/compiler-cli": "^6.0.2",
45     "@angular/language-service": "^6.0.2",
46     "@types/jasmine": "~2.8.6",
47     "@types/jasminewd2": "~2.0.3",
48     "@types/node": "~8.9.4",
49     "codelyzer": "~4.2.1",
50     "jasmine-core": "~2.99.1",
51     "jasmine-spec-reporter": "~4.2.1",
52     "karma": "~1.7.1",
53     "karma-chrome-launcher": "~2.2.0",
54     "karma-coverage-istanbul-reporter": "^1.4.3",
55     "karma-jasmine": "~1.1.1",
56     "karma-jasmine-html-reporter": "^0.2.2",
57     "protractor": "~5.3.0",
58     "ts-node": "~5.0.1",
59     "tslint": "~5.9.1",
60     "typescript": "^2.7.2"
61   }
62 }
```

```
./code/website/angular.json

1  {
2      "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
3      "version": 1,
4      "newProjectRoot": "projects",
5      "projects": {
6          "Therabot-PWA": {
7              "root": "",
8              "sourceRoot": "src",
9              "projectType": "application",
10             "prefix": "app",
11             "schematics": {
12                 "@schematics/angular:component": {
13                     "styleext": "scss"
14                 }
15             },
16             "architect": {
17                 "build": {
18                     "builder": "@angular-devkit/build-angular:browser",
19                     "options": {
20                         "outputPath": "dist",
21                         "index": "src/index.html",
22                         "main": "src/main.ts",
23                         "polyfills": "src/polyfills.ts",
24                         "tsConfig": "src/tsconfig.app.json",
25                         "assets": [
26                             "src/favicon.ico",
27                             "src/assets",
28                             "src/manifest.json"
29                         ],
30                         "styles": [
31                             {
32                                 "input": "node_modules/@angular/material/prebuilt-themes/indigo-pink.css"
33                             },
34                             "src/styles.scss"
35                         ],
36                         "scripts": []
37                     },
38                     "configurations": {
39                         "production": {
40                             "fileReplacements": [
41                             {
42                                 "replace": "src/environments/environment.ts",
43                                 "with": "src/environments/environment.prod.ts"
44                             }
45                         ],
46                         "optimization": true,
47                         "outputHashing": "all",
48                         "sourceMap": false,
49                         "extractCss": true,
50                         "namedChunks": false,
51                         "aot": true,
52                         "extractLicenses": true,
53                         "vendorChunk": false,
54                         "buildOptimizer": true
55                     }
56                 }
57             }
58         }
59     }
60 }
```

```
54          "buildOptimizer": true,
55          "serviceWorker": true
56      }
57  },
58 },
59 "serve": {
60     "builder": "@angular-devkit/build-angular:dev-server",
61     "options": {
62         "browserTarget": "Therabot-PWA:build"
63     },
64     "configurations": {
65         "production": {
66             "browserTarget": "Therabot-PWA:build:production"
67         }
68     }
69 },
70 "extract-i18n": {
71     "builder": "@angular-devkit/build-angular:extract-i18n",
72     "options": {
73         "browserTarget": "Therabot-PWA:build"
74     }
75 },
76 "test": {
77     "builder": "@angular-devkit/build-angular:karma",
78     "options": {
79         "main": "src/test.ts",
80         "polyfills": "src/polyfills.ts",
81         "tsConfig": "src/tsconfig.spec.json",
82         "karmaConfig": "src/karma.conf.js",
83         "styles": [
84             {
85                 "input": "node_modules/@angular/material/prebuilt-themes/
indigo-pink.css"
86             },
87             "src/styles.scss"
88         ],
89         "scripts": [],
90         "assets": [
91             "src/favicon.ico",
92             "src/assets",
93             "src/manifest.json"
94         ]
95     }
96 },
97 "lint": {
98     "builder": "@angular-devkit/build-angular:tslint",
99     "options": {
100         "tsConfig": [
101             "src/tsconfig.app.json",
102             "src/tsconfig.spec.json"
103         ],
104         "exclude": [
105             "**/node_modules/**"
106         ]
107     }
108 }
```

```

109      }
110    },
111    "Therabot-PWA-e2e": {
112      "root": "e2e/",
113      "projectType": "application",
114      "architect": {
115        "e2e": {
116          "builder": "@angular-devkit/build-angular:protractor",
117          "options": {
118            "protractorConfig": "e2e/protractor.conf.js",
119            "devServerTarget": "Therabot-PWA:serve"
120          }
121        },
122        "lint": {
123          "builder": "@angular-devkit/build-angular:tslint",
124          "options": {
125            "tsConfig": "e2e/tsconfig.e2e.json",
126            "exclude": [
127              "**/node_modules/**"
128            ]
129          }
130        }
131      }
132    },
133  },
134  "defaultProject": "Therabot-PWA"
135 }

```

./code/website/ngsw-config.json

```

1  {
2    "index": "/index.html",
3    "assetGroups": [
4      {
5        "name": "app",
6        "installMode": "prefetch",
7        "resources": {
8          "files": [
9            "/favicon.ico",
10           "/index.html",
11           "/*.css",
12           "/*.js"
13         ]
14       }
15     },
16     {
17       "name": "assets",
18       "installMode": "lazy",
19       "updateMode": "prefetch",
20       "resources": {
21         "files": [
22           "/assets/**"
23         ]
24       }
25     }
26   ]
27 }

```

./code/website/src/main.ts

```

1 import { enableProdMode } from '@angular/core';
2 import { platformBrowserDynamic } from '@angular/platform-browser-
    dynamic';
3
4 import { AppModule } from './app/app.module';
5 import { environment } from './environments/environment';
6
7 import 'auth0-js';
8 import 'hammerjs';
9
10 if (environment.production) {
11     enableProdMode();
12 }
13
14 platformBrowserDynamic().bootstrapModule(AppModule)
15     .catch(err => console.log(err));

```

./code/website/src/index.html

```

1 <!doctype html>
2 <html lang="en">
3
4 <head>
5     <meta charset="utf-8">
6     <title>Therabot</title>
7     <base href="/">
8
9     <meta name="viewport" content="width=device-width, initial-scale=1">
10    <link rel="icon" type="image/x-icon" href="favicon.ico">
11
12    <meta name="description" content="A progressive web application
        written in Angular 6 for Therabot, a stress and anxiety reducing AI
        agent and application that aims to lessen the impact around
        depression and related mental illnesses.">
13    <meta name="author" content="Amruth Pillai">
14
15    <meta property="og:title" content="Therabot">
16    <meta property="og:description" content="A progressive web application
        written in Angular 6 for Therabot, a stress and anxiety reducing AI
        agent and application that aims to lessen the impact around
        depression and related mental illnesses.">
17    <meta property="og:image" content="https://therabot.me/assets/og-share
        -image.jpg">
18    <meta property="og:url" content="https://therabot.me/">
19
20    <link href="https://use.fontawesome.com/releases/v5.0.13/css/all.css"
        rel="stylesheet">
21    <link href="https://fonts.googleapis.com/css?family=PT+Sans" rel="
        stylesheet">
22    <link href="https://fonts.googleapis.com/css?family=Product+Sans:100"
        rel="stylesheet">
23
24    <link rel="manifest" href="manifest.json">
25    <meta name="theme-color" content="#2986e6">
26 </head>
27

```

```

28 <body>
29   <app-root></app-root>
30 </body>
31
32 </html>
```

./code/website/src/styles.scss

```

1 /* You can add global styles to this file, and also import other style
   files */
2 @import 'variables';
3 @import '~font-awesome/scss/font-awesome';
4 @import '~bootstrap/scss/bootstrap';
5
6 body {
7   font-family: 'PT Sans';
8 }
9
10 @for $i from 1 through 6 {
11   h#{7 - $i} {
12     font-family: 'Product Sans';
13   }
14 }
15
16 .text-therablue {
17   color: $therablue;
18 }
```

./code/website/src/polyfills.ts

```

1 /**
2  * This file includes polyfills needed by Angular and is loaded before
3  * the app.
4  * You can add your own extra polyfills to this file.
5  *
6  * This file is divided into 2 sections:
7  * 1. Browser polyfills. These are applied before loading ZoneJS and
8  *    are sorted by browsers.
9  * 2. Application imports. Files imported after ZoneJS that should be
10 *    loaded before your main
11 *    file.
12 *
13 * The current setup is for so-called "evergreen" browsers; the last
14 * versions of browsers that
15 * automatically update themselves. This includes Safari >= 10, Chrome
16 * >= 55 (including Opera),
17 * Edge >= 13 on the desktop, and iOS 10 and Chrome on mobile.
18 *
19 * Learn more in https://angular.io/docs/ts/latest/guide/browser-support.html
20 */
21
22 ****
23
24 * BROWSER POLYFILLS
```

```

19  */
20
21 /** IE9, IE10 and IE11 requires all of the following polyfills. */
22 // import 'core-js/es6/symbol';
23 // import 'core-js/es6/object';
24 // import 'core-js/es6/function';
25 // import 'core-js/es6/promise';
26 // import 'core-js/es6/parse-int';
27 // import 'core-js/es6/parse-float';
28 // import 'core-js/es6/number';
29 // import 'core-js/es6/math';
30 // import 'core-js/es6/string';
31 // import 'core-js/es6/date';
32 // import 'core-js/es6/array';
33 // import 'core-js/es6/regexp';
34 // import 'core-js/es6/map';
35 // import 'core-js/es6/weak-map';
36 // import 'core-js/es6/set';
37
38 /** IE10 and IE11 requires the following for NgClass support on SVG
39   elements */
40 // import 'classlist.js'; // Run `npm install --save classlist.js`.
41
42 /**
43  * Evergreen browsers require these.
44  */
45 // Used for reflect-metadata in JIT. If you use AOT (and only Angular
46   decorators), you can remove.
46 import 'core-js/es7/reflect';
47
48 /**
49  * Web Animations `@angular/platform-browser/animations`
50  * Only required if AnimationBuilder is used within the application and
51   using IE/Edge or Safari.
52  * Standard animation support in Angular DOES NOT require any polyfills
53   (as of Angular 6.0).
53 */
54 // import 'web-animations-js'; // Run `npm install --save web-
55   animations-js`.
55
56 /**
57  * By default, zone.js will patch all possible macroTask and DomEvents
58  * user can disable parts of macroTask/DomEvents patch by setting
59   following flags
59 */
60
61 // (window as any).__Zone_disable_requestAnimationFrame = true; //(
62   disable patch requestAnimationFrame
62 // (window as any).__Zone_disable_on_property = true; // disable patch
63   onProperty such as onclick
63 // (window as any).__zone_symbol__BLACK_LISTED_EVENTS = ['scroll', '(
64   mousemove']; // disable patch specified eventNames
64
65 /*

```

```

66  * in IE/Edge developer tools, the addEventListener will also be wrapped
67  * by zone.js
68  */
69 // (window as any).__Zone_enable_cross_context_check = true;
70
71 /*
*****  

72 * Zone JS is required by default for Angular itself.
73 */
74 import 'zone.js/dist/zone'; // Included with Angular CLI.
75
76
77 /*
*****  

78 * APPLICATION IMPORTS
79 */
80

./code/website/src/manifest.json

1 {
2   "name": "Therabot - Your Pocket Therapist",
3   "short_name": "Therabot",
4   "theme_color": "#2986e6",
5   "background_color": "#fafafa",
6   "display": "standalone",
7   "scope": "/",
8   "start_url": "/",
9   "icons": [
10     {
11       "src": "assets/icons/icon-72x72.png",
12       "sizes": "72x72",
13       "type": "image/png"
14     },
15     {
16       "src": "assets/icons/icon-96x96.png",
17       "sizes": "96x96",
18       "type": "image/png"
19     },
20     {
21       "src": "assets/icons/icon-128x128.png",
22       "sizes": "128x128",
23       "type": "image/png"
24     },
25     {
26       "src": "assets/icons/icon-144x144.png",
27       "sizes": "144x144",
28       "type": "image/png"
29     },
30     {
31       "src": "assets/icons/icon-152x152.png",
32       "sizes": "152x152",
33       "type": "image/png"

```

```

34     },
35     {
36       "src": "assets/icons/icon-192x192.png",
37       "sizes": "192x192",
38       "type": "image/png"
39     },
40     {
41       "src": "assets/icons/icon-384x384.png",
42       "sizes": "384x384",
43       "type": "image/png"
44     },
45     {
46       "src": "assets/icons/icon-512x512.png",
47       "sizes": "512x512",
48       "type": "image/png"
49     }
50   ]
51 }

./code/website/src/environments/environment.ts

1 // This file can be replaced during build by using the `fileReplacements
  ` array.
2 // `ng build ---prod` replaces `environment.ts` with `environment.prod.ts`.
3 // The list of file replacements can be found in `angular.json`.
4
5 export const environment = {
6   production: false,
7   appUri: 'http://localhost:4200/',
8   azure: {
9     subscriptionKey: '0fe8f2e6c4c744c2873043e3d14cc5fd',
10    uriBase: 'https://eastasia.api.cognitive.microsoft.com/face/v1.0/
detect?returnFaceAttributes=emotion',
11  },
12  firebase: {
13    apiKey: 'AIzaSyB7fiA_ejcifqTxuu03c04A9cfb6Ds5rTw',
14    authDomain: 'therabot-3bc85.firebaseio.com',
15    databaseURL: 'https://therabot-3bc85.firebaseio.com',
16    projectId: 'therabot-3bc85',
17    storageBucket: 'therabot-3bc85.appspot.com',
18    messagingSenderId: '408349965775'
19  },
20  auth: {
21    clientId: '69CPgotfkX7S3DfA507Xt7pw9KvsnCFN',
22    domain: 'therabot.auth0.com',
23    audience: 'https://therabot.auth0.com/userinfo',
24    redirect: 'http://localhost:4200/callback',
25    scope: 'openid profile email'
26  }
27};
28
29 /*
30  * In development mode, to ignore zone related error stack frames such
  as
31  * `zone.run`, `zoneDelegate.invokeTask` for easier debugging, you can

```

```

32   * import the following file, but please comment it out in production
33   * mode
34   */
35 // import 'zone.js/dist/zone-error'; // Included with Angular CLI.

          ./code/website/src/app/app.module.ts

1 import { BrowserModule } from '@angular/platform-browser';
2 import { HttpClientModule } from '@angular/common/http';
3 import { NgModule } from '@angular/core';
4
5 import { MaterialModule } from './material/material.module';
6 import { BootstrapModule } from './bootstrap/bootstrap.module';
7 import { AppRoutingModule } from './app-routing.module';
8 import { AppComponent } from './app.component';
9
10 import { AuthService } from './services/auth.service';
11 import { environment } from '../environments/environment';
12
13 import { AngularFireModule } from 'angularfire2';
14 import { AngularFirestoreModule } from 'angularfire2/firestore';
15 import { BrowserAnimationsModule } from '@angular/platform-browser/
    animations';
16 import { NavbarComponent } from './navbar/navbar.component';
17 import { HomeComponent } from './home/home.component';
18 import { DashboardComponent } from './dashboard/dashboard.component';
19
20 import { ServiceWorkerModule } from '@angular/service-worker';
21 import { CarouselComponent } from './home/carousel/carousel.component';
22 import { CallbackComponent } from './callback/callback.component';
23 import { ProfileComponent } from './dashboard/profile/profile.component'
    ;
24 import { FooterComponent } from './footer/footer.component';
25 import { DearDiaryComponent } from './dear-diary/dear-diary.component';
26 import { PhotoAlbumComponent } from './photo-album/photo-album.component
    ';
27 import { EmoteFeelingsComponent } from './emote-feelings/emote-feelings.
    component';
28
29 @NgModule({
30   declarations: [
31     AppComponent,
32     NavbarComponent,
33     HomeComponent,
34     DashboardComponent,
35     CarouselComponent,
36     CallbackComponent,
37     ProfileComponent,
38     FooterComponent,
39     DearDiaryComponent,
40     PhotoAlbumComponent,
41     EmoteFeelingsComponent,
42   ],
43   imports: [
44     BrowserModule,

```

```

45     AppRoutingModule,
46     HttpClientModule,
47     MaterialModule,
48     BootstrapModule,
49     AngularFireModule,
50     AngularFireModule.enablePersistence(),
51     AngularFireModule.enablePersistence(),
52     BrowserAnimationsModule,
53     ServiceWorkerModule.register('/ngsw-worker.js', { enabled:
54       environment.production }),
55   ],
56   providers: [AuthService],
57   bootstrap: [AppComponent]
58 })

export class AppModule { }

```

./code/website/src/app/app.component.ts

```

1 import { Component } from '@angular/core';
2 import { AuthService } from './services/auth.service';
3 import { setTheme } from 'ngx-bootstrap/utils';
4
5 @Component({
6   selector: 'app-root',
7   templateUrl: './app.component.html',
8   styleUrls: ['./app.component.scss']
9 })
10 export class AppComponent {
11
12   public profile: any;
13
14   constructor(public auth: AuthService) {
15     setTheme('bs4');
16     this.auth.handleAuthentication();
17   }
18 }

19 }

```

./code/website/src/app/app.component.scss

./code/website/src/app/app.component.html

```

1 <app-navbar></app-navbar>
2
3 <div style="margin-top: 80px;">
4   <router-outlet></router-outlet>
5 </div>
6
7 <app-footer></app-footer>

```

./code/website/src/app/app-routing.module.ts

```

1 import { NgModule } from '@angular/core';
2 import { Routes, RouterModule, CanActivate } from '@angular/router';
3

```

```

4 import { HomeComponent } from './home/home.component';
5 import { DashboardComponent } from './dashboard/dashboard.component';
6 import { DearDiaryComponent } from './dear-diary/dear-diary.component';
7 import { PhotoAlbumComponent } from './photo-album/photo-album.component';
8 import { EmoteFeelingsComponent } from './emote-feelings/emote-feelings.component';
9 import { CallbackComponent } from './callback/callback.component';
10 import { AuthGuard } from './guards/auth.guard';
11
12 const routes: Routes = [
13   {
14     path: 'home',
15     component: HomeComponent
16   },
17   {
18     path: 'dashboard',
19     component: DashboardComponent,
20     canActivate: [AuthGuard]
21   },
22   {
23     path: 'dear-diary',
24     component: DearDiaryComponent,
25     canActivate: [AuthGuard]
26   },
27   {
28     path: 'photo-album',
29     component: PhotoAlbumComponent,
30     canActivate: [AuthGuard]
31   },
32   {
33     path: 'emote-feelings',
34     component: EmoteFeelingsComponent,
35     canActivate: [AuthGuard]
36   },
37   {
38     path: 'callback',
39     component: CallbackComponent
40   },
41   {
42     path: '**',
43     redirectTo: 'home'
44   }
45 ];
46
47 @NgModule({
48   imports: [RouterModule.forRoot(routes)],
49   exports: [RouterModule]
50 })
51 export class AppRoutingModule {}

```

./code/website/src/app/bootstrap/bootstrap.module.ts

```

1 import { NgModule } from '@angular/core';
2 import { CarouselModule } from 'ngx-bootstrap';
3

```

```

4  @NgModule({
5    imports: [
6      CarouselModule.forRoot()
7    ],
8    exports: [
9      CarouselModule
10   ],
11   declarations: []
12 })
13 export class BootstrapModule { }

./code/website/src/app/callback/callback.component.html

1 <div class="container py-5">
2   <div class="row">
3     <div class="col text-center">
4       <mat-spinner class="mx-auto my-5"></mat-spinner>
5       <h2>Loading...</h2>
6     </div>
7   </div>
8 </div>

./code/website/src/app/dashboard/dashboard.component.html

1 <app-profile></app-profile>
2
3 <mat-divider class="container mx-auto my-5"></mat-divider>
4
5 <div class="container">
6   <h3 class="mb-4 text-center">What do you want to do?</h3>
7   <div class="card-columns">
8     <div class="card" [class.border-dark]="mouseOnCard1" (mouseover)="mouseOnCard1=true" (mouseout)="mouseOnCard1=false" routerLink="/dear-diary">
9       <div class="card-body text-center">
10         <i class="fas fa-book text-therablue my-3" style="font-size: 300%;"></i>
11         <h2>Dear, Diary</h2>
12         <p class="small">Write about your day, store your deepest thoughts and feelings and be assured that no one else can ever read your mind.
13       </div>
14     </div>
15   </div>
16   <div class="card" [class.border-dark]="mouseOnCard2" (mouseover)="mouseOnCard2=true" (mouseout)="mouseOnCard2=false" routerLink="/photo-album">
17     <div class="card-body text-center">
18       <i class="fas fa-image text-success my-3" style="font-size: 300%;"></i>
19       <h2>Photo Album</h2>
20       <p class="small">Store your most memorable photos and memories along with a story that makes you feel better and you can always get back to it.</p>
21     </div>
22   </div>
23 </div>
```

```

24      </div>
25      <div class="card" [class.border-dark]="mouseOnCard3" (mouseover)="
26        mouseOnCard3=true" (mouseout)="mouseOnCard3=false" (click)="
27        gotoMessenger()">
28          <div class="card-body text-center">
29            <i class="fas fa-comments text-danger my-3" style="font-size:
30              300%;"></i>
31            <h2>Just Talk</h2>
32            <p class="small">Powered by our intelligent AI chatbot, you can
33              just have a simple conversation with Therabot if you need someone
34              to talk to.</p>
35          </div>
36        </div>
37        <div class="card" [class.border-dark]="mouseOnCard4" (mouseover)="
38        mouseOnCard4=true" (mouseout)="mouseOnCard4=false" routerLink="/
39        emote-feelings">
40          <div class="card-body text-center">
41            <i class="fas fa-smile text-warning my-3" style="font-size:
42              300%;"></i>
43            <h2>Emote Your Feelings</h2>
44            <p class="small">Using our state-of-the-art machine learning
45              model, telling us how you feel has never been easier. Just point
46              your
47                face to the camera and be natural, we'll know.</p>
48          </div>
49        </div>
50      </div>
51    </div>
52  </div>
53
```

./code/website/src/app/dashboard/dashboard.component.ts

```

1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-dashboard',
5   templateUrl: './dashboard.component.html',
6   styleUrls: ['./dashboard.component.scss']
7 })
8 export class DashboardComponent implements OnInit {
9
10   public mouseOnCard1: boolean;
11   public mouseOnCard2: boolean;
12   public mouseOnCard3: boolean;
13   public mouseOnCard4: boolean;
14
15   constructor() { }
16
17   ngOnInit() { }
18
19   public gotoMessenger() {
20     window.open('https://m.me/therabot.me');
21   }
22
23 }
```

./code/website/src/app/dear-diary/dear-diary.component.html

```

1 <div class="container pt-5">
2   <div class="row">
3     <div class="d-flex flex-layout-column justify-content-between col"
4       >
5       <h1>Dear, Diary...</h1>
6       <button mat-button>+ ADD</button>
7     </div>
8   </div>
9
10 <div class="container mt-4">
11   <div class="card-columns">
12
13   <div class="card">
14     
16     <div class="card-body">
17       <h4 class="card-title">It was a fine day...</h4>
18       <p class="card-subtitle mb-3">Lorem ipsum, dolor sit amet
19         consectetur adipisicing elit. Deleniti minima ut fugit magnam
20         ratione aut reiciendis
21           asperiores, est mollitia veritatis aliquid distinctio optio
22             numquam excepturi eligendi? Iusto earum provident inventore.</p>
23             <div class="d-flex flex-layout-column justify-content-between">
24               <p class="small text-muted">14th May 2018</p>
25               <a href="#" class="delete-link text-danger small">Delete</a>
26             </div>
27           </div>
28         </div>
29
30   <div class="card">
31     
33     <div class="card-body">
34       <h4 class="card-title">My first driving experience...</h4>
35       <p class="card-subtitle mb-3">Lorem ipsum, dolor sit amet
36         consectetur adipisicing elit. Deleniti minima ut fugit magnam
37         ratione aut reiciendis
38           asperiores, est mollitia veritatis aliquid distinctio optio
39             numquam excepturi eligendi? Iusto earum provident inventore.</p>
40             <div class="d-flex flex-layout-column justify-content-between">
41               <p class="small text-muted">21st April 2018</p>
42               <a href="#" class="delete-link text-danger small">Delete</a>
43             </div>
44           </div>
45         </div>
46       </div>
47     </div>
48   </div>
49
50   </div>
51 </div>

```

./code/website/src/app/emote-feelings/emote-feelings.component.html

```

1 <div class="container pt-5">
2   <div class="row">
3     <div class="col">

```

```

4      <h1>Emote Your Feelings</h1>
5      <p>Using our state of the art machine learning model, we can
6          easily detect faces and their emotions in a given photo,
7          of upto 25 faces. Use this service for the times when you want
8          to tell Therabot how you're feeling, but you just
9              can't put it to words...</p>
10     <p class="small font-italic">Don't worry, we don't store any of
11         your photos uploaded here. We work on a pure predict and forget
12         policy.</p>
13     </div>
14   </div>
15 </div>
16
17 <div class="container mt-4">
18   <div class="row">
19     <div classclass="row mb-2">
22         <div class="col">
23           <label for="file-upload">
24             <a style="text-decoration: none;" mat-flat-button color="
25 warn">
26               Upload a Picture
27             </a>
28           </label>
29           <input id="file-upload" class="invisible" type="file" #file (
30 change)="reset(); readUrl($event); detectEmotion(file.files);">
31         </div>
32       </div>
33
34     <div class="row">
35       <div class="col">
36         <img class="img-fluid" *ngIf="url" [src]="url">
37       </div>
38     </div>
39
40   </div>
41
42   <div class="col">
43     <div *ngIf="faceData">
44       <p>We detected that there were
45         <span class="font-weight-bold">{{ faceData.length }} face(s)</
46 span> in the photo that you sent to us.</p>
47       <p>We detected that you are showcasing signs of:</p>
48       <h2 class="my-1" *ngFor="let emotion of detectedEmotions">
49         <span class="badge badge-dark">
50           {{ emotion }}
51         </span>
52       </h2>
53     </div>
54   </div>
55 </div>
56 </div>
57
58 </div>
59 </div>
60
61 </div>
62
63 </div>
64
65 </div>
66
67 </div>
68
69 </div>
70
71 </div>
72
73 </div>
74
75 </div>
76
77 </div>
78
79 </div>
80
81 </div>
82
83 </div>
84
85 </div>
86
87 </div>
88
89 </div>
90
91 </div>
92
93 </div>
94
95 </div>
96
97 </div>
98
99 </div>
100
101 </div>
102
103 </div>
104
105 </div>
106
107 </div>
108
109 </div>
110
111 </div>
112
113 </div>
114
115 </div>
116
117 </div>
118
119 </div>
120
121 </div>
122
123 </div>
124
125 </div>
126
127 </div>
128
129 </div>
130
131 </div>
132
133 </div>
134
135 </div>
136
137 </div>
138
139 </div>
140
141 </div>
142
143 </div>
144
145 </div>
146
147 </div>
148
149 </div>
150
151 </div>
152
153 </div>
154
155 </div>
156
157 </div>
158
159 </div>
160
161 </div>
162
163 </div>
164
165 </div>
166
167 </div>
168
169 </div>
170
171 </div>
172
173 </div>
174
175 </div>
176
177 </div>
178
179 </div>
180
181 </div>
182
183 </div>
184
185 </div>
186
187 </div>
188
189 </div>
190
191 </div>
192
193 </div>
194
195 </div>
196
197 </div>
198
199 </div>
200
201 </div>
202
203 </div>
204
205 </div>
206
207 </div>
208
209 </div>
210
211 </div>
212
213 </div>
214
215 </div>
216
217 </div>
218
219 </div>
220
221 </div>
222
223 </div>
224
225 </div>
226
227 </div>
228
229 </div>
230
231 </div>
232
233 </div>
234
235 </div>
236
237 </div>
238
239 </div>
240
241 </div>
242
243 </div>
244
245 </div>
246
247 </div>
248
249 </div>
250
251 </div>
252
253 </div>
254
255 </div>
256
257 </div>
258
259 </div>
260
261 </div>
262
263 </div>
264
265 </div>
266
267 </div>
268
269 </div>
270
271 </div>
272
273 </div>
274
275 </div>
276
277 </div>
278
279 </div>
280
281 </div>
282
283 </div>
284
285 </div>
286
287 </div>
288
289 </div>
290
291 </div>
292
293 </div>
294
295 </div>
296
297 </div>
298
299 </div>
300
301 </div>
302
303 </div>
304
305 </div>
306
307 </div>
308
309 </div>
310
311 </div>
312
313 </div>
314
315 </div>
316
317 </div>
318
319 </div>
320
321 </div>
322
323 </div>
324
325 </div>
326
327 </div>
328
329 </div>
330
331 </div>
332
333 </div>
334
335 </div>
336
337 </div>
338
339 </div>
340
341 </div>
342
343 </div>
344
345 </div>
346
347 </div>
348
349 </div>
350
351 </div>
352
353 </div>
354
355 </div>
356
357 </div>
358
359 </div>
360
361 </div>
362
363 </div>
364
365 </div>
366
367 </div>
368
369 </div>
370
371 </div>
372
373 </div>
374
375 </div>
376
377 </div>
378
379 </div>
380
381 </div>
382
383 </div>
384
385 </div>
386
387 </div>
388
389 </div>
390
391 </div>
392
393 </div>
394
395 </div>
396
397 </div>
398
399 </div>
400
401 </div>
402
403 </div>
404
405 </div>
406
407 </div>
408
409 </div>
410
411 </div>
412
413 </div>
414
415 </div>
416
417 </div>
418
419 </div>
420
421 </div>
422
423 </div>
424
425 </div>
426
427 </div>
428
429 </div>
430
431 </div>
432
433 </div>
434
435 </div>
436
437 </div>
438
439 </div>
440
441 </div>
442
443 </div>
444
445 </div>
446
447 </div>
448
449 </div>
450
451 </div>
452
453 </div>
454
455 </div>
456
457 </div>
458
459 </div>
460
461 </div>
462
463 </div>
464
465 </div>
466
467 </div>
468
469 </div>
470
471 </div>
472
473 </div>
474
475 </div>
476
477 </div>
478
479 </div>
480
481 </div>
482
483 </div>
484
485 </div>
486
487 </div>
488
489 </div>
490
491 </div>
492
493 </div>
494
495 </div>
496
497 </div>
498
499 </div>
500
501 </div>
502
503 </div>
504
505 </div>
506
507 </div>
508
509 </div>
510
511 </div>
512
513 </div>
514
515 </div>
516
517 </div>
518
519 </div>
520
521 </div>
522
523 </div>
524
525 </div>
526
527 </div>
528
529 </div>
530
531 </div>
532
533 </div>
534
535 </div>
536
537 </div>
538
539 </div>
540
541 </div>
542
543 </div>
544
545 </div>
546
547 </div>
548
549 </div>
550
551 </div>
552
553 </div>
554
555 </div>
556
557 </div>
558
559 </div>
560
561 </div>
562
563 </div>
564
565 </div>
566
567 </div>
568
569 </div>
570
571 </div>
572
573 </div>
574
575 </div>
576
577 </div>
578
579 </div>
580
581 </div>
582
583 </div>
584
585 </div>
586
587 </div>
588
589 </div>
590
591 </div>
592
593 </div>
594
595 </div>
596
597 </div>
598
599 </div>
600
601 </div>
602
603 </div>
604
605 </div>
606
607 </div>
608
609 </div>
610
611 </div>
612
613 </div>
614
615 </div>
616
617 </div>
618
619 </div>
620
621 </div>
622
623 </div>
624
625 </div>
626
627 </div>
628
629 </div>
630
631 </div>
632
633 </div>
634
635 </div>
636
637 </div>
638
639 </div>
640
641 </div>
642
643 </div>
644
645 </div>
646
647 </div>
648
649 </div>
650
651 </div>
652
653 </div>
654
655 </div>
656
657 </div>
658
659 </div>
660
661 </div>
662
663 </div>
664
665 </div>
666
667 </div>
668
669 </div>
670
671 </div>
672
673 </div>
674
675 </div>
676
677 </div>
678
679 </div>
680
681 </div>
682
683 </div>
684
685 </div>
686
687 </div>
688
689 </div>
690
691 </div>
692
693 </div>
694
695 </div>
696
697 </div>
698
699 </div>
700
701 </div>
702
703 </div>
704
705 </div>
706
707 </div>
708
709 </div>
710
711 </div>
712
713 </div>
714
715 </div>
716
717 </div>
718
719 </div>
720
721 </div>
722
723 </div>
724
725 </div>
726
727 </div>
728
729 </div>
730
731 </div>
732
733 </div>
734
735 </div>
736
737 </div>
738
739 </div>
740
741 </div>
742
743 </div>
744
745 </div>
746
747 </div>
748
749 </div>
750
751 </div>
752
753 </div>
754
755 </div>
756
757 </div>
758
759 </div>
760
761 </div>
762
763 </div>
764
765 </div>
766
767 </div>
768
769 </div>
770
771 </div>
772
773 </div>
774
775 </div>
776
777 </div>
778
779 </div>
780
781 </div>
782
783 </div>
784
785 </div>
786
787 </div>
788
789 </div>
790
791 </div>
792
793 </div>
794
795 </div>
796
797 </div>
798
799 </div>
800
801 </div>
802
803 </div>
804
805 </div>
806
807 </div>
808
809 </div>
810
811 </div>
812
813 </div>
814
815 </div>
816
817 </div>
818
819 </div>
820
821 </div>
822
823 </div>
824
825 </div>
826
827 </div>
828
829 </div>
830
831 </div>
832
833 </div>
834
835 </div>
836
837 </div>
838
839 </div>
840
841 </div>
842
843 </div>
844
845 </div>
846
847 </div>
848
849 </div>
850
851 </div>
852
853 </div>
854
855 </div>
856
857 </div>
858
859 </div>
860
861 </div>
862
863 </div>
864
865 </div>
866
867 </div>
868
869 </div>
870
871 </div>
872
873 </div>
874
875 </div>
876
877 </div>
878
879 </div>
880
881 </div>
882
883 </div>
884
885 </div>
886
887 </div>
888
889 </div>
890
891 </div>
892
893 </div>
894
895 </div>
896
897 </div>
898
899 </div>
900
901 </div>
902
903 </div>
904
905 </div>
906
907 </div>
908
909 </div>
910
911 </div>
912
913 </div>
914
915 </div>
916
917 </div>
918
919 </div>
920
921 </div>
922
923 </div>
924
925 </div>
926
927 </div>
928
929 </div>
930
931 </div>
932
933 </div>
934
935 </div>
936
937 </div>
938
939 </div>
940
941 </div>
942
943 </div>
944
945 </div>
946
947 </div>
948
949 </div>
950
951 </div>
952
953 </div>
954
955 </div>
956
957 </div>
958
959 </div>
960
961 </div>
962
963 </div>
964
965 </div>
966
967 </div>
968
969 </div>
970
971 </div>
972
973 </div>
974
975 </div>
976
977 </div>
978
979 </div>
980
981 </div>
982
983 </div>
984
985 </div>
986
987 </div>
988
989 </div>
990
991 </div>
992
993 </div>
994
995 </div>
996
997 </div>
998
999 </div>
999 </div>
```

./code/website/src/app/emote-feelings/emote-feelings.component.ts

```

1 import { Component, OnInit } from '@angular/core';
2 import { environment } from '../../environments/environment';
3 import { HttpClient, HttpHeaders } from '@angular/common/http';
4
5 @Component({
6   selector: 'app-emote-feelings',
7   templateUrl: './emote-feelings.component.html',
8   styleUrls: ['./emote-feelings.component.scss']
9 })
10 export class EmoteFeelingsComponent implements OnInit {
11
12   public faceData: Array<any>;
13   public detectedEmotions = [];
14   public url;
15
16   subscriptionKey = environment.azure.subscriptionKey;
17   uriBase = environment.azure.uriBase;
18
19   httpOptions = {
20     headers: new HttpHeaders({
21       'Content-Type': 'application/octet-stream',
22       'Ocp-Apim-Subscription-Key': this.subscriptionKey
23     })
24   };
25
26   constructor(private http: HttpClient) { }
27
28   ngOnInit() { }
29
30   public reset() {
31     this.faceData = null;
32     this.detectedEmotions = [];
33     this.url = null;
34   }
35
36   public detectEmotion(files) {
37     this.http.post<Array<any>>(this.uriBase, files[0], this.httpOptions)
38       .subscribe(data => {
39         this.faceData = data;
40         data.forEach(element => {
41           const e = element.faceAttributes.emotion;
42           this.detectedEmotions.push(Object.keys(e).reduce((a, b) => e[a]
43             ] > e[b] ? a : b));
44         });
45       },
46       error => {
47         console.error(error);
48       });
49
50   public readUrl(ev: any) {
51     if (ev.target.files && ev.target.files[0]) {
52       const reader = new FileReader();
53
54       reader.onload = (event: any) => {
55         this.url = event.target.result;

```

```

56     };
57
58     reader.readAsDataURL(ev.target.files[0]);
59   }
60 }
61
62 }

./code/website/src/app/footer/footer.component.html

1 <div class="container py-5">
2   <div class="row">
3     <div class="col text-center mb-4">
4       <mat-divider class="col-md-6 mx-auto mb-4"></mat-divider>
5       
6       <span class="mx-3 small">
7         <i class="fas fa-heart text-danger"></i>
8       </span>
9       
10      </div>
11    </div>
12    <div class="row">
13      <div class="col text-center">
14        <p class="small">Copyright &copy; 2018
15          <span class="text-therablue">Therabot</span>. All Rights
    Reserved.</p>
16      </div>
17    </div>
18    <div class="row">
19      <div class="col text-center">
20        <button mat-button>
21          <span class="small font-weight-bold">Privacy Policy</span>
22        </button>
23        <button mat-button>
24          <span class="small font-weight-bold">Terms of Use</span>
25        </button>
26        <button mat-button>
27          <span class="small font-weight-bold">Contact Us</span>
28        </button>
29      </div>
30    </div>
31 </div>

```

./code/website/src/app/guards/auth.guard.ts

```

1 import { Injectable } from '@angular/core';
2 import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot,
3   Router } from '@angular/router';
4 import { AuthService } from '../services/auth.service';
5 import { Observable } from 'rxjs';
6
6 @Injectable({
7   providedIn: 'root'
8 })

```

```

9  export class AuthGuard implements CanActivate {
10    constructor(public auth: AuthService, public router: Router) {}
11
12    canActivate(
13      next: ActivatedRouteSnapshot,
14      state: RouterStateSnapshot): Observable<boolean> | Promise<boolean>
15      | boolean {
16        if (!this.auth.isAuthenticated()) {
17          this.router.navigate(['home']);
18          return false;
19        }
20
21        return true;
22      }

```

./code/website/src/app/home/home.component.html

```

1 <div class="container-fluid">
2   <div class="row">
3     <app-carousel class="d-none d-md-block"></app-carousel>
4   </div>
5 </div>
6
7 <div class="container my-5">
8   <div class="row">
9     <div class="col-lg-4 text-center">
10       
11     </div>
12     <div class="col text-center text-lg-left">
13       <h2 class="font-weight-light mb-4">Introduction</h2>
14       <p class="small">
15         This study aims to highlight the use cases of a variety of
16         technologies in the field of Artificial Intelligence, Speech
17         Synthesis,
18         Personalization, User Character Segmentation and Automated Text/
19         Speech-based Chatbots. To illustrate the uses, we
20         have chosen a specific field of expertise, mental wellness. We
21         have built an artificially chatbot that is designed
22         to understand the complexities of 'its user, help the user feel
23         more self-aware and spontaneously check back on the
24         user from time to time. We have also devised a method to extract
25         key likes and dislikes from the 'users messages
26         using NLP techniques.
27       </p>
28       <p class="small">
29         The goal of Therabot as a product to the consumer is not to
30         overthrow the need for real life therapists, but to simply allow
31         the user to feel more comfortable talking about his/her
32         experiences, thus decreasing the stigma that perpetuates
33         around mental health in our country and around the world. We
34         want to make mental wellness more accessible and educate
35         users on the true meaning of depression and anxiety. The
36         objective of our research is to show how chatbots powered

```

```

27      by artificial intelligence can be not only used for service
28      based QA systems, but also in unique niche markets such
29      as medical health, distress systems, education etc. Using our
30      unique methods of generating the 'users responses and
31      creating a schema for each individual user, we are able to bring
32      more personalization and unique feeling to every
33      user.
34      </p>
35      </div>
36  </div>
37
38 <mat-divider class="my-5"></mat-divider>
39
40 <div class="container my-5">
41   <div class="row">
42     <div class="col">
43       <h2 class="mb-4 text-center">Modules</h2>
44       <div class="card-columns">
45         <div class="card">
46           <div class="card-body">
47             <h4><i class="fas fa-book text-therablue pr-1"></i> Journal
48             Writing</h4>
49             <p class="small">Write about your day, store your deepest
50             thoughts and feelings and be assured that no one else can ever read
51             your mind.</p>
52           </div>
53         </div>
54
55         <div class="card">
56           <div class="card-body">
57             <h4><i class="fas fa-image text-success pr-1"></i> Photo
58             Albums</h4>
59             <p class="small">Store your most memorable photos and
60             memories along with a story that makes you feel better and you can
61             always get back to it.</p>
62           </div>
63         </div>
64
65         <div class="card">
66           <div class="card-body">
67             <h4><i class="fas fa-comments text-danger pr-1"></i> Just
68             Talk</h4>
69             <p class="small">Powered by our intelligent AI chatbot, you
70             can just have a simple conversation with Therabot if you need
71             someone to talk to.</p>
72           </div>
73         </div>
74       </div>
75     </div>
76   </div>
77
78 <mat-divider class="my-5"></mat-divider>
79
80 <div class="container my-5">
```

```

71   <div class="row">
72     <div class="col">
73       <h2 class="mb-4">Cognitive Behavioral Therapy</h2>
74       <p class="small">Cognitive behavioral therapy (CBT) is a short-
75         term therapy technique used by counselors and therapists to teach
76         individuals to change their unwanted behaviors by changing their
77         thought patterns. The premise of cognitive behavioral therapy is
78         that our thought patterns (cognition) and interpretations of life
79         events greatly influence how we behave and, ultimately, how we feel
80         .</p>
81       <p class="small">CBT is a form of psychotherapy that focuses on
82         how your thoughts, beliefs and attitudes affect your feelings and
83         behavior. It aims to teach you effective coping strategies for
84         dealing with different problems throughout life. CBT can help you
85         make sense of overwhelming problems by breaking them down into
86         smaller parts.</p>
87       <p class="small">One of the key tenets of CBT is that distorted
88         thinking leads to distress and problematic behaviors, whereas
89         thinking realistically with less negativity allows individuals to
90         respond to challenging life circumstances in an effective way.
91         Research shows this technique is an effective therapy for not only
92         depression and panic disorder, but many illnesses and dysfunctional
93         behaviors.</p>
94     </div>
95     <div class="col-lg-5">
96       
97     </div>
98   </div>
99 </div>

```

./code/website/src/app/home/carousel/carousel.component.html

```

1 <carousel>
2   <slide>
3     
5     <div class="carousel-caption d-none d-md-block">
6       <p class="small">Photo by Green Chameleon</p>
7     </div>
8   </slide>
9   <slide>
10    
12    <div class="carousel-caption d-none d-md-block">
13      <p class="small">Photo by Alex Ivashenko</p>
14    </div>
15  </slide>
16  <slide>
17    
19    <div class="carousel-caption d-none d-md-block">
20      <p class="small">Photo by Kristina Flour</p>

```

```
./code/website/src/app/material/material.module.ts

1 import { NgModule } from '@angular/core';
2
3 import { ScrollDispatchModule } from '@angular/cdk/scrolling';
4 import { CdkTableModule } from '@angular/cdk/table';
5 import { CdkTreeModule } from '@angular/cdk/tree';
6 import {
7   MatAutocompleteModule, MatBadgeModule, MatBottomSheetModule,
8   MatButtonModule,
9   MatButtonToggleModule, MatCardModule, MatCheckboxModule,
10  MatChipsModule, MatDatepickerModule,
11  MatDialogModule, MatDividerModule, MatExpansionModule,
12  MatFormFieldModule, MatGridListModule,
13  MatIconModule, MatInputModule, MatListModule, MatMenuModule,
14  MatPaginatorModule,
15  MatProgressBarModule, MatProgressSpinnerModule, MatRadioModule,
16  MatRippleModule, MatSelectModule,
17  MatSidenavModule, MatSliderModule, MatSlideToggleModule,
18  MatSnackBarModule, MatSortModule,
19  MatStepperModule, MatTableModule, MatTabsModule, MatToolbarModule,
20  MatTooltipModule, MatTreeModule
21 } from '@angular/material';

22 @NgModule({
23   imports: [
24     CdkTableModule,
25     CdkTreeModule,
26     MatAutocompleteModule,
27     MatBadgeModule,
28     MatBottomSheetModule,
29     MatButtonModule,
30     MatButtonToggleModule,
31     MatCardModule,
32     MatCheckboxModule,
33     MatChipsModule,
34     MatDatepickerModule,
35     MatDialogModule,
36     MatDividerModule,
37     MatExpansionModule,
38     MatFormFieldModule,
39     MatGridListModule,
40     MatIconModule,
41     MatInputModule,
42     MatListModule,
43     MatMenuModule,
44     MatPaginatorModule,
45     MatProgressBarModule,
46     MatProgressSpinnerModule,
47     MatRadioModule,
48     MatRippleModule,
49     MatSelectModule,
50     MatSidenavModule,
51     MatSlideToggleModule,
52     MatSliderModule,
53     MatSnackBarModule,
```

```
48      MatSortModule,
49      MatStepperModule,
50      MatTableModule,
51      MatTabsModule,
52      MatToolbarModule,
53      MatTooltipModule,
54      MatTreeModule,
55      ScrollDispatchModule,
56  ],
57  exports: [
58      CdkTableModule,
59      CdkTreeModule,
60      MatAutocompleteModule,
61      MatBadgeModule,
62      MatBottomSheetModule,
63      MatButtonModule,
64      MatButtonToggleModule,
65      MatCardModule,
66      MatCheckboxModule,
67      MatChipsModule,
68      MatDatepickerModule,
69      MatDialogModule,
70      MatDividerModule,
71      MatExpansionModule,
72      MatFormFieldModule,
73      MatGridListModule,
74      MatIconModule,
75      MatInputModule,
76      MatListModule,
77      MatMenuModule,
78      MatPaginatorModule,
79      MatProgressBarModule,
80      MatProgressSpinnerModule,
81      MatRadioModule,
82      MatRippleModule,
83      MatSelectModule,
84      MatSidenavModule,
85      MatSlideToggleModule,
86      MatSliderModule,
87      MatSnackBarModule,
88      MatSortModule,
89      MatStepperModule,
90      MatTableModule,
91      MatTabsModule,
92      MatToolbarModule,
93      MatTooltipModule,
94      MatTreeModule,
95      ScrollDispatchModule,
96  ]
97 })
98 export class MaterialModule { }
```

./code/website/src/app/navbar/navbar.component.html

```
1 <mat-toolbar class="fixed-top navbar">
2   <mat-toolbar-row>
```

```

3      
5      <span class="example-spacer"></span>
6      <button mat-button (click)="auth.login()" *ngIf="!auth.
7          isAuthenticated()">Login</button>
8
9      <button mat-button routerLink="/dashboard" routerLinkActive="active"
10         *ngIf="auth.isAuthenticated()">Dashboard</button>
11     <button mat-button (click)="auth.logout()" *ngIf="auth.
12         isAuthenticated()">Logout</button>
13   </mat-toolbar-row>
14 </mat-toolbar>
```

./code/website/src/app/photo-album/photo-album.component.html

```

1 <div class="container pt-5">
2   <div class="row">
3     <div class="d-flex flex-layout-column justify-content-between col">
4       <h1>Photo Album</h1>
5       <button mat-button>+ ADD</button>
6     </div>
7   </div>
8 </div>
9
10 <div class="container mt-4">
11   <div class="card-columns">
12     <div class="card">
13       
14     </div>
15
16     <div class="card">
17       
18     </div>
19
20     <div class="card">
21       
22     </div>
23
24     <div class="card">
25       
26     </div>
27
28     <div class="card">
29       
30     </div>
31
32     <div class="card">
33       
34     </div>
35   </div>
36 </div>
```

./code/website/src/app/services/auth.service.ts

```

1 import { Injectable } from '@angular/core';
2 import * as auth0 from 'auth0-js';
```

```
3 import { Router } from '@angular/router';
4 import { environment } from '../../environments/environment';
5
6 (window as any).global = window;
7
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class AuthService {
12
13   userProfile: any;
14   accessToken: string;
15   authenticated: boolean;
16
17   auth0 = new auth0.WebAuth({
18     clientID: environment.auth.clientID,
19     domain: environment.auth.domain,
20     responseType: 'token id_token',
21     audience: environment.auth.audience,
22     redirectUri: environment.auth.redirect,
23     scope: environment.auth.scope
24   });
25
26   constructor(public router: Router) {
27     // Check session to restore login if not expired
28     if (Date.now() < JSON.parse(localStorage.getItem('expires_at'))) {
29       this.handleAuthentication();
30     }
31
32     const accessToken = localStorage.getItem('access_token');
33     if (accessToken && !this.userProfile) {
34       this.getProfile((err, profile) => {
35         this.userProfile = profile;
36       });
37     }
38   }
39
40   public login(): void {
41     this.auth0.authorize();
42   }
43
44   public handleAuthentication(): void {
45     this.auth0.parseHash((err, authResult) => {
46       if (authResult && authResult.accessToken) {
47         window.location.hash = '';
48         this.getUserProfile(authResult);
49       } else if (err) {
50         this.router.navigate(['home']);
51         console.log(err);
52       }
53     });
54   }
55
56   private getUserProfile(authResult): void {
57     this.auth0.client.userInfo(authResult.accessToken, (err, profile) =>
58     {
```

```
58     if (profile) {
59         this.setSession(authResult, profile);
60     }
61 });
62 }
63
64 private setSession(authResult, profile): void {
65     // Set the time that the Access Token will expire at
66     const expiresAt = JSON.stringify((authResult.expiresIn * 1000) + new
67     Date().getTime());
68     localStorage.setItem('access_token', authResult.accessToken);
69     localStorage.setItem('expires_at', expiresAt);
70
71     this.accessToken = authResult.accessToken;
72     this.userProfile = profile;
73     this.authenticated = true;
74
75     this.router.navigate(['dashboard']);
76 }
77
78 public logout(): void {
79     // Remove tokens and expiry time from localStorage
80     localStorage.removeItem('access_token');
81     localStorage.removeItem('expires_at');
82
83     this.userProfile = undefined;
84     this.accessToken = undefined;
85     this.authenticated = false;
86
87     // Go back to the home route
88     this.router.navigate(['home']);
89 }
90
91 public isAuthenticated(): boolean {
92     // Check whether the current time is past the
93     // Access Token's expiry time
94     const expiresAt = JSON.parse(localStorage.getItem('expires_at'));
95     return new Date().getTime() < expiresAt;
96 }
97
98 public getProfile(cb): void {
99     const accessToken = localStorage.getItem('access_token');
100
101     if (!accessToken) {
102         throw new Error('Access Token must exist to fetch profile');
103     }
104
105     const self = this;
106     this.auth0.client.userInfo(accessToken, (err, profile) => {
107         if (profile) {
108             self.userProfile = profile;
109         }
110         cb(err, profile);
111     });
112 }
```

113 }

8.2 Chatbot

./code/chatbot/README.md

```

1 # Therabot
2 Therabot is a therapeutic chat-bot that helps users to overcome mental
   issues like depression, loneliness, PTSD or even procrastination.
```

./code/chatbot/package.json

```

1 {
2   "name": "therabot",
3   "version": "1.0.0",
4   "description": "Therabot is a therapeutic chat-bot that helps users to
      overcome mental issues like depression, loneliness, PTSD or even
      procrastination.",
5   "main": "index.js",
6   "dependencies": {
7     "apiai": "^4.0.3",
8     "body-parser": "^1.18.2",
9     "dotenv": "^4.0.0",
10    "express": "^4.16.3",
11    "firebase-admin": "^5.11.0",
12    "nodemon": "^1.17.2",
13    "npm": "^5.8.0",
14    "request": "^2.85.0",
15    "serve-index": "^1.9.1"
16  },
17  "devDependencies": {
18    "eslint": "^4.19.0",
19    "eslint-config-standard": "^11.0.0",
20    "eslint-plugin-import": "^2.9.0",
21    "eslint-plugin-node": "^6.0.1",
22    "eslint-plugin-promise": "^3.7.0",
23    "eslint-plugin-standard": "^3.0.1"
24  },
25  "scripts": {
26    "start": "node index.js",
27    "test": "echo \\"$Error: no test specified\\" && exit 1"
28  },
29  "repository": {
30    "type": "git",
31    "url": "git+https://github.com/AmruthPillai/Therabot.git"
32  },
33  "author": "im.amruth@gmail.com",
34  "license": "ISC",
35  "bugs": {
36    "url": "https://github.com/AmruthPillai/Therabot/issues"
37  },
38  "homepage": "https://github.com/AmruthPillai/Therabot#readme"
39 }
```

./code/chatbot/index.js

```

1 // Requirements
2 const express = require('express')
3 const serveIndex = require('serve-index')
4 const bodyParser = require('body-parser')
5 require('dotenv').config()
6 const app = express()
7
8 // Variables
9 const port = process.env.PORT || 5000
10
11 // Parse application/x-www-form-urlencoded
12 app.use(bodyParser.urlencoded({ 'extended': 'true' }))
13
14 // Parse application/json
15 app.use(bodyParser.json())
16
17 // Parse application/vnd.api+json as json
18 app.use(bodyParser.json({ type: 'application/vnd.api+json' }))
19
20 // Serve static assets
21 app.use('/assets', express.static('assets'), serveIndex('assets', { 'icons': true}))
22
23 // Firebase
24 let firebase = require('../services.firebaseio.service')
25 firebase.initializeApp()
26
27 // Routes
28 require('../routes.js')(app)
29
30 app.listen(port, () => console.log('Webhook server is listening on port
    ' + port))

```

./code/chatbot/routes.js

```

1 const messageWebhookController = require('../controllers/messageWebhook')
2
3 module.exports = function (app) {
4     app.get('/', function (req, res) {
5         res.send('You must point your chatbot to the /webhook route.')
6     })
7
8     app.get('/webhook', function (req, res) {
9         res.send('You must POST your request.')
10    })
11
12    app.post('/webhook', messageWebhookController)
13 }

```

./code/chatbot/controllers/conversationsController.js

```

1 // Conversations
2 const recordJournal = require('../conversations/recordJournal')
3 const randomVideo = require('../conversations/randomVideo')
4 const phq9 = require('../conversations/phq9')
5

```

```

6 module.exports = {
7   recordJournal,
8   randomVideo,
9   phq9
10 }

./code/chatbot/controllers/messageWebhook.js

1 // To Update User Information
2 const UserService = require('../services/user.service')
3
4 // Conversations
5 const conversations = require('./conversationsController')
6
7 module.exports = (req, res) => {
8   // Invoked Action
9   console.log('Invoked Action: ' + req.body.queryResult.action)
10
11  // Collect Essential Request Data
12  let request = {
13    source: req.body.originalDetectIntentRequest,
14    queryText: req.body.queryResult.queryText,
15    action: req.body.queryResult.action,
16    fulfillmentText: req.body.queryResult.fulfillmentText,
17    fulfillmentMessages: req.body.queryResult.fulfillmentMessages,
18    parameters: req.body.queryResult.parameters,
19    allRequiredParamsPresent: req.body.queryResult.
20      allRequiredParamsPresent,
21    session: req.body.session
22  }
23
24  // Source: Google Assistant
25  if (request.source.source === 'google') {
26    request.userId = req.body.originalDetectIntentRequest.payload.user.
27      userId
28
29
30  // Source: Facebook
31  if (request.source.payload.source === 'facebook') {
32    request.userId = req.body.originalDetectIntentRequest.payload.data.
33      sender.id
34
35    UserService.saveFacebookUser(request.userId)
36  }
37
38  switch (request.action) {
39    case 'reduce_stress.watch_videos':
40      conversations.randomVideo.sendRandomYoutubeVideo(res)
41      break
42
43    case 'record_journal.get_energy_level.get_journal_entry':
44      request.energyLevel = req.body.queryResult.outputContexts[0].
        parameters.energy_level

```

```
45     conversations.recordJournal.storeJournalEntry(request.userId,
46         request.energyLevel, request.queryText)
47         .then(() => res.send(request.fulfillmentText).end())
48     break
49
50     case 'phq9.question1':
51         conversations.phq9.storeAnswer(request.userId, 1, request.
52             parameters['phq-answers'])
53         .then(() => res.send({ fulfillmentMessages: request.
54             fulfillmentMessages }).end())
55
56     case 'phq9.question2':
57         conversations.phq9.storeAnswer(request.userId, 2, request.
58             parameters['phq-answers'])
59         .then(() => res.send({ fulfillmentMessages: request.
60             fulfillmentMessages }).end())
61
62     case 'phq9.question3':
63         conversations.phq9.storeAnswer(request.userId, 3, request.
64             parameters['phq-answers'])
65         .then(() => res.send({ fulfillmentMessages: request.
66             fulfillmentMessages }).end())
67
68     case 'phq9.question4':
69         conversations.phq9.storeAnswer(request.userId, 4, request.
70             parameters['phq-answers'])
71         .then(() => res.send({ fulfillmentMessages: request.
72             fulfillmentMessages }).end())
73
74     case 'phq9.question5':
75         conversations.phq9.storeAnswer(request.userId, 5, request.
76             parameters['phq-answers'])
77         .then(() => res.send({ fulfillmentMessages: request.
78             fulfillmentMessages }).end())
79
80     case 'phq9.question6':
81         conversations.phq9.storeAnswer(request.userId, 6, request.
82             parameters['phq-answers'])
83         .then(() => res.send({ fulfillmentMessages: request.
84             fulfillmentMessages }).end())
85
86     case 'phq9.question7':
```

```

87     conversations.phq9.storeAnswer(request.userId, 7, request.
parameters['phq-answers'])
88     .then(() => res.send({ fulfillmentMessages: request.
fulfillmentMessages }).end())
89
90     break
91
92 case 'phq9.question8':
93     conversations.phq9.storeAnswer(request.userId, 8, request.
parameters['phq-answers'])
94     .then(() => res.send({ fulfillmentMessages: request.
fulfillmentMessages }).end())
95
96     break
97
98 case 'phq9.question9':
99     conversations.phq9.storeAnswer(request.userId, 9, request.
parameters['phq-answers'])
100    .then(() => {
101        conversations.phq9.calcPhq9Score(request.userId)
102        .then((score) => {
103            res.send({ fulfillmentText: 'You scored ' + score + ' out
of 27. Thank you for taking the test, this helps me understand you
much better!' })
104        })
105    })
106
107    break
108 }
109 }
```

./code/chatbot/conversations/phq9.js

```

1 const firebase = require('../services.firebaseio.service')
2
3 module.exports = {
4   storeAnswer,
5   calcPhq9Score
6 }
7
8 function storeAnswer (userId, questionNumber, answer) {
9   return new Promise(function (resolve, reject) {
10     var phq9Ref = firebase.getDB().collection('users').doc(userId).
collection('phq9')
11
12     phq9Ref.doc(questionNumber.toString()).get().then(doc => {
13       phq9Ref.doc(questionNumber.toString()).set({
14         answer: answer,
15         updatedAt: firebase.getFieldValue().serverTimestamp()
16       }).then(() => resolve())
17     })
18   })
19 }
20
21 function calcPhq9Score (userId) {
22   return new Promise(function (resolve, reject) {
```

```
23     var phq9Ref = firebase.getDB().collection('users').doc(userId).  
24         collection('phq9')  
25  
26     let phq9Score = 0  
27     let questionNumbers = ['1', '2', '3', '4', '5', '6', '7', '8', '9']  
28     questionNumbers.forEach(questionNumber => {  
29         getPhq9Score(phq9Ref, questionNumber)  
30             .then(answer => {  
31                 phq9Score += answer  
32  
33                 if (questionNumber === '9') {  
34                     phq9Ref.doc('total').set({  
35                         totalScore: phq9Score,  
36                         updatedAt: firebase.getFieldValue().serverTimestamp()  
37                     })  
38  
39                     return resolve(phq9Score)  
40                 }  
41             })  
42         })  
43     })  
44  
45     function getPhq9Score (phq9Ref, questionNumber) {  
46         return new Promise(function (resolve, reject) {  
47             phq9Ref.doc(questionNumber).get().then(doc => {  
48                 return resolve(parseInt(doc.data().answer))  
49             })  
50         })  
51     }  
52 }
```

./code/chatbot/conversations/randomVideo.js

```
1 module.exports = {  
2     sendRandomYoutubeVideo  
3 }  
4  
5 function sendRandomYoutubeVideo (res) {  
6     return new Promise(function (resolve, reject) {  
7         let randomFunnyVideo = funnyVideos[Math.floor(Math.random() *  
funnyVideos.length)]  
8         res.send({  
9             'payload': {  
10                 'google': {  
11                     'richResponse': {  
12                         'items': [  
13                             {  
14                                 'simpleResponse': {  
15                                     'textToSpeech': 'Here\'s a video I think you might  
like...'  
16                                 }  
17                             },  
18                             {  
19                                 'basicCard': {  
20                                     'title': 'Suggested Video',  
21                                     'image': {  
22                                         'url': randomFunnyVideo  
23                                     }  
24                                 }  
25                             }  
26                         }  
27                     }  
28                 }  
29             }  
30         })  
31     })  
32 }
```

```

22             'url': 'https://img.youtube.com/vi/' +
randomFunnyVideo + '/0.jpg'
23         },
24         'buttons': [
25             {
26                 'title': 'https://www.youtube.com/watch?v=' +
randomFunnyVideo,
27                 'openUrlAction': {
28                     'url': 'https://www.youtube.com/watch?v=' +
randomFunnyVideo
29                         }
30                     }
31                 ]
32             }
33         ]
34     }
35   ],
36   'facebook': [
37     { 'text': 'Alright, here\'s one that\'s pretty funny...' },
38     {
39       'attachment': {
40         'type': 'template',
41         'payload': {
42             'template_type': 'open_graph',
43             'elements': [
44               {
45                 'url': funnyVideos[Math.floor(Math.random() *
funnyVideos.length)]
46               }
47             ]
48           }
49         }
50       }
51     ],
52     {
53       'text': 'How about one more? ',
54       'quick_replies': [
55         {
56           'content_type': 'text',
57           'title': ' Yes',
58           'payload': 'Watch another video'
59         },
60         {
61           'content_type': 'text',
62           'title': ' No',
63           'payload': 'NO'
64         }
65       ]
66     }
67   ]
68 }
69 }).end()
70 })
71 }
72
73 let funnyVideos = [

```

```

74     'M1dj019aSFQ',
75     'iPW75Z04pIA',
76     'aEzzLXBH3rU',
77     '9KZComfMmGo',
78     'GIc3aFVv6-E'
79 ]

```

./code/chatbot/conversations/recordJournal.js

```

1 const firebase = require('../services.firebaseio.service')
2
3 module.exports = {
4   storeJournalEntry
5 }
6
7 function storeJournalEntry (userId, energyLevel, journalEntry) {
8   return new Promise(function (resolve, reject) {
9     var journalsRef = firebase.getDB().collection('users').doc(userId).
10       collection('journals')
11
12     journalsRef.add({
13       energyLevel,
14       journalEntry
15     }).then(function (docRef) {
16       docRef.update({ updatedAt: firebase.getFieldValue().
17         serverTimestamp() })
18       resolve()
19     })
20   })
21 }
22
23 }

```

./code/chatbot/curls/persistent_menu.curl

```

1 curl -X POST -H "Content-Type: application/json" -d '{
2   "persistent_menu": [
3     {
4       "locale": "default",
5       "composer_input_disabled": false,
6       "call_to_actions": [
7         {
8           "type": "postback",
9           "title": " Toolbox",
10          "payload": "TOOLBOX"
11        }
12      ]
13    }
14  }' "https://graph.facebook.com/v2.6/me/messenger_profile?access_token=
15   EAAFVg3DgeZBMBAKMmqLxhYuiyo6F19SSecbbF3sZA9ZCjEIcs7s3PU4gVuerqaKZAVuoY4DzfQ3gYwAf
16   "

```

./code/chatbot/services.firebaseio.service.js

```

1 const admin = require('firebase-admin')
2 const FieldValue = admin.firestore.FieldValue
3 const serviceAccount = require('./Therabot-ece122d5ba5b.json')
4
5 module.exports = {

```

```

6     getDB ,
7     getFieldValue ,
8     initializeApp
9 }
10
11 let db
12
13 function initializeApp () {
14   admin.initializeApp({
15     credential: admin.credential.cert(serviceAccount)
16   })
17
18   db = admin.firestore()
19 }
20
21 function getFieldValue () {
22   return FieldValue
23 }
24
25 function getDB () {
26   return db
27 }

./code/chatbot/services/user.service.js

1 const request = require('request')
2
3 const firebase = require('./firebase.service')
4
5 module.exports = {
6   saveGoogleUser,
7   saveFacebookUser,
8   getUser
9 }
10
11 function saveGoogleUser (userId) {
12   return new Promise((resolve, reject) => {
13     let user = {
14       source: 'google',
15       createdAt: firebase.getFieldValue().serverTimestamp()
16     }
17
18     var userProfileDoc = firebase.getDB().collection('users').doc(userId)
19     userProfileDoc.get().then(doc => {
20       if (!doc.exists) {
21         userProfileDoc.set(user)
22       }
23     })
24   })
25 }
26
27 function saveFacebookUser (userId, firstName, lastName) {
28   return new Promise((resolve, reject) => {
29     getFacebookData(userId, function (err, userData) {
30       if (err) { throw new Error(err) }

```

```

31
32     let user = {
33         first_name: userData.first_name,
34         last_name: userData.last_name,
35         profile_pic: userData.profile_pic,
36         gender: userData.gender,
37         locale: userData.locale,
38         source: 'facebook',
39         createdAt: firebase.getFieldValue().serverTimestamp()
40     }
41
42     var userProfileDoc = firebase.getDB().collection('users').doc(
43         userId)
44     userProfileDoc.get().then(doc => {
45         if (!doc.exists) {
46             userProfileDoc.set(user)
47         }
48     })
49 }
50 }
51
52 function getUser (userId) {
53     return new Promise((resolve, reject) => {
54         var userProfileDoc = firebase.getDB().collection('users').doc(userId)
55         userProfileDoc.get().then(doc => {
56             if (doc.exists) {
57                 return resolve(doc.data())
58             }
59         })
60     })
61 }
62
63 function getFacebookData (facebookId, callback) {
64     request({
65         method: 'GET',
66         url: 'https://graph.facebook.com/v2.8/' + facebookId,
67         qs: { access_token: process.env.FB_PAGE_ACCESS_TOKEN }
68     }, function (err, response, body) {
69         if (err) { throw new Error(err) }
70         let userData = JSON.parse(response.body)
71
72         callback(err, userData)
73     })
74 }

```

8.3 Machine Learning

./code/machine-learning/model.py

```

1 # things we need for NLP
2 import nltk
3 from nltk.stem.lancaster import LancasterStemmer
4 stemmer = LancasterStemmer()

```

```

5
6 # things we need for Tensorflow
7 import numpy as np
8 import tflearn
9 import tensorflow as tf
10 import random
11
12 # import our chat-bot intents file
13 import json
14 with open('intents.json') as json_data:
15     intents = json.load(json_data)
16
17 words = []
18 classes = []
19 documents = []
20 ignore_words = ['?']
21 # loop through each sentence in our intents patterns
22 for intent in intents['intents']:
23     for pattern in intent['patterns']:
24         # tokenize each word in the sentence
25         w = nltk.word_tokenize(pattern)
26         # add to our words list
27         words.extend(w)
28         # add to documents in our corpus
29         documents.append((w, intent['tag']))
30         # add to our classes list
31         if intent['tag'] not in classes:
32             classes.append(intent['tag'])
33
34 # stem and lower each word and remove duplicates
35 words = [stemmer.stem(w.lower()) for w in words if w not in ignore_words]
36 words = sorted(list(set(words)))
37
38 # remove duplicates
39 classes = sorted(list(set(classes)))
40
41 print (len(documents), "documents")
42 print (len(classes), "classes", classes)
43 print (len(words), "unique stemmed words", words)
44
45 # create our training data
46 training = []
47 output = []
48 # create an empty array for our output
49 output_empty = [0] * len(classes)
50
51 # training set, bag of words for each sentence
52 for doc in documents:
53     # initialize our bag of words
54     bag = []
55     # list of tokenized words for the pattern
56     pattern_words = doc[0]
57     # stem each word
58     pattern_words = [stemmer.stem(word.lower()) for word in
pattern_words]
```

```

59     # create our bag of words array
60     for w in words:
61         bag.append(1) if w in pattern_words else bag.append(0)
62
63     # output is a '0' for each tag and '1' for current tag
64     output_row = list(output_empty)
65     output_row[classes.index(doc[1])] = 1
66
67     training.append([bag, output_row])
68
69 # shuffle our features and turn into np.array
70 random.shuffle(training)
71 training = np.array(training)
72
73 # create train and test lists
74 train_x = list(training[:,0])
75 train_y = list(training[:,1])
76
77 # reset underlying graph data
78 tf.reset_default_graph()
79 # Build neural network
80 net = tflearn.input_data(shape=[None, len(train_x[0])])
81 net = tflearn.fully_connected(net, 8)
82 net = tflearn.fully_connected(net, 8)
83 net = tflearn.fully_connected(net, len(train_y[0]), activation='softmax')
84 net = tflearn.regression(net)
85
86 # Define model and setup tensorboard
87 model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
88 # Start training (apply gradient descent algorithm)
89 model.fit(train_x, train_y, n_epoch=1000, batch_size=8, show_metric=True)
90 model.save('model.tflearn')
91
92 def clean_up_sentence(sentence):
93     # tokenize the pattern
94     sentence_words = nltk.word_tokenize(sentence)
95     # stem each word
96     sentence_words = [stemmer.stem(word.lower()) for word in sentence_words]
97     return sentence_words
98
99 # return bag of words array: 0 or 1 for each word in the bag that exists
100 # in the sentence
100 def bow(sentence, words, show_details=False):
101     # tokenize the pattern
102     sentence_words = clean_up_sentence(sentence)
103     # bag of words
104     bag = [0]*len(words)
105     for s in sentence_words:
106         for i,w in enumerate(words):
107             if w == s:
108                 bag[i] = 1
109                 if show_details:
110                     print ("found in bag: %s" % w)

```

```

111
112     return(np.array(bag))
113
114 p = bow("is your shop open today?", words)
115 print (p)
116 print (classes)
117
118 print(model.predict([p]))
119
120 # save all of our data structures
121 import pickle
122 pickle.dump( {'words':words, 'classes':classes, 'train_x':train_x, 'train_y':train_y}, open( "training_data", "wb" ) )

```

./code/machine-learning/response.py

```

1 # things we need for NLP
2 import nltk
3 from nltk.stem.lancaster import LancasterStemmer
4 stemmer = LancasterStemmer()
5
6 # things we need for Tensorflow
7 import numpy as np
8 import tflearn
9 import tensorflow as tf
10 import random
11
12 # restore all of our data structures
13 import pickle
14 data = pickle.load( open( "training_data", "rb" ) )
15 words = data['words']
16 classes = data['classes']
17 train_x = data['train_x']
18 train_y = data['train_y']
19
20 # import our chat-bot intents file
21 import json
22 with open('intents.json') as json_data:
23     intents = json.load(json_data)
24
25 # Build neural network
26 net = tflearn.input_data(shape=[None, len(train_x[0])])
27 net = tflearn.fully_connected(net, 8)
28 net = tflearn.fully_connected(net, 8)
29 net = tflearn.fully_connected(net, len(train_y[0]), activation='softmax')
30 net = tflearn.regression(net)
31
32 # Define model and setup tensorboard
33 model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
34
35 def clean_up_sentence(sentence):
36     # tokenize the pattern
37     sentence_words = nltk.word_tokenize(sentence)
38     # stem each word

```

```

39     sentence_words = [stemmer.stem(word.lower()) for word in
40     sentence_words]
41
42 # return bag of words array: 0 or 1 for each word in the bag that exists
43 # in the sentence
44 def bow(sentence, words, show_details=False):
45     # tokenize the pattern
46     sentence_words = clean_up_sentence(sentence)
47     # bag of words
48     bag = [0]*len(words)
49     for s in sentence_words:
50         for i,w in enumerate(words):
51             if w == s:
52                 bag[i] = 1
53             if show_details:
54                 print ("found in bag: %s" % w)
55
56
57 p = bow("is your shop open today?", words)
58 print (p)
59 print (classes)
60
61 # load our saved model
62 model.load('./model.tflearn')
63
64 # create a data structure to hold user context
65 context = {}
66
67 ERROR_THRESHOLD = 0.25
68 def classify(sentence):
69     # generate probabilities from the model
70     results = model.predict([bow(sentence, words)])[0]
71     # filter out predictions below a threshold
72     results = [[i,r] for i,r in enumerate(results) if r>ERROR_THRESHOLD]
73     # sort by strength of probability
74     results.sort(key=lambda x: x[1], reverse=True)
75     return_list = []
76     for r in results:
77         return_list.append((classes[r[0]], r[1]))
78     # return tuple of intent and probability
79     return return_list
80
81 def response(sentence, userID='123', show_details=False):
82     results = classify(sentence)
83     # if we have a classification then find the matching intent tag
84     if results:
85         # loop as long as there are matches to process
86         while results:
87             for i in intents['intents']:
88                 # find a tag matching the first result
89                 if i['tag'] == results[0][0]:
90                     # set context for this intent if necessary
91                     if 'context_set' in i:

```

```

92             if show_details: print ('context:', i['
93                 context_set'])
94
95             # check if this intent is contextual and applies to
96             # this user's conversation
97             if not 'context_filter' in i or \
98                 (userID in context and 'context_filter' in i and
99                 i['context_filter'] == context(userID)):
100                 if show_details: print ('tag:', i['tag'])
101                 # a random response from the intent
102                 return print(random.choice(i['responses']))
103
104     results.pop(0)
105
106 classify('is your shop open today?')
107 response('is your shop open today?')

```

./code/machine-learning/simple/firebase.py

```

1 import firebase_admin
2 from firebase_admin import credentials
3 from firebase_admin import firestore
4
5 # Use a service account
6 cred = credentials.Certificate('Therabot-ece122d5ba5b.json')
7 #firebase_admin.initialize_app(cred)
8
9 db = firestore.client()
10
11 doc_ref = db.collection(u'users')
12
13 '''
14 .collection('dislikes').document('star')
15
16 data = {
17     u'checked': u'no',
18     u'score': sentiment_score,
19     u'type': sentiment_type
20 }
21 doc_ref.set(data, firestore.CreateIfMissingOption(True))
22 '''

```

./code/machine-learning/simple/sentiment-score.py

```

1 #Sentiment Analysis
2 '''
3 1. Calculate the sentiment score of the line of text(POS Tagger)
4 2. Extract the words of interest
5
6 '''
7 import nltk
8 import firebase_initialize
9 from nltk.tokenize import word_tokenize
10 from nltk.sentiment.vader import SentimentIntensityAnalyzer
11

```

```

12 str1 = "i WAS appreciated for my creative nlp project"
13 str7 = "i like ppts, i AM not unhappy with the presentation"
14 str2 = "I like Dulqueer Salmaan"
15 str3 = "i LIKE YELLOW FRUITS"
16 str4 = 'The incredibly intimidating NLP scares people away'
17 str5 = "i like the purple color of your wall"
18 category_list, phrases = set([]), set([])
19
20 #Begin - Sentiment and Keyword Extract Function
21 def sentiment_extract(chosen_str, userId):
22     sid = SentimentIntensityAnalyzer()
23     print(chosen_str)
24     ss = sid.polarity_scores(chosen_str)
25     '''
26     param k: neg/neu/pos/compound
27     param ss[k]: score
28     '''
29     #Max of the 3 sentiments
30     positive = ss['pos']
31     negative = ss['neg']
32     neutral = ss['neu']
33     compound = ss['compound']
34
35     if positive>negative:
36         sentiment_score = positive
37         sentiment_type = 'positive'
38     else:
39         sentiment_score = negative
40         sentiment_type = 'negative'
41
42     #for k in ss:
43     #    print('{0}: {1}, '.format(k, ss[k]), end='')
44
45     print(sentiment_score, sentiment_type)
46
47     text = word_tokenize(chosen_str.lower())
48     data = nltk.pos_tag(text)
49     print(data)
50
51     list_tagger = ['NN', 'VB', 'NNS', 'JJ']
52     list_neglect = ['i', 'for', 'with', 'like', 'hate', 'favorite', 'horrible', 'unhappy']
53     list_neighbor_tags = ['JJ', 'NN', 'NNS']
54     pp_word, pp_tag, prev_tag, prev_word, count, prev_count = None, None, None, None, 0, 0
55
56     each_phrase = ''
57     len=0
58     for word,tag in data:
59         count = count+1
60         if word in list_happy or word in list_sad or word in list_likes
61             or word in list_dislikes or word in list_fears:
62                 category_list.add(word)
63                 if tag in list_tagger and word not in list_neglect:
64                     #Combine two related words
65                     if prev_tag in list_neighbor_tags and count-prev_count <= 1:

```

```

65             if(len==0):
66                 each_phrase = each_phrase + prev_word + ' ' + word +
67
68             len=1
69         else:
70             each_phrase = each_phrase + ' ' + word + ' '
71             #print(prev_word, word, count, prev_count)
72             #print(each_phrase)
73             phrases.add(each_phrase)
74         else:
75             if prev_count is not 0:
76                 #phrases.add(each_phrase)
77                 phrases.add(word)
78                 #print(word, count, prev_count)
79             prev_tag, prev_word, prev_count = tag, word, count
80     push_data_to_db(sentiment_score, sentiment_type)
81 #End - Sentiment Extract Function
82
83 list_happy = ['pleasure', 'delight', 'joy', 'happy']
84 list_likes = ['adore', 'like']
85 list_dislikes = ['dislike', 'hate', 'bad']
86 list_fears = ['scary', 'scares', 'afraid']
87 list_sad = ['sorrow', 'cry']
88
89 #Begin - Push to firebase
90 def push_data_to_db(sentiment_score, sentiment_type):
91     u = 'u'
92
93     for w in category_list:
94         for p in phrases:
95             data = {
96                 p: {
97                     u'checked': u'no',
98                     u'score': sentiment_score,
99                     u'type': sentiment_type
100                }
101            new_doc_ref = doc_ref.document(userId).collection(u'keywords')
102            .document(w)
103            new_doc_ref.update(data, firestore.CreateIfMissingOption(
104                True))
105
106 userId = '1526158037467949'
107 sentiment_extract(str5, userId)
108 #push_data_to_db()
109 print('****')
110 for w in category_list:
111     print(w)
112 print('****')
113 for w in phrases:
114     print(w)
115 print('****')
116 print(doc_ref)

```

./code/machine-learning/simple/start-talking.py

```
1 import re
2
3 def get_responses():
4     #Function start
5     print("--User asks something--", '\nReturn a Response')
6     #Function end
7
8 def set_responses():
9     #Function start
10    print("Thanks for the answer!")
11    #Function end
12
13 #Start Talking to Therabot
14 print('Therabot Here!')
15 print('Say Ask or Answer Questions to Get Started!')
16
17 '''
18 bot for asking questions and learning
19 1. gather questions and ask
20 2. store the response and question as a pair
21
22 bot to answer
23 1. gather question answer pair, based on the similarity(of the category)
   to the question return a response
24 2. check the question, response pair and reply
25 '''
26 a = input().lower()
27 if re.match('ask', a):
28     print("Okay, go on ask me something.")
29     get_responses()
30 elif re.match('answer', a):
31     print("Okay, here are some things I'm preoccupied with.")
32     set_responses()
33 else:
34     print('Invalid Input')
```

References

- [1] A. Argal et al. “Intelligent travel chatbot for predictive recommendation in echo platform”. In: *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. 2018, pp. 176–183. DOI: [10.1109/CCWC.2018.8301732](https://doi.org/10.1109/CCWC.2018.8301732).
- [2] ES Atwell. “Web chatbots: the next generation of speech systems?” In: *European CEO* (2005), pp. 142 –143. URL: <http://eprints.whiterose.ac.uk/81678/>.
- [3] Y. Bengio, P. Simard, and P. Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. ISSN: 1045-9227. DOI: [10.1109/72.279181](https://doi.org/10.1109/72.279181).
- [4] Leon Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of COMPSTAT’2010*. Ed. by Yves Lechevallier and Gilbert Saporta. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186. ISBN: 978-3-7908-2604-3.
- [5] *Designing Conversational Agents for Energy Feedback*. May 2018.
- [6] G. M. D’silva et al. “Real world smart chatbot for customer care using a software as a service (SaaS) architecture”. In: *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2017, pp. 658–664. DOI: [10.1109/I-SMAC.2017.8058261](https://doi.org/10.1109/I-SMAC.2017.8058261).
- [7] F. Ertam and G. Aydin. “Data classification with deep learning using Tensorflow”. In: *2017 International Conference on Computer Science and Engineering (UBMK)*. 2017, pp. 755–758. DOI: [10.1109/UBMK.2017.8093521](https://doi.org/10.1109/UBMK.2017.8093521).
- [8] Ahmed Fadhil. “A Conversational Interface to Improve Medication Adherence: Towards AI Support in Patient’s Treatment”. In: (Mar. 2018).
- [9] Asbjørn Følstad et al. *SIG: Chatbots for Social Good*. Apr. 2018.
- [10] Joan Haliburn et al. “Partnerships in Mental Health: Collaboration in Short-Term Dynamic Interpersonal Psychotherapy with Adolescents”. In: 52 (May 2018).

- [11] Sharea Ijaz et al. "Psychological therapies for treatment-resistant depression in adults". In: 5 (May 2018).
- [12] *India is facing a possible mental health epidemic, says President - The Hindu.* URL: <http://www.thehindu.com/news/national/karnataka/india-is-facing-a-possible-mental-health-epidemic-says-president/article22335971.ece>.
- [13] Alice Kerly, Phil Hall, and Susan Bull. "Bringing chatbots into education: Towards natural language negotiation of open learner models". In: *Knowledge-Based Systems* 20.2 (2007). AI 2006, pp. 177 –185. ISSN: 0950-7051. DOI: <https://doi.org/10.1016/j.knosys.2006.11.014>. URL: <http://www.sciencedirect.com/science/article/pii/S0950705106001912>.
- [14] X. Lu, A. Zhou, and H. Yang. "A novel method design for diagnosis of psychological symptoms of depression using speech analysis". In: *2017 International Conference on Orange Technologies (ICOT)*. 2017, pp. 18–21. DOI: [10.1109/ICOT.2017.8336078](https://doi.org/10.1109/ICOT.2017.8336078).
- [15] D. Madhu et al. "A novel approach for medical assistance using trained chatbot". In: *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*. 2017, pp. 243–246. DOI: [10.1109/ICICCT.2017.7975195](https://doi.org/10.1109/ICICCT.2017.7975195).
- [16] Seema Mehrotra et al. "Development and Pilot Testing of an Internet-Based Self-Help Intervention for Depression for Indian Users". In: 8 (Mar. 2018), p. 36.
- [17] Martino Mensio. *Deep Semantic Learning for Conversational Agents*. Apr. 2018.
- [18] *Mental health awareness: The Indian scenario*. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5479084/>.
- [19] Stuart A. Montgomery and Marie Åsberg. "A New Depression Scale Designed to be Sensitive to Change". In: *British Journal of Psychiatry* 134.4 (1979), pp. 382–389. DOI: [10.1192/bjp.134.4.382](https://doi.org/10.1192/bjp.134.4.382).
- [20] K. J. Oh et al. "A Chatbot for Psychiatric Counseling in Mental Healthcare Service Based on Emotional Dialogue Analysis and Sentence Generation". In: *2017 18th IEEE International Conference on Mobile Data Management (MDM)*. 2017, pp. 371–375. DOI: [10.1109/MDM.2017.64](https://doi.org/10.1109/MDM.2017.64).
- [21] Juanan Pereira and Oscar Díaz. *A quality analysis of Facebook Messenger's most popular chatbots*. Apr. 2018.
- [22] "Prediction of Student Academic Performance using Neural Network, Linear Regression and Support Vector Regression: A Case Study". In: 180 (May 2018), pp. 39–47.

- [23] B Abu Shawar and ES Atwell. *Chatbots: can they serve as natural language interfaces to QA corpus?* 2010. URL: <http://eprints.whiterose.ac.uk/82296/>.
- [24] Bayan Shawar and Eric Atwell. “Chatbots: Are they Really Useful?” In: 22 (Jan. 2007), pp. 29–49.
- [25] *Stress and Anxiety: Causes and Management*. URL: <https://www.healthline.com/health/stress-and-anxiety>.
- [26] *World Health Organization, Mental health in India*. URL: http://www.searo.who.int/india/topics/mental_health/about_mentalhealth/en/.
- [27] X. Zhou et al. “Visually Interpretable Representation Learning for Depression Recognition from Facial Images”. In: *IEEE Transactions on Affective Computing* (2018), pp. 1–1. ISSN: 1949-3045. DOI: 10.1109/TAFFC.2018.2828819.