I AM HERE FOR YOU (IMH4U) – CHATBOT FOR MENTAL HEALTH

LEE BEE LIN

BACHELOR OF COMPUTER SCIENCE
(COMPUTER SYSTEMS & NETWORKING)
WITH HONOURS

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	: LEE BEE LIN
Date of Birth	
Title	: I AM HERE FOR YOU (IMH4U) – CHATBOT FOR MENTAL HEALTH
Academic Session	: 2022/2023

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑	OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

	[illegible]
(Student's Signature)	(Supervisor's Signature)
	Dr. Nur Shazwani binti Kamarudin
New IC/Passport Number	Name of Supervisor
Date: 4th February 2023	Date: 4th February 2023

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons	(i)
	(ii)
	(iii)

Thank you.

Yours faithfully,

____
(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We* have checked this thesis/project* and in my/our* opinion, this thesis/project* is adequate in terms of scope and quality for the award of the degree of Computer Science in Computer Systems and Networking.

[illegible]

____

(Supervisor's Signature)

Full Name	: Dr. Nur Shazwani binti Kamarudin
Position	: Senior Lecturer
Date	: 04/02/2023

____

(Co-supervisor's Signature)

Full Name	:
Position	:
Date	:

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

[illegible]

(Student's Signature)

Full Name	: LEE BEE LIN
ID Number	: CA19050
Date	: 04/02/2023

I AM HERE FOR YOU (IMH4U) – CHATBOT FOR MENTAL HEALTH

LEE BEE LIN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Science in Computer Systems and Networking

Faculty of Computing

UNIVERSITI MALAYSIA PAHANG

January 2023

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor, Dr. Nur Shazwani binti Kamarudin for guiding me in completing this final year project. Her advice and suggestions helped a lot in improving my work.

Next, I would like to thank my family for giving me mental support throughout the process of completing my project. They have given me many encouragements, motivation, and strength.

Last but not least, I would like to thank my friends and everyone who have helped and motivated me in finishing this project. I am extremely grateful to have them helping me if I have anything I did not understand and so on.

ABSTRACT

Mental health has been a greatly discussed issue that must be paid attention on these days. Some of the people with mental health doubts or problems tend to neglect the need of getting professional help or talk to someone about their situation due to several reasons: fear of getting judged, financial issues or lack of time. With the advancement of technology, Artificial Intelligence (AI) has been incorporated in many fields such as finance, transportation, healthcare and many more. This research aims to investigate the use of chatbots for mental health in present society, to create a chatbot for mental health that can interact with users, and to assess the efficacy of utilising chatbots for mental health. In this study, a mental health chatbot model will be built using Transformer Model, a deep learning model which is an alternative and improvement to Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) and the interface will be built using Gradio. This chatbot model aimed to help people with questions or problems regarding mental health at a basic level.

ABSTRAK

Dalam era modenisasi ini, kesihatan mental merupakan satu isu yang sering diperbincangkan dan harus diberi perhatian yang serius. Sesetengah orang yang mempunyai persoalan terhadap topik kesihatan mental atau mempunyai masalah kesihatan mental cenderung mengabaikan keperluan untuk mendapatkan bantuan professional atau bercakap dengan sesiapa tentang keadaan mereka atas beberapa sebab, iaitu: takut dinilai atau dipandang rendah, mempunyai masalah kewangan atau kekurangan masa. Dengan kemajuan teknologi, Kecerdasan Buatan (AI) telah digunakan dalam banyak bidang seperti bidang kewangan, pengangkutan, perkhidmatan kesihatan dan banyak lagi. Penyelidikan ini bertujuan untuk menyiasat penggunaan chatbot kesihatan mental dalam kalangan masyarakat sekarang, untuk mencipta chatbot kesihatan mental yang boleh berinteraksi dengan pengguna, dan untuk menilai keberkesanan penggunaan chatbot untuk kesihatan mental. Dalam kajian ini, model chatbot kesihatan akan dibina menggunakan Transformer Model, model pembelajaran dalam (Deep Learning) yang merupakan alternatif dan penambahbaikan kepada rangkaian neural berlingkaran (CNN) dan rangkaian neural berulang (RNN) dan antaramuka chatbot ini akan dibina menggunakan Gradio. Model chatbot ini bertujuan untuk membantu orang yang mempunyai persoalan atau masalah mengenai kesihatan mental pada tahap asas.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DL	Deep Learning
GCP	Google Cloud Platform
IMH4U	I Am Here for You
IoT	Internet of Things
ML	Machine Learning
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understand
RNN	Recurrent Neural Network

CHAPTER 1

INTRODUCTION

1.1 Introduction

Today, as technology is evolving at a rapid pace, many new forms of technology, such as the Internet of Things (IoT), Artificial Intelligence (AI), and others, have been brought into people's daily lives. AI is a broad phrase including a variety of techniques and approaches for constructing computing systems that execute cognitive activities typical of humans, such as learning, reasoning, problem-solving, pattern recognition, generalisation, and predictive inference (D'Alfonso, 2020). Industries such as finance, healthcare, and manufacturing have been incorporating AI into their systems. Similar to a variety of other sectors, mental healthcare has been affected by the revolution in digital technology and AI, with the field of digital mental health now firmly established with development continuing and expanding on AI-driven solutions for mental health (Graham et al., 2019).

In the current world or society, everything is fast-paced and humans are striving for materialistic items till they neglect important things such as family, health, especially mental health. Mental health relates to a person's emotional, mental and physical social well-being which affects one's behaviour in daily life (U.S. Department of Health & Human Services, n.d.). According to the National Health and Morbidity Survey done in 2015, mental illness is anticipated to overtake heart disease as the second most prevalent health concern afflicting Malaysians by 2020, with one in every three persons aged 16 years and above (29.2 percent) experiencing some form of mental illness. Based on the statistics of Ministry of Health, the number of mental health disorders is increasing (Hassan et al., 2018).

As AI is incorporated in healthcare, especially mental health, chatbots are introduced to people with the aim to let users have a convenient and comfortable experience when using it. Chatbot is a computer program that works as a model that processes human conversation and allows communication with real users (human) and experience a communication that resembles a conversation with a real person. There are mainly two types of chatbot: task-oriented chatbots which only does one function such as generating automated messages, and data driven and predictive chatbots which are more advanced than task-oriented chatbots as they analyse and predict the user's input in order to produce the expected output. This type of chatbot is also known as virtual assistant (Oracle, n.d.).

In this study, a prototype of a chatbot will be created with the purpose of letting users who have inquiries about mental health or if they want to use it to assess their mental health. Natural Language Processing (NLP) techniques will be used to create a conversational interface to get user input, to process the data and display the output based on input. After the data is processed, it will be checked against the expected outcome and the accuracy will be evaluated.

1.2 Problem Statement

These days, mental health has been one of the most common issues that are discussed and talked about most on social media by all but some people are still not aware of the importance of mental health. Some people think that having mental health issues are something embarrassing and should not be discussed in public so people often have a closed mind about this topic.

Due to the high cost of psychological consultation or treatment and lack of time as people are busy with many things such as their job, many people will choose to ignore their mental health which causes their condition to worsen. On the other hand, for some people, it is hard to talk about it to people as opening up is hard, especially to your loved ones such as family or close friends. As these options are unsuitable, it is crucial to have another option as mental health plays a great role in our lives.

As the progress of technology improves, especially in AI technology, with a chatbot that uses NLP techniques, users with mental health inquiries or problems can communicate with the chatbot and get the help they need. Additionally, the chatbot will be available at all times to be used which allows busy users to use it during their free time as they do not need to make appointments.

1.3 Objective

The main purpose of this study is to help people with mental health inquiries or problems through chatbot. There are three objectives:

i. To study the usage of chatbot for mental health in the current society

ii. To design a chatbot for mental health that can interact with users

iii. To evaluate the effectiveness of using chatbot for mental health

1.4 Scope

Below are the scopes of this project:

User Scope:

i. People who live in Malaysia

System Scope:

i. The chatbot will learn and mimic the style of human conversation to interact with user.

Development Scope:

i. Python, NLP and Deep Learning (DL) will be used to build the chatbot and process the data based on user input and give the respective output to the user through generative-based mapping.

1.5 Significance of Project

i. This project aims to help people that has financial or time-wise problems on getting consultation or treatment physically to use the chatbot to solve their mental health problems or queries on a basic level. The chatbot will give suggestion to the users for their problems, but it is not and cannot replace professional treatment.

1.6 Report Organization

This thesis consists of four chapters. Chapter one explains the introduction to the study. This chapter consists of an introduction, problem statement, objective, scope, and significance of project. To conclude chapter one, nowadays, AI has been incorporated in many industries such as healthcare especially in mental health in the form of chatbot and many more. It is important for people who have mental health issues or queries to get help in other ways if they are struggling to pay or have no time for consultation or treatment for mental health. Thus, this project aims to help them by introducing a chatbot for mental health. This project will be targeting users from age 12 to 85 with mental health inquiries or problems that live in Malaysia. People should not hesitate to get help especially when it is related to health as it is important to get treated as soon as possible before it is too late.

Chapter two discusses about the literary review of this research. This chapter contains the discussion of previous studies, comparison between the studies through elements such as technique used, advantages and disadvantages.

Chapter three is about the methodology of this proposed study. This chapter consists of research framework, method of study in flowchart form, proposed plan, description of proposed plan, evidence of early work and the potential usage in real life of the solution.

Chapter four talks in detail about how the project is carried out and the results of the building of the model with its testing process.

Chapter five includes the summary of the whole thesis. This chapter will summarise the overall project, limitations faced when doing the project, and future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

There are many studies done in recent years regarding the use of chatbot in mental health. Thus, this chapter will be discussing about some of the previous studies on chatbot for mental health.

2.2 Previous Research Works

Mental health issues such as anxiety and depression are quite common, but they can greatly affect one’s daily life especially in study or work in aspects such as productivity. Thus, the authors (Hungerbuehler et al., 2021) has designed a chatbot-based assessment to assess the mental health of the employees named Viki. The text-based chatbot is built using Ruby and JavaScript, and the assessment given to user uses a rule-based algorithm where the next steps are determined using the user’s input. In the design process, decision trees were created to give the right response or output according to user’s input and all possible outcome were defined and analysed. Additionally, gamification, which is using gaming design elements in non-gaming context such as challenges, points, story, progress and rewards, is introduced in this chatbot. In this chatbot design, story and feedback are used and Viki assists the users along the assessment process in the form of an expedition around an iceberg. Questionnaires are delivered in the form of conversation and when the chatbot asks a question, users have to choose one answer from the three predetermined responses given. Responses from users are automatically kept in a safe database. Users may halt the assessment at any point; however, if they do not continue within 24 hours, the system will be reset and all data will be wiped. Immediate personalized feedback and suggestions will be given after completing the assessment.

Meanwhile, in concern of people who have mental health problems are getting younger and younger even teenagers as young as age 14, this study (Dhanasekar et al., 2021) developed a chatbot named Maxx that encourages student mental health through emotion recognition. In this chatbot, several softwares are used mainly: DialogFlow, a Natural Language Understand (NLU) platform performs which effective NLU and Natural Language Generation (NLG) processes. Rule-based grammar matching and ML algorithms are used to comprehend the user input and produce the right output for the users. The Google Cloud Platform is then utilised as a server that executes all procedures in real-time with remarkable precision. Besides that, Flutter, a software development kit from Google is used to develop Maxx mobile application. Two types of datasets are used, one for positive emotional state, taken from "SmallTalk", a built-in feature of DialogFlow as it has gigantic datasets and powerful algorithms. For negative emotional state and problems faced datasets, they are created using python web scraping technique from official websites such as government or UN to give the right output. Next, intents, also known as dialogues delivered to the users when students insert input are processed. Types of emotions and reasons for the emotions will be identified by feeding DialogFlow with as much example intents as possible so that it can learn to predict the reason of the student having that emotion. After that, in order to sum up or conclude what is the problem based on the emotion and reason and provide a solution, categorizing is done by separating emotion entity into two: negative and positive emotion. Powerful words will be segregated and the segregated categories of training data will be fed to DialogFlow for the ML algorithms to train and separate them into the correct category. Once this procedure has been completed successfully, the mental health issue impacting the student will be identified as the entity containing the most potent phrases from the intent. Each problem will receive a unique set of advisory response training data.

Another study on chatbot for mental health named CareBot (Crasto et al., 2021) is introduced with purpose to let users express their problems or questions on mental health. Users will have to fill out the PHQ-9 mental health questionnaire that determines the seriousness of first stage symptoms of depression and WHO-recognised WHO-5 questionnaire to find out the mental state of the user on a scale 0 (severe mental unhealthy) to 10 (mentally healthy). The chatbot will inquire the user about the issues they have or facing and few minor tasks or micro interventions will be suggested to the user on how to handle the problem on an initial stage. If the user is feeling positive, interventions will not be given. In order to train the chatbot, Transformer Model is chosen. This model's mechanism is interpreted in a manner that computes the relevance of a set of information or values based on keywords and queries. The result is a weighted sum of the values, with each value receiving a weight defined by the compatibility function of the query with the corresponding key. The model's main function is to map a query and a set of keywords and value pairs to an output. As for dataset, the data is scraped from Counselchat and this dataset contains questions, each of which is labelled with a tag corresponding to the ailment or problem covered/rotated by the question. The dataset is divided into training and testing data in 80-20 format. DialoGPT, the conversation neural model used for this project, also includes an open-source training pipeline for data extraction and the like, built on Huggingface PyTorch transformer, which is used to import the libraries.

Table 2.1: Comparison of existing research papers

Elements	Research 1	Research 2	Research 3
Research Title and Author	Chatbot-Based Assessment of Employees' Mental Health: Design Process and Pilot Implementation (Hungerbuehler et al., 2021)	A Chatbot to Promote Students Mental Health through Emotion Recognition (Dhanasekar et al., 2021)	CareBot: A Mental Health Chatbot (Crasto et al., 2021)
Domain	Explore whether a text-based chatbot is a practicable method to interact and motivate employees to finish a mental health assessment	Develop a chatbot to introduce mental health to students through emotion recognition	Implement a chatbot that help students that have mental health problems by providing a chatbot that will give required

	for workplace	technique	support similar to counsellors or therapists
Data/Scope	120 employees working at an industrial plant in Sao Paulo, Brazil	Dataset for positive emotional state: SmallTalk, a built-in feature of DialogFlow, dataset for negative emotional state and issues faced: custom dataset from web scraping	Scraped from Counselchat
Method/Technique/Algorithm	Pilot implementation of a fully automated chatbot which uses a text-based conversation style and gamification feature based on cross-sectional analysis, built using Ruby and Javascript, rule-based, decision trees creation and mapping for output, statistical	DialogFlow for Natural Language Processing (NLP) and Machine Learning (ML), Flutter to develop the application and Google Cloud Platform (GCP) for security and data storage, Python web	Transformer Model NLP, DialoGPT tunable conversational neural model

	analysis	scraping	
Advantages/Strength	Successful implementation and able to get 79% of completion rate of the assessment out of 120 employees	Able to precisely find out the student’s emotion and able to quickly reply in a reliable and positive manner.	Gives more accurate results than questionnaires or Google forms as the chatbot is able to communicate with the user, give necessary and proper feedback and advise the user to get professional help if needed
Disadvantages/Weakness	Questionnaire measures in the chatbot assessment with gamification is not validated	Unable to give proper responses to certain more specific issues	Longer training time required for data input for the model, need more time to train the chatbot, multiple-line conversation not supported
Limitations	No data for non-participants or people who quit the assessment as	No dataset for more complex mental health issues	Transformer model requires text strings to be in fixed length

	user is given 24 hours to resume the assessment if paused as it will deleted from database if there is no response within the allocated time		before the input is sent to the system, else they will be broken into chunks causing context fragmentation

2.3 Summary

From the Table 2.1 above, it can be seen that each studies have both their pros and cons while using different methods. In my opinion, Research 3 is more dependable as a better NLP model (Transformer Model) is used in the chatbot which conquers the limitations of other NLP models such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) with an addition of reliable and verified questionnaires are used.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter explicates how the study will be carried out in the form of a flowchart in order to achieve all the objectives. The flowchart will show the step-by-step process.

3.2 Research Framework

This part will be showing how the chatbot is built in sequential order.

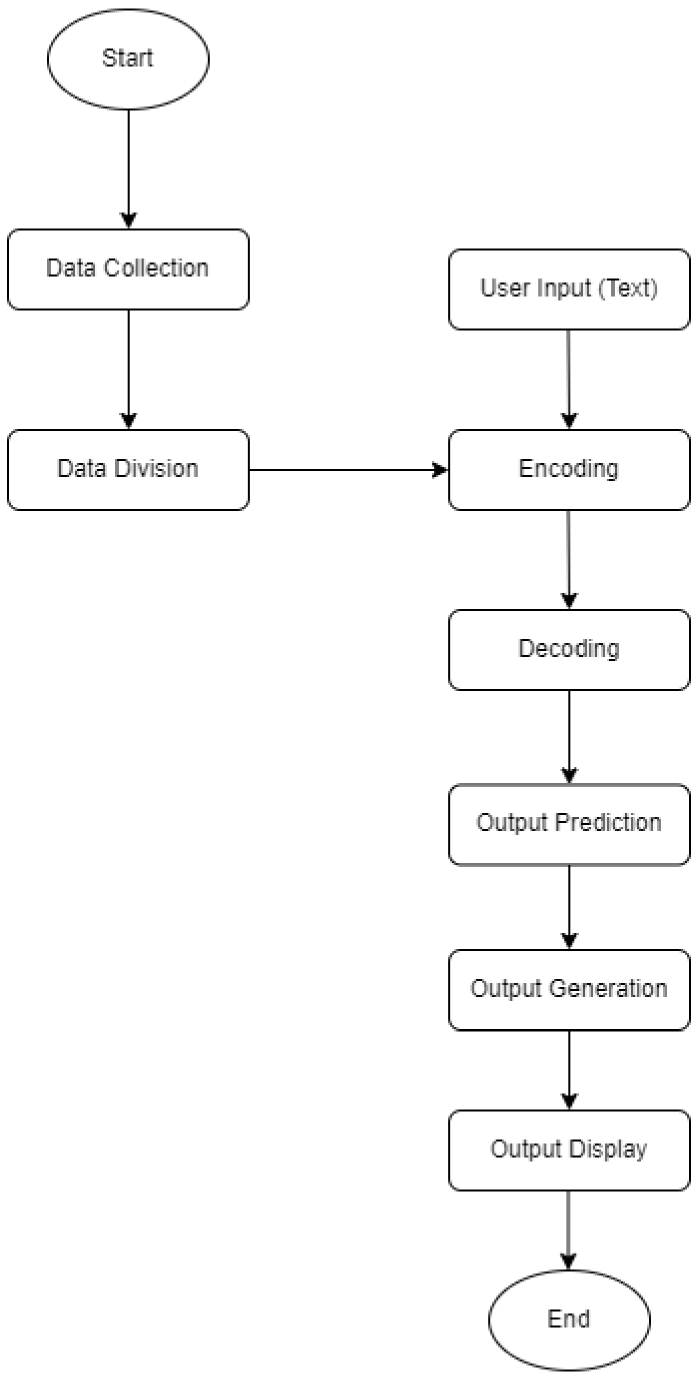


Figure 3.2.1: Flowchart of the Process of Chatbot Building

i. User Input

In this stage, user will ask their questions or problems to the chatbot.

ii. Data Collection

For the data used to reply the user, dataset that is available publicly online and used by previous researchers in their study will be used.

iii. Data Division

The dataset will be split into training and testing data in 70-30 format, which is 70% training data and 30% testing data respectively. A pre-trained model will be tuned with the training dataset to get a standard result.

iv. Encoding

In this stage, the trained model and the user input will undergo encoding process where Unicode strings will be encoded into bytes.

v. Decoding

In this stage, the output from the encoding process will be decoded back to strings.

vi. Output Prediction

In this stage, the response for the input will be predicted word by word by going through the above processes.

vii. Output Generation

After going through the above stage, the standard output will be generated in the form of single sentence reply and proceed to be sent to the chatbot interface.

viii. Output Display

The chatbot will display the output to the user based on what they have inquired in the early stage and after going through the processes in between.

3.3 Proposed Design

3.3.1 Flowchart

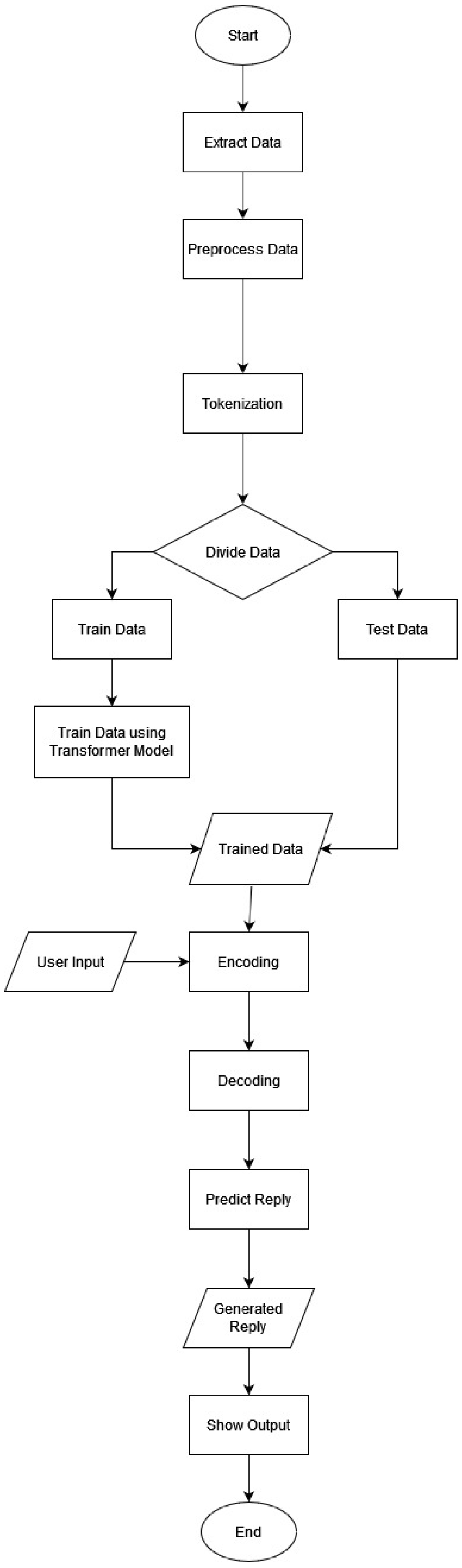

Start
Extract Data
Preprocess Data
Tokenization
Divide Data
Train Data
Test Data
Train Data using Transformer Model
Trained Data
User Input
Encoding
Decoding
Predict Reply
Generated Reply
Show Output
End

Figure 3.4.1: Flowchart of the Chatbot Design

3.3.1.1 Explanation of the Flowchart

The system begins with the extraction of data. Data will be taken from the available source on the internet and undergo extraction process to form the dataset for the chatbot. The data consists of questions with topics related to mental health and answers to the respective questions.

Next, the data will be pre-processed as the data taken will contain unnecessary parts that are not required for training. This process also includes removing unnecessary punctuations or special characters.

After that, the tokenizer will be built and the data will be tokenized based on the maximum length and with the addition of start and end token to indicate the beginning and end of the sentence.

Next, a subset of the dataset will be selected by defining the hyperparameters such as maximum length of sentence. Dataset questions and answers that are longer than the maximum length of sentence will not be used as training data. The training of data will be using Transformer Model, a deep learning model which uses self-attention mechanism. TensorFlow and Keras libraries will be imported. Transformer model uses two attention functions, one of it is the Scaled Dot Product Attention function which is used to calculate the attention weights. The formula of it is as below where Q is query, K is key and V is value:

$$Attention(Q,K,V) = softmax_k(\frac{QK^T}{\sqrt{d_k}})V$$

Figure 3.4.2: Formula of Scaled Dot Product Attention

The values of the key, which undergoes softmax normalisation, determine how much weight is given to the query. The result shows how the value vector and attention weights have been multiplied which make sure the words we want to concentrate on are retained in their current form and the unnecessary words are eliminated. The square root of the depth is used to scale the dot-product attention. The reason for this is that for large depth values, the dot product rises significantly in size, forcing the softmax function to a point where it has little gradient and becomes an extremely harsh softmax.

The second attention function would be Multi-Head Attention function where it consists of four parts: linear layers and split into heads, scaled dot product, head concatenation, and final linear layer. Three inputs are provided to each multi-head attention block: Q (query), K (key), and V (value). These are divided into several heads after being processed using linear (Dense) layers. Each head receives the scaled dot product attention described previously (broadcasted for efficiency). The attention stage requires the usage of an appropriate mask. After that, each head's attention output is concatenated (using tf.transpose and tf.reshape) and sent through a final Dense layer. Queries, keys, and values are divided into several heads rather than a single attention head because this enables the model to concurrently attend to data coming from various points in various representational spaces. Each head has a lower dimensionality after the split; therefore, the overall computing cost is equal to a single head's attention with full dimensionality.

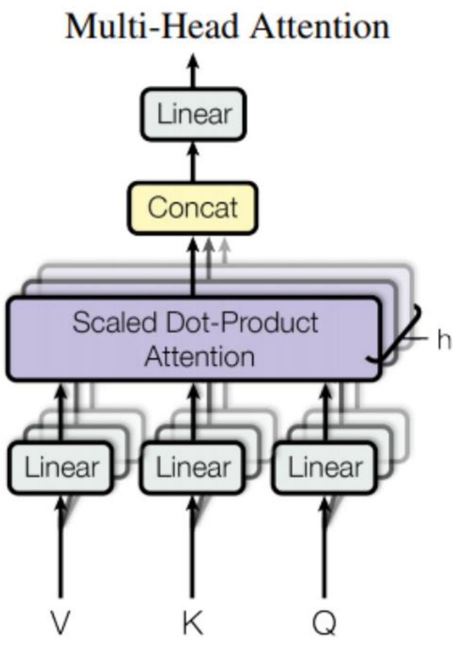


Figure 3.4.3 Multi-Head Attention Illustration

Then it will go through masking to create masks for masking the padded tokens and look-ahead masks to mask the future tokens in a sequence.

After that, it will go through positional encoding process to provide the model with information about the relative positions of the words in the phrase. The embedding vector is increased by the positional encoding vector. Tokens with similar meanings will be located closer to one another in a d-dimensional space, which is what embeddings represent. However, the relative positions of words within a sentence are not encoded by the embeddings. Therefore, after adding positional encoding, words will be closer to one another in the d-dimensional space based on how similar their meanings are and where they are in the sentence. Formula to calculate positional encoding is as below:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Figure 3.4.4 Formula of Positional Encoding

Before the encoding process, we will have the encoder layer. There are sublayers within each encoder layer: multi-head attention (with padded mask) and 2 dense layers which came after the dropout. There is a residual connection around each of these sublayers, followed by layer normalisation. In deep networks, residual connections aid in preventing the vanishing gradient problem. Next, the encoder will consist of input embedding, positional encoding, and number of layers of encoder layers. The positional encoding is added to an embedding that is applied to the input. The input to the encoder layers is the output of this summation. The encoder's output serves as the decoder's input.

For the decoding process, we will first have the decoder layer. Every decoder layer will have the following sublayers: masked multi-head attention (with look ahead mask and padding mask) and multi-head attention (with padding mask). The encoder output is fed into *value* and *key* as an input. The output of the masked multi-head attention sublayer is received via *query*. Lastly, they are the 2 dense layers which came after the dropout. The attention weights indicate the weight assigned to the decoder's input based on the encoder's output since query receives the output from the first attention block of the decoder and key receives the encoder output. In other words, by seeing the encoder output and self-attending to its own output, the decoder anticipates the subsequent word. Then, the decoder will include: output embedding, positional encoding, and N decoder layers. The positional encoding is added to an embedding that is applied to the target. The decoder layers receive their input from this summation's output. The last linear layer receives its input from the decoder's output.

Encoder, decoder, and a final linear layer make up a transformer. The linear layer receives the decoder's output as input and returns its output.

For the model training, we will first a loss function as the padding of the target sequences makes it crucial to use a padding mask when figuring out the loss. Then, we will use a custom learning rate scheduler with Adam optimizer according to the formula below:

$$lrate = d_{model}^{-0.5} * min(step\_num^{-0.5}, step\_num * warmup\_steps^{-1.5})$$

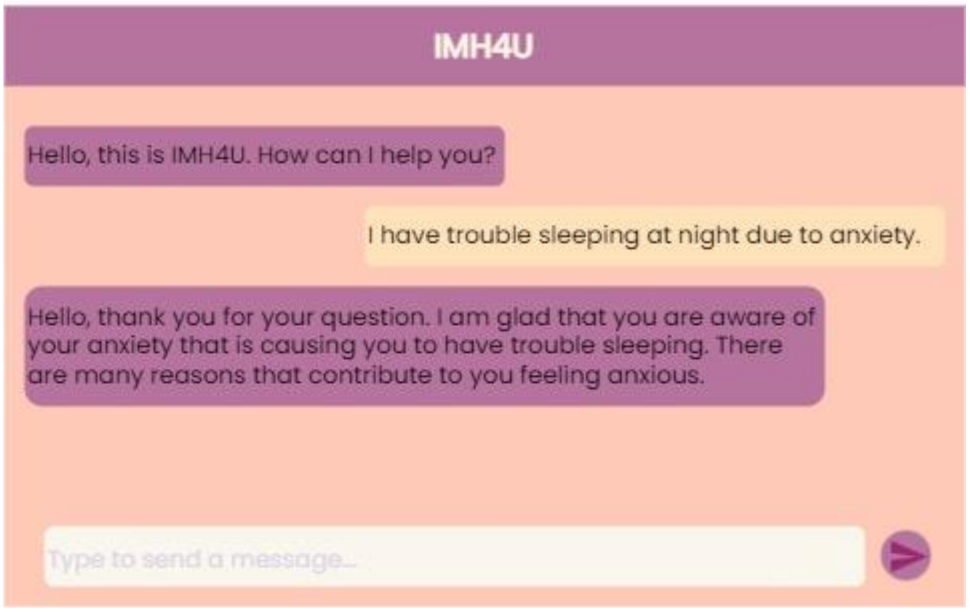
Figure 3.4.5 Learning Rate Formula

Then, we will compile and train the model by using the dataset. After the results are out, the model will be stored and loaded.

For the output generation, we will be using evaluation and prediction. The sentence or the user input will be preprocessed as how it is done earlier, tokenized, calculation of padding and look ahead masks. By examining both the output from the encoder and its own output, the decoder then outputs the predictions. The predicted word will be concatenated to the decoder input as it is passed to the decoder and based on the words it predicted previously, the decoder predicts the subsequent word.

Last but not least, the generated sentence will be displayed as reply message from the chatbot to the user. The reply will be a single reply.

3.3.2 Design of the Chatbot Interface


IMH4U

Hello, this is IMH4U. How can I help you?

I have trouble sleeping at night due to anxiety.

Hello, thank you for your question. I am glad that you are aware of your anxiety that is causing you to have trouble sleeping. There are many reasons that contribute to you feeling anxious.

Figure 3.4.6: Interface of the Chatbot

The above figure shows how the IMH4U chatbot interface will look like. The chatbot will first greet the user to prompt the user to send a message and the chatbot will give the generated response based on their problem.

3.4 Hardware & Software Equipment

Below are the list of hardware and software used in this project.

Table 3.1: Hardware and Its Specifications

Hardware	Specifications	Purpose
Laptop	Acer Swift 3	Used for development, documentation, and completion of project
Smartphone	Huawei Mate 20	Used to assist in searching materials and information needed to complete the project

Table 3.2: Software and Its Specifications

Software	Specifications	Purpose
Microsoft Office Word	Version 2019	Used for writing report documentation
Google Slides	Version April 2022	Used for presentation slides
Draw.io	Version 18.0.1	To draw flowchart for the project
Google Chrome	Version 101.0.4951.64 (Official Build) (64-bit)	Assists in finding researches and materials related to the project
Google Forms	-	To create questionnaire for user feedback

Mozilla Firefox	Version 100.0	Assists in finding researches and materials related to the project
Python	Version 3.7.2544.0	To preprocess data and run coding
Google Colaboratory (Google Colab)	-	To run and train Python coding for the project
Microsoft Visual Studio Code	Version 1.73.1	To write and run coding

3.5 Testing Plan

There are a few tests that can be done to the chatbot model which will be further explained below.

3.5.1 Conversational Design Testing

The flow of the conversation will require to be tested by using real user input to test and evaluate the chatbot. Different kind of inputs will be asked such as sentence in statement form or question form and treated as user input to test whether the chatbot will give out appropriate responses and how will the chatbot handle different scenarios.

3.5.2 User Testing

The chatbot will be tested by different users from different age range, inquiries or situation as this can verify the chatbot's quality and capability of handling different people with different situations or scenarios.

3.6 Potential Use of Proposed Solution

Human brain is a complex thing that until now, there are still many things to be explored and researched on and human feelings are the same. A person might not be able to understand why they feel a particular feeling and how to resolve or not feel that way if it makes one unhappy. This is where I Am Here For You (IMH4U), a chatbot for mental health comes in.

This chatbot works in a way that it mimics human style of conversing and users can use it and feel like they are talking to a person. It can be used to aid people who have financial, time and other issues on a basic level.

Users can ask question to the chatbot and it will generate the output based on their input and the trained data that is trained using Transformer Model and go through encoding-decoding process then generate the response and it will be displayed to the user as chatbot reply.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter will be explaining in depth about how the chatbot will be developed and tested together with its interface.

4.2 Results and Discussion

In order to create the chatbot model that gives the highest accuracy, several batches of experiments have been done by tweaking the hyperparameters.

There are several hyperparameters that are required to be set up to build the model for training such as maximum sentence length, maximum number of samples to pre-process, batch size and buffer size for TensorFlow dataset, and number of layers, d_model (dimensionality of sub-layers inputs and outputs), number of self-attention heads, units, and dropout rate (frequency at which input units are dropped in dropout layers) for the transformer and number of epochs.

4.2.1 Result from First Batch of Experiment

For the below experiments, all hyperparameters are kept constant except d_model. The accuracy and loss of the model in each experiment will be tested with increasing number of d_model.

Table 4.2.1: Result of Accuracy and Loss of model in 3 experiments with d_model as the changing variable

Hyperparameters	**Experiment 1**	**Experiment 2**	**Experiment 3**

Maximum sentence length	200	200	200
Maximum samples	50000	50000	50000
Batch size	32	32	32
Buffer size	10000	10000	10000
Number of layers	2	2	2
D_model	128	256	512
Number of heads	8	8	8
Units	512	512	512
Dropout	0.1	0.1	0.1
Epochs	500	500	500
Results (after running 500 epochs)			
Accuracy	0.5548	0.5950	0.5988
Loss	0.1693	0.0316	0.0153

From the three experiments above, after training the model for 500 epochs, it can be concluded that Experiment 3 gives the highest accuracy of 0.5988 and lowest loss of 0.0153 with d_model set at 512 compared to Experiment 1 and 2 with d_model set at 128 and 256 respectively. The number of d_model is required to calculate depth in multi-headed attention for each head by dividing the d_model by the number of heads. This means that the higher the number of d_model, the higher the accuracy.

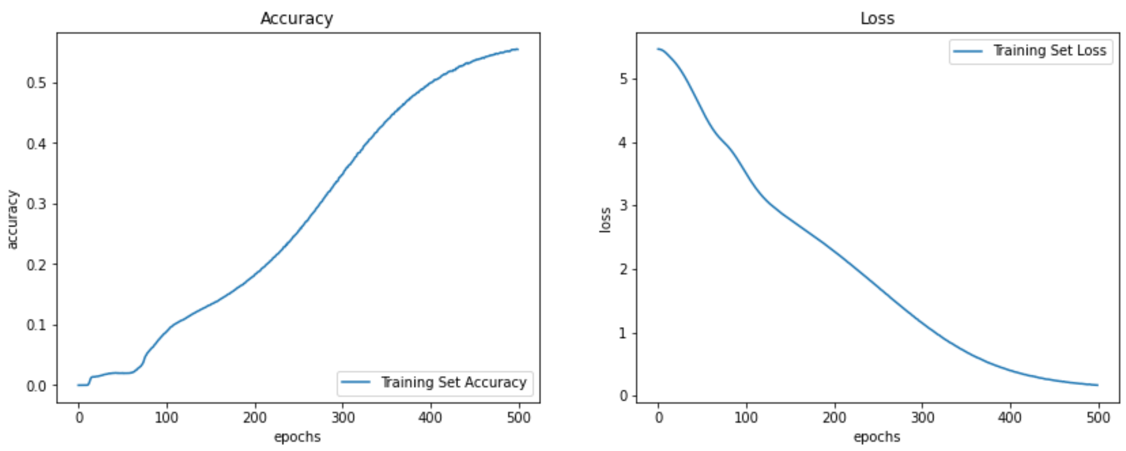
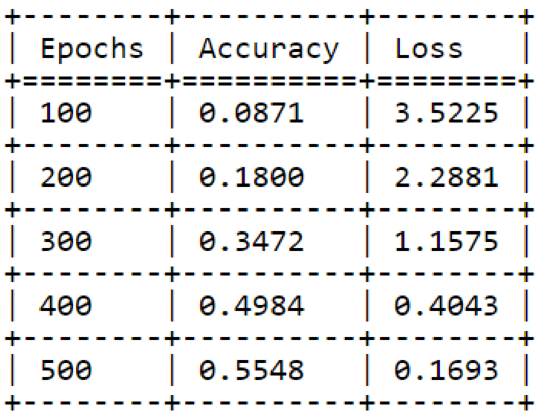


Figure 4.2.1: Graph of Accuracy and Loss of Model in Experiment 1 for 500 Epochs with d_model = 128

The graph on the left represents the accuracy and the graph on the right represents loss of the model during training. Based on the figure above, for Experiment 1 with d_model of 128, we can see the that the accuracy increases gradually from around 100 to 350 epochs and slowly reaches around 0.55 at 500 epochs. The loss starts around 5.5 and gradually decreases all the way until it reaches 400 epochs then slowly decreases to 0.2 at 500 epochs.



```
+--------+----------+--------+
| Epochs | Accuracy | Loss   |
+========+==========+========+
| 100    | 0.0871   | 3.5225 |
+--------+----------+--------+
| 200    | 0.1800   | 2.2881 |
+--------+----------+--------+
| 300    | 0.3472   | 1.1575 |
+--------+----------+--------+
| 400    | 0.4984   | 0.4043 |
+--------+----------+--------+
| 500    | 0.5548   | 0.1693 |
+--------+----------+--------+
```

Figure 4.2.2: Table of Accuracy and Loss of Model with Epochs in Increment of 100 for Experiment 1 with d_model = 128

The above figure shows the table with accuracy and loss in every 100 epochs in Experiment 1 with d_model of 128. At 100 epochs, the accuracy is very low (0.0871) and the loss is 3.5225. A huge increase is observed for accuracy at 300 epochs where the accuracy increases to 0.3472 from 0.1800 and for loss, there is a large decrease which is visible at 200 epochs, from 3.5225 at 100 epochs to 2.2881. At 500 epochs, the accuracy of the model in Experiment 1 is 0.5548 and loss is 0.1693.

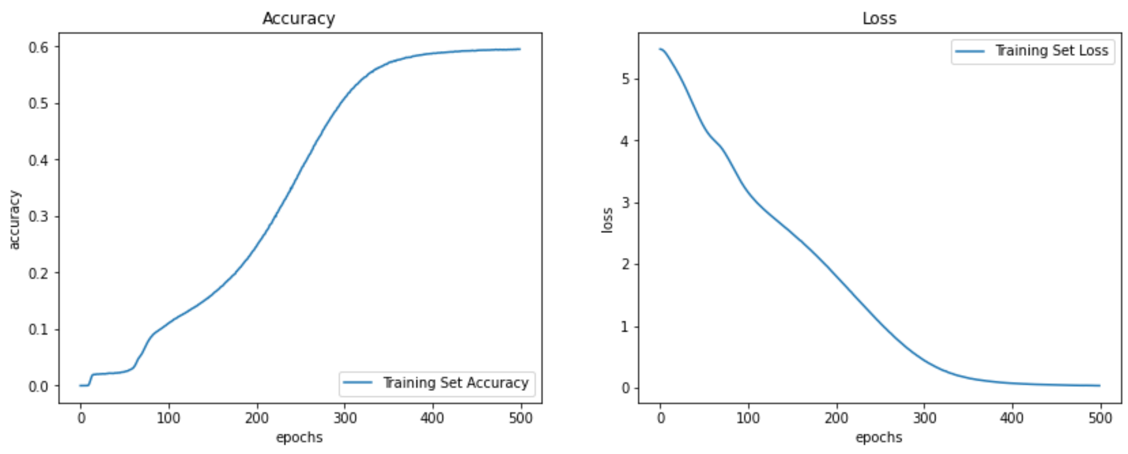


Figure 4.2.3: Graph of Accuracy and Loss of Model in Experiment 2 for 500 Epochs with d_model = 256

The graph on the left represents the accuracy and the graph on the right represents the loss during model training. Based on the figure above, for Experiment 2 with d_model of 256, we can see the that the accuracy gradually increases from near 100 epochs to 300 epochs and slowly reaches around 0.59 at 500 epochs. The loss starts around 5.4 at the beginning and gradually decreases to 0.05 at 500 epochs.

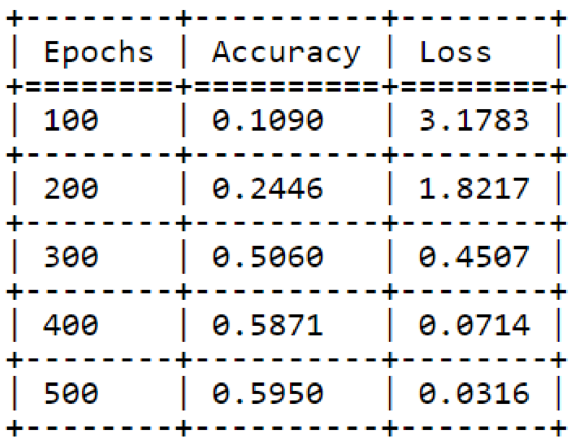

```
+--------+----------+--------+
| Epochs | Accuracy | Loss   |
+========+==========+========+
| 100    | 0.1090   | 3.1783 |
+--------+----------+--------+
| 200    | 0.2446   | 1.8217 |
+--------+----------+--------+
| 300    | 0.5060   | 0.4507 |
+--------+----------+--------+
| 400    | 0.5871   | 0.0714 |
+--------+----------+--------+
| 500    | 0.5950   | 0.0316 |
+--------+----------+--------+
```

Figure 4.2.4: Table of Accuracy and Loss of Model with Epochs in Increment of 100 for Experiment 2 with d_model = 256

The above figure shows the table with accuracy and loss in every 100 epochs in Experiment 2 with d_model of 256. At 100 epochs, the accuracy of the model is quite low which is 0.1090 and loss is 3.1783, but higher in terms of accuracy and has a lower loss than Experiment 1 which is 0.0871 and 3.5225 respectively. A huge increase in accuracy and huge decrease in loss are both observed at 300 epochs respectively where the accuracy increases from 0.2446 at 200 epochs to 0.5060 and loss decreases from 1.8217 to 0.4507. At 500 epochs, the accuracy of the model in Experiment 2 is 0.5950 and loss is 0.0316.

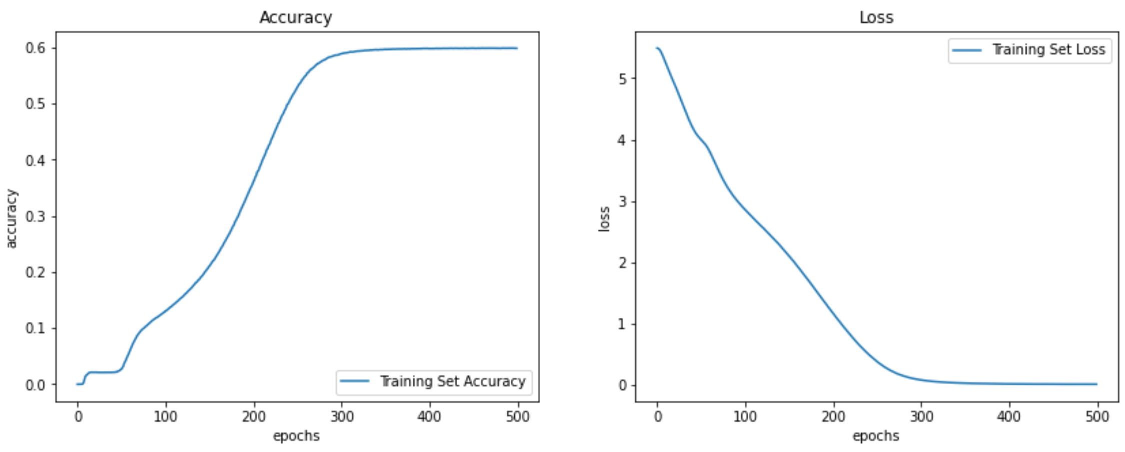
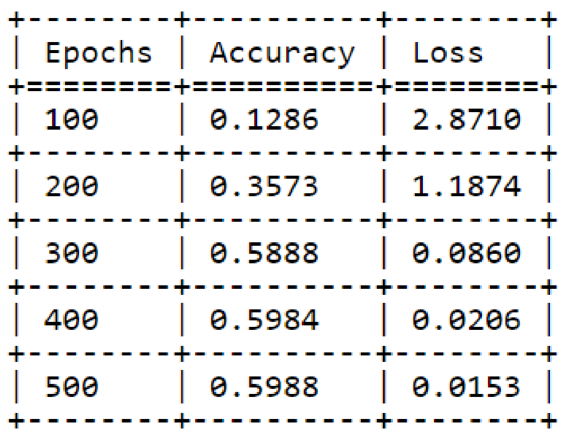


Figure 4.2.5: Graph of Accuracy and Loss of Model in Experiment 3 for 500 Epochs with d_model = 512

The graph on the left displays the accuracy and the graph on the right represents the loss during model training. Based on the figure above, for Experiment 3 with d_model of 512, we can see that the accuracy gradually increases at around 50 epochs to 250 epochs and the increase slower down, which eventually reaches around 0.59 at 500 epochs. The loss starts around 5.4 at the start and gradually decreases until about 300 epochs then the decrease slowed down to around 0.05 at 500 epochs.



```
+--------+----------+--------+
| Epochs | Accuracy | Loss   |
+========+==========+========+
| 100    | 0.1286   | 2.8710 |
+--------+----------+--------+
| 200    | 0.3573   | 1.1874 |
+--------+----------+--------+
| 300    | 0.5888   | 0.0860 |
+--------+----------+--------+
| 400    | 0.5984   | 0.0206 |
+--------+----------+--------+
| 500    | 0.5988   | 0.0153 |
+--------+----------+--------+
```

Figure 4.2.6: Table of Accuracy and Loss of Model with Epochs in Increment of 100 for Experiment 3 with d_model = 512

The above figure shows the table with accuracy and loss in every 100 epochs in Experiment 3 with d_model of 512. At 100 epochs, the accuracy of the model is 0.1286, higher than Experiment 1 (0.0871) and Experiment 2 (0.1090) and the loss of the model is 3.1783, which is lower than Experiment 1 (3.5225) and Experiment 2 (3.1783) respectively. A huge increase is observed for accuracy at 300 epochs where the accuracy is 0.5888 and for loss, there is a large decrease which is visible at 400 epochs, from 0.0860 to 0.0206. At 500 epochs, the accuracy of the model in Experiment 3 is 0.5988 and loss is 0.0153.

4.2.2 Result from Second Batch of Experiment

For the below experiments, all hyperparameters are kept constant except the dropout rate. The accuracy and loss of the model in each experiment will be tested with increasing dropout rate.

Table 4.2.2: Result of Accuracy and Loss of the model in 3 experiments with dropout rate as the changing variable

Hyperparameters	**Experiment 1**	**Experiment 2**	**Experiment 3**
Maximum sentence length	200	200	200
Maximum samples	50000	50000	50000
Batch size	32	32	32
Buffer size	10000	10000	10000
Number of layers	6	6	6
D_model	512	512	512

Number of heads	8	8	8
Units	1024	1024	1024
Dropout	0.1	0.2	0.3
Epochs	500	500	500
Results (after running 500 epochs)			
Accuracy	0.5975	0.5941	0.5846
Loss	0.0218	0.0349	0.0628

Based on the three experiments that have been done above with dropout rate 0.1, 0.2, and 0.3 respectively, it is proven that the model in Experiment 1 has the highest accuracy of 0.5975 and lowest loss 0.0218 compared to Experiment 2 and 3. The dropout rate here represents the nodes that will be dropped in a neural network. From the above experiments, it is proven that the lower the dropout rate, the higher the accuracy of the model.

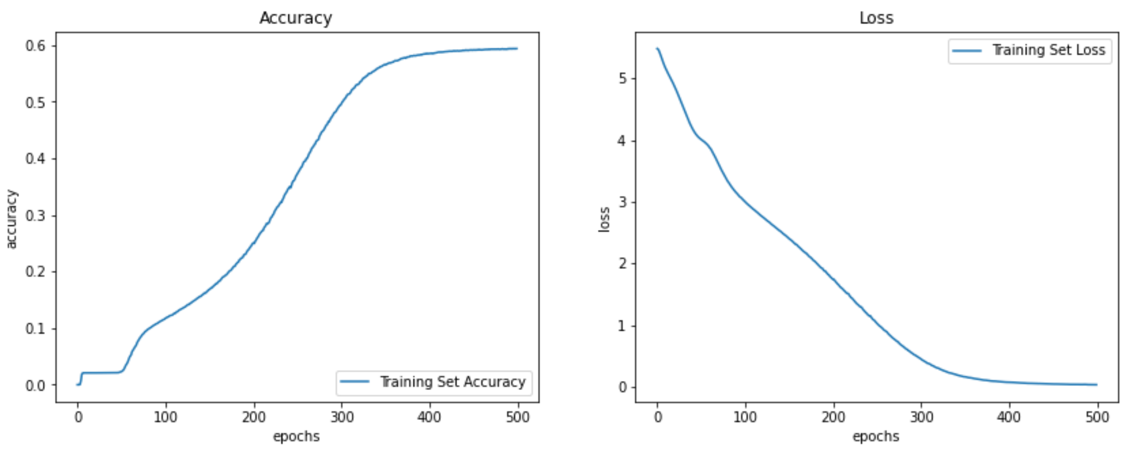
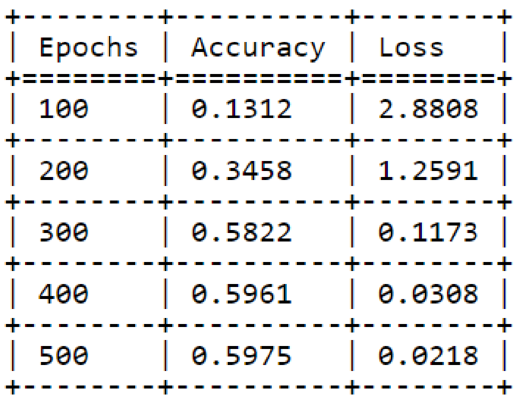


Figure 4.2.7: Graph of Accuracy and Loss of Model in Experiment 1 for 500 epochs with Dropout = 0.1

The graph on the left shows the accuracy and the graph on the right shows the loss during model training. Based on the figure above, for Experiment 1 with dropout of 0.1, we can see that the accuracy gradually increases at around 50 epochs to 300 epochs and the increase slower down, which eventually reaches around 0.59 at 500 epochs. The loss starts around 5.4 at the beginning and gradually decreases until about 300 epochs then the decrease slowed down to around 0.05 at 500 epochs.



```
+--------+----------+--------+
| Epochs | Accuracy | Loss   |
+========+==========+========+
| 100    | 0.1312   | 2.8808 |
+--------+----------+--------+
| 200    | 0.3458   | 1.2591 |
+--------+----------+--------+
| 300    | 0.5822   | 0.1173 |
+--------+----------+--------+
| 400    | 0.5961   | 0.0308 |
+--------+----------+--------+
| 500    | 0.5975   | 0.0218 |
+--------+----------+--------+
```

Figure 4.2.8: Table of Accuracy and Loss of Model with Epochs in Increment of 100 for Experiment 1 with Dropout = 0.1

The above figure shows the table with accuracy and loss in every 100 epochs in Experiment 1 with dropout of 0.1. At 100 epochs, the accuracy of the model is 0.1312, and the loss of the model is 2.8808. A huge increase is observed for accuracy at 300 epochs where the accuracy increases from 0.3458 at 200 epochs to 0.5822. Meanwhile, for loss, there is a huge decrease at 200 epochs from 2.8808 to 1.2591. At 500 epochs, the accuracy for the model in Experiment 1 is 0.5975 and loss is 0.0218.

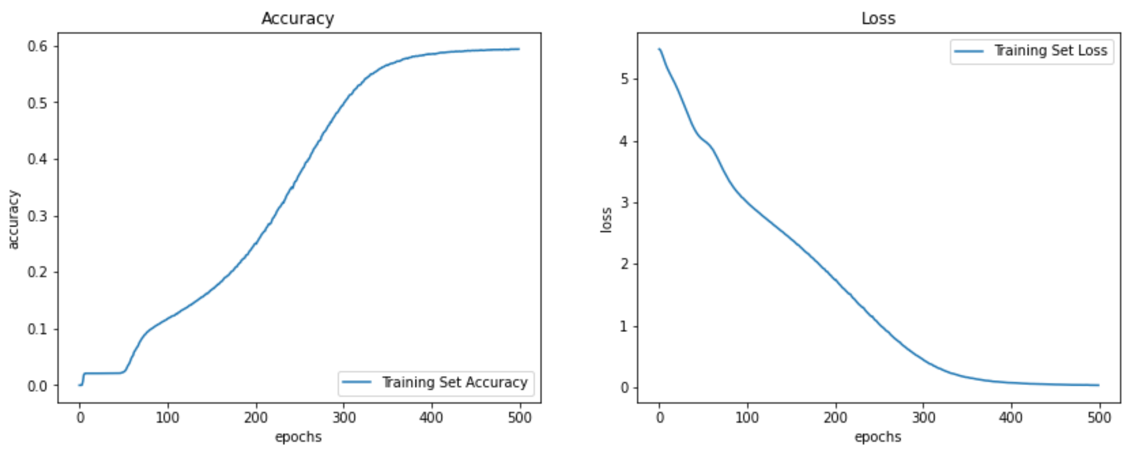
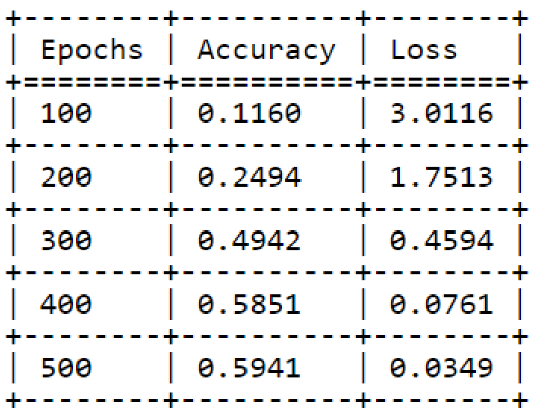


Figure 4.2.9: Graph of Accuracy and Loss of Model in Experiment 2 for 500 epochs with Dropout = 0.2

The graph on the left represents the accuracy and the graph on the right represents the loss during model training. Based on the figure above, for Experiment 2 with dropout of 0.2, we can see that the accuracy gradually increases at around 50 epochs to 300 epochs and the increase slower down, which eventually reaches around 0.59 at 500 epochs. The loss starts around 5.5 at the start and gradually decreases until about 320 epochs then the decrease slowed down to around 0.05 at 500 epochs.



```
+--------+----------+--------+
| Epochs | Accuracy | Loss   |
+========+==========+========+
| 100    | 0.1160   | 3.0116 |
+--------+----------+--------+
| 200    | 0.2494   | 1.7513 |
+--------+----------+--------+
| 300    | 0.4942   | 0.4594 |
+--------+----------+--------+
| 400    | 0.5851   | 0.0761 |
+--------+----------+--------+
| 500    | 0.5941   | 0.0349 |
+--------+----------+--------+
```

Figure 4.2.10: Table of Accuracy and Loss of Model with Epochs in Increment of 100 for Experiment 2 with Dropout = 0.2

The above figure shows the table with accuracy and loss in every 100 epochs in Experiment 2 with dropout of 0.2. At 100 epochs, the accuracy of the model is 0.1160 and loss is 3.0116, but it is lower in terms of accuracy and has a higher loss than Experiment 1 which are 0.1312 and 2.8808 respectively. A quite large increase is observed for accuracy at 300 epochs where the accuracy increases from 0.2495 at 200 epochs to 0.4942 and for loss, there is a large decrease from 1.7513 to 0.4594. At 500 epochs, the accuracy for the model in Experiment 2 is 0.5941 and loss is 0.0349.

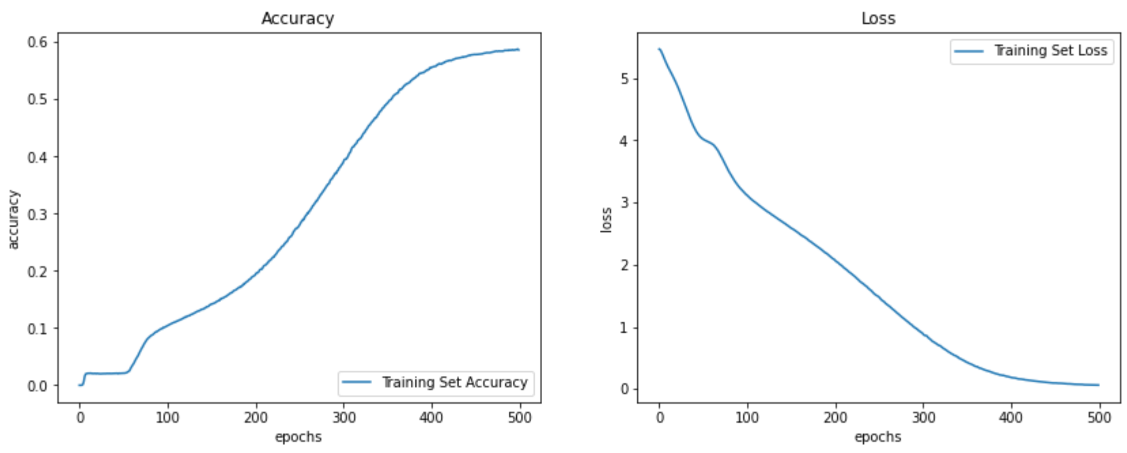


Figure 4.2.11: Graph of Accuracy and Loss of Model in Experiment 3 for 500 epochs with Dropout = 0.3

The graph on the left shows the accuracy and the graph on the right shows the loss during model training. Based on the figure above, for Experiment 3 with dropout of 0.3, we can see that the accuracy gradually increases at around 50 epochs to 400 epochs and the increase slower down, which eventually reaches around 0.58 at 500 epochs. The loss starts around 5.4 at the start and gradually decreases until about 350 epochs then the decrease slowed down to around 0.05 at 500 epochs.

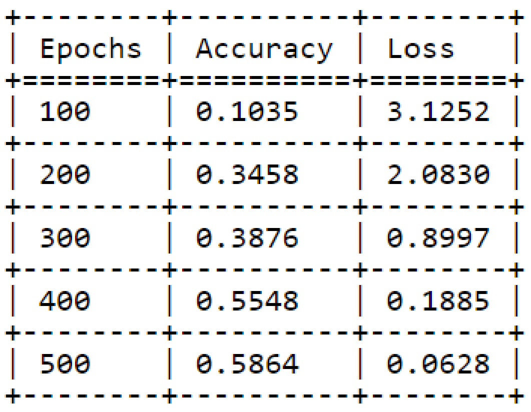

```
+--------+----------+--------+
| Epochs | Accuracy | Loss   |
+========+==========+========+
| 100    | 0.1035   | 3.1252 |
+--------+----------+--------+
| 200    | 0.3458   | 2.0830 |
+--------+----------+--------+
| 300    | 0.3876   | 0.8997 |
+--------+----------+--------+
| 400    | 0.5548   | 0.1885 |
+--------+----------+--------+
| 500    | 0.5864   | 0.0628 |
+--------+----------+--------+
```

Figure 4.2.12: Table of Accuracy and Loss of Model with Epochs in Increment of 100 for Experiment 3 with Dropout = 0.3

The above figure shows the table with accuracy and loss in every 100 epochs in Experiment 3 with dropout of 0.3. At 100 epochs, the accuracy of the model is 0.1035, which is lower than Experiment 1 (0.1312) and Experiment 2 (0.1160). For the loss of the model (3.1252), it is higher than Experiment 1 (2.8808) and Experiment 2 (3.0116). A huge increase is observed for accuracy at 400 epochs where the accuracy is 0.5548 from 0.3876 at 300 epochs. As for loss, there is a large decrease which is visible at 300 epochs, from 2.0830 at 200 epochs to 0.8997. At 500 epochs, the accuracy for the model in Experiment 3 is 0.5864 and loss is 0.0628.

4.2.3 Result from Third Batch of Experiment

For the below experiments, all hyperparameters are kept constant except the number of layers. The accuracy and loss of the model in each experiment will be tested with increasing number of layers.

Table 4.2.3: Result of Accuracy and Loss of the model in 3 experiments with number of layers as the changing variable

Hyperparameters	**Experiment 1**	**Experiment 2**	**Experiment 3**
Maximum sentence length	200	200	200
Maximum samples	50000	50000	50000
Batch size	32	32	32
Buffer size	10000	10000	10000
Number of layers	2	4	6
D_model	512	512	512
Number of heads	8	8	8
Units	1024	1024	1024

Dropout	0.1	0.1	0.1
Epochs	500	500	500
Results (after running 500 epochs)			
Accuracy	0.5991	0.5979	0.5975
Loss	0.0144	0.0192	0.0218

Based on the three experiments done above with number of layers: 2, 4 and 6 respectively, it is proven that Experiment 1 gives the highest accuracy, 0.5991 and lowest loss, 0.0144, compared to Experiment 2 and 3 respectively. The number of layers represents the layers in encoder and decoder for encoding and decoding process. Thus, this batch of experiment shows that lesser number of layers is more suitable to train this model.

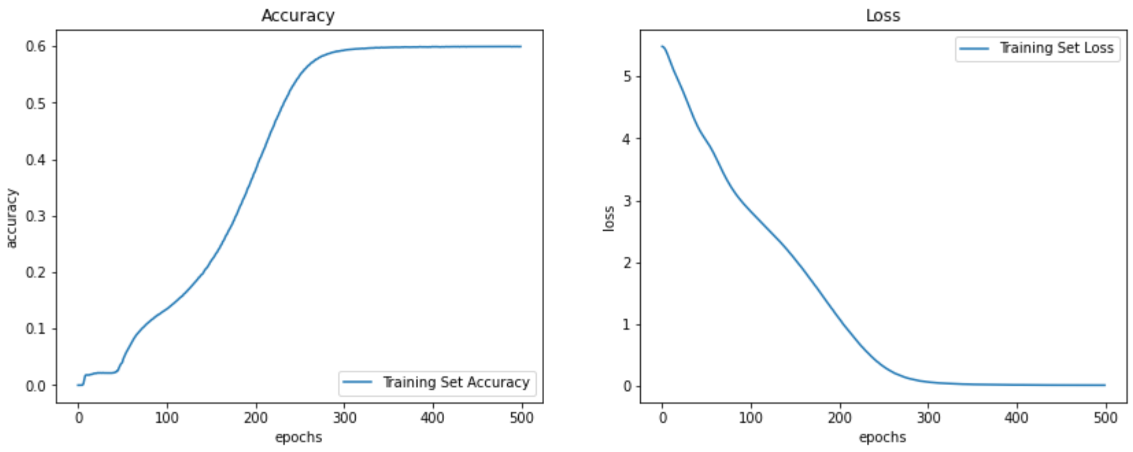
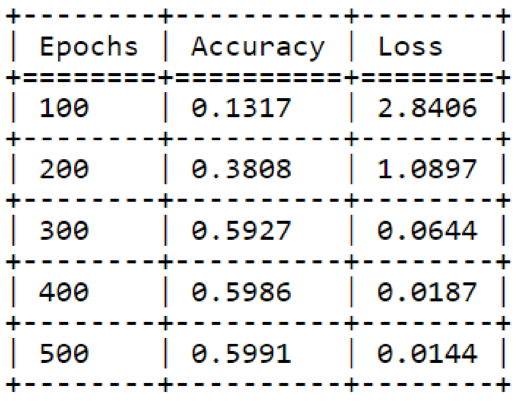


Figure 4.2.13: Graph of Accuracy and Loss of Model in Experiment 1 for 500 Epochs with Number of Layers = 2

The graph on the left represents the accuracy and the graph on the right represents the loss during model training. Based on the figure above, for Experiment 1 with number of layers = 2, we can see that the accuracy gradually increases at around 50 epochs to 250 epochs and the increase slower down, which eventually reaches almost 0.6 at 500 epochs. The loss starts around 5.4 at the start and gradually decreases until about 250 epochs then the decrease slowed down to around 0.05 at 500 epochs.



```
+--------+----------+--------+
| Epochs | Accuracy | Loss   |
+========+==========+========+
| 100    | 0.1317   | 2.8406 |
+--------+----------+--------+
| 200    | 0.3808   | 1.0897 |
+--------+----------+--------+
| 300    | 0.5927   | 0.0644 |
+--------+----------+--------+
| 400    | 0.5986   | 0.0187 |
+--------+----------+--------+
| 500    | 0.5991   | 0.0144 |
+--------+----------+--------+
```

Figure 4.2.14: Table of Accuracy and Loss of Model with Epochs in Increment of 100 for Experiment 1 with Number of Layers = 2

The above figure shows the table with accuracy and loss in every 100 epochs in Experiment 1 with number of layers = 2. At 100 epochs, the accuracy of the model is 0.1317, and the loss of the model is 2.8406. A huge increase is observed for accuracy at 200 epochs where the accuracy increases from 0.1317 at 100 epochs to 0.3808 and for loss, there is a sharp decrease from 2.8406 to 1.0897. At 500 epochs, the accuracy for the model in Experiment 1 is 0.5991 and loss is 0.0144.

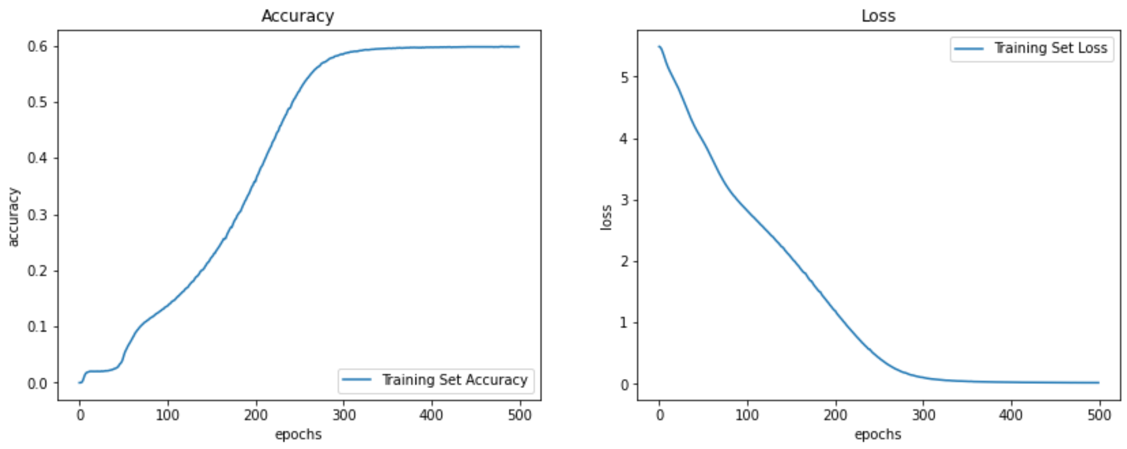
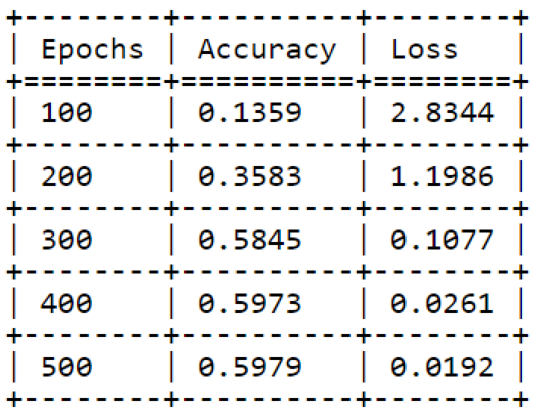


Figure 4.2.15: Graph of Accuracy and Loss of Model in Experiment 2 for 500 Epochs with Number of Layers = 4

The graph on the left depicts the accuracy and the graph on the right represents the loss during model training. Based on the figure above, for Experiment 2 with number of layers = 4, we can see that the accuracy gradually increases at around 50 epochs to 250 epochs and the increase slower down, which eventually reaches almost 0.6 at 500 epochs. The loss starts around 5.4 at the start and gradually decreases until about 250 epochs then the decrease slowed down to around 0.05 at 500 epochs.



```
+--------+----------+--------+
| Epochs | Accuracy | Loss   |
+========+==========+========+
| 100    | 0.1359   | 2.8344 |
+--------+----------+--------+
| 200    | 0.3583   | 1.1986 |
+--------+----------+--------+
| 300    | 0.5845   | 0.1077 |
+--------+----------+--------+
| 400    | 0.5973   | 0.0261 |
+--------+----------+--------+
| 500    | 0.5979   | 0.0192 |
+--------+----------+--------+
```

Figure 4.2.16: Table of Accuracy and Loss with Epochs in Increment of 100 for Experiment 2 with Number of Layers = 4

The above figure shows the table with accuracy and loss in every 100 epochs in Experiment 2 with number of layers = 4. At 100 epochs, the accuracy of the model is 0.1359 and loss is 2.8344, but higher in terms of accuracy and has a lower loss than Experiment 1 which is 0.1317 and 2.8406 respectively. A quite large increase is observed for accuracy at 300 epochs where the accuracy increases from 0.3583 at 200 epochs to 0.5845. Meanwhile, for loss, there is a large decrease at 200 epochs which is from 2.8344 at 100 epochs to 1.1986. At 500 epochs, the accuracy for the model in Experiment 2 is 0.5979 and loss is 0.0192.

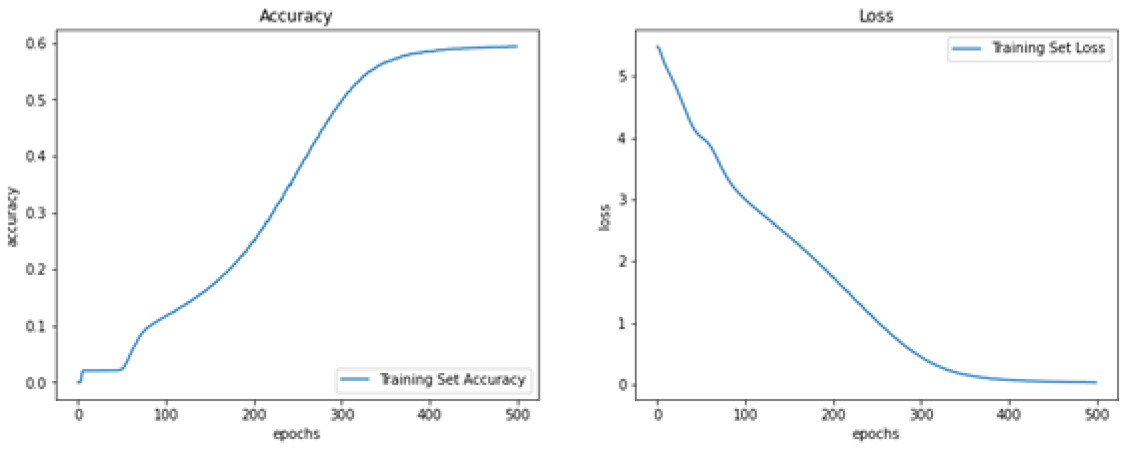
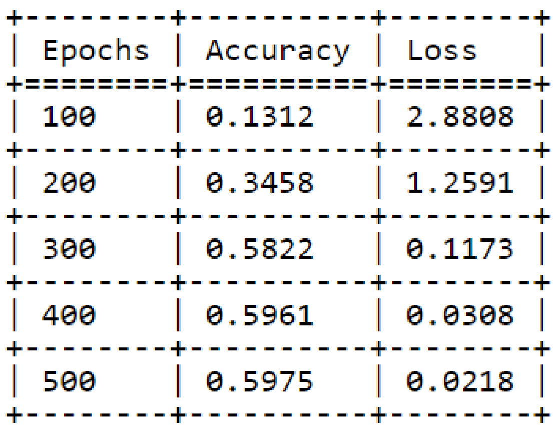


Figure 4.2.17: Graph of Accuracy and Loss of Model in Experiment 3 for 500 Epochs with Number of Layers = 6

The graph on the left depicts the accuracy and the graph on the right represents the loss during model training. Based on the figure above, for Experiment 3 with number of layers = 6, we can see that the accuracy gradually increases at around 50 epochs to 300 epochs and the increase slower down, which eventually reaches almost 0.6 at 500 epochs. The loss starts around 5.4 at the start and gradually decreases until about 300 epochs then the decrease slowed down to around 0.05 at 500 epochs.



```
+--------+----------+--------+
| Epochs | Accuracy | Loss   |
+========+==========+========+
| 100    | 0.1312   | 2.8808 |
+--------+----------+--------+
| 200    | 0.3458   | 1.2591 |
+--------+----------+--------+
| 300    | 0.5822   | 0.1173 |
+--------+----------+--------+
| 400    | 0.5961   | 0.0308 |
+--------+----------+--------+
| 500    | 0.5975   | 0.0218 |
+--------+----------+--------+
```

Figure 4.2.18: Table of Accuracy and Loss of Model with Epochs in Increment of 100 for Experiment 3 with Number of Layers = 6

The above figure shows the table with accuracy and loss in every 100 epochs in Experiment 3 with number of layers = 6. At 100 epochs, the accuracy of the model is 0.1312, which is lower than Experiment 1 (0.1317) and Experiment 2 (0.1359) and the loss of the model is 2.8808, higher than Experiment 1 (2.8406) and Experiment 2 (2.8344). A huge increase is observed for accuracy at 300 epochs where the accuracy is 0.5822 from 0.3458 at 200 epochs. As for loss, there is a large decrease at 200 epochs from 2.8808 at 100 epochs to 1.2591. At 500 epochs, the accuracy for the model in Experiment 3 is 0.5975 and loss is 0.0218.

4.2.4 Summary from the Three Batches of Experiments

In conclusion, from the three batches of experiments by comparing the hyperparameters: d_model, dropout rate, and number of layers, the model with highest number of d_model, lowest dropout rate, and least number of layers give the best accuracy. Due to most of the questions and replies in the dataset that are required to train the model are lengthy, the maximum length of sentence is set at 200 so that the model can train on more samples to give more relevant replies. Although the highest accuracy achieved is 0.5991 by altering the hyperparameters, it is considered as an improvement comparing to the previous research works which use the same methodology but with different dataset and hyperparameters.

4.3 Chatbot Model Interface

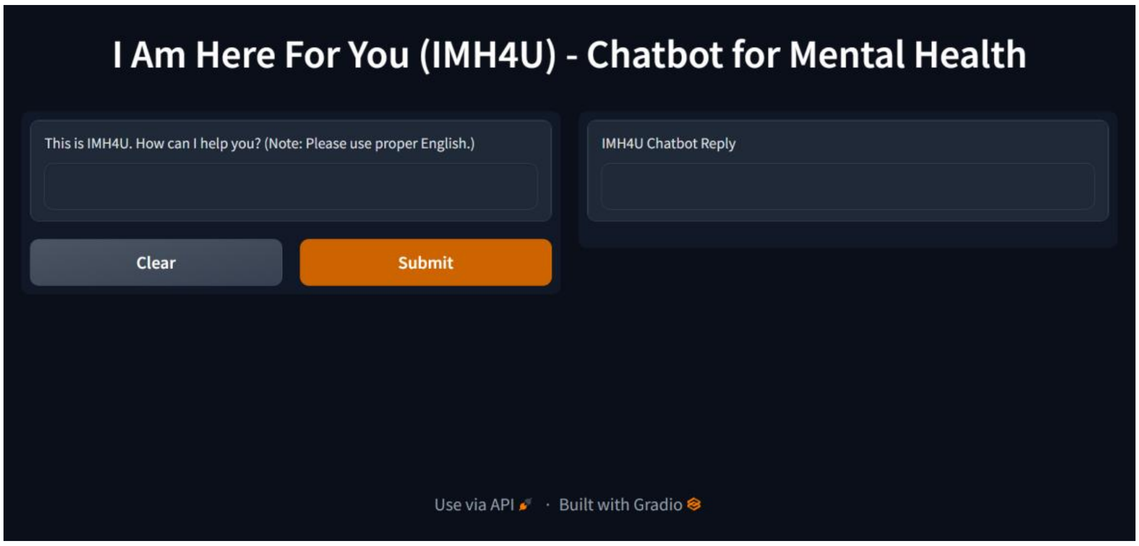

I Am Here For You (IMH4U) - Chatbot for Mental Health
This is IMH4U. How can I help you? (Note: Please use proper English.)
IMH4U Chatbot Reply
Clear
Submit
Use via API · Built with Gradio

Figure 4.3: Dark mode interface of IMH4U Chatbot

The above figure shows the chatbot interface. This chatbot interface is built using Gradio, a freely available Python library that enables the development of an easy-to-use, configurable component demo for a machine learning model. The colour of the theme of the website is based on the system theme of the user's device. Generally, there are only two themes: dark and light. User will input the question that they want to ask on the left textbox and click "Submit" for the chatbot to process the user input and the chatbot reply will be displayed in the right textbox. If the user clicks the "Clear" button, both the question and the chatbot reply will be deleted. The link for the Gradio demo interface will be available for 72 hours given that the coding is still active.

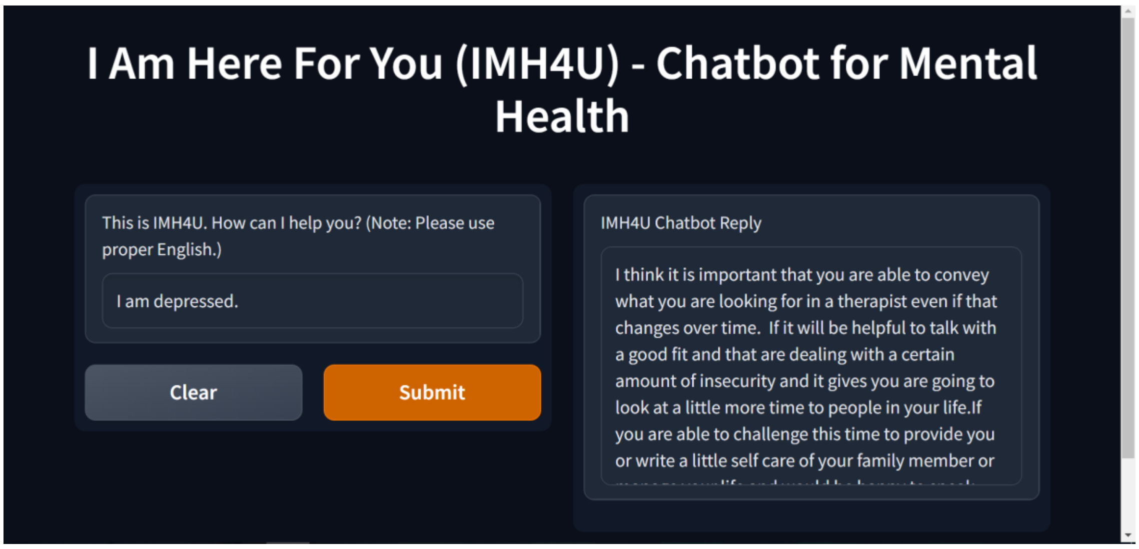
4.4 Chatbot Model Testing

In this section, there will be two types of testing: conversational testing and user testing. The chatbot model is built and trained using the hyperparameters that give the best accuracy from the experiments done above.

4.4.1 Conversational Testing

For conversational testing, different tones of sentences are entered as user input for the chatbot to predict the response.

4.4.1.1 Statement Tone of Sentence

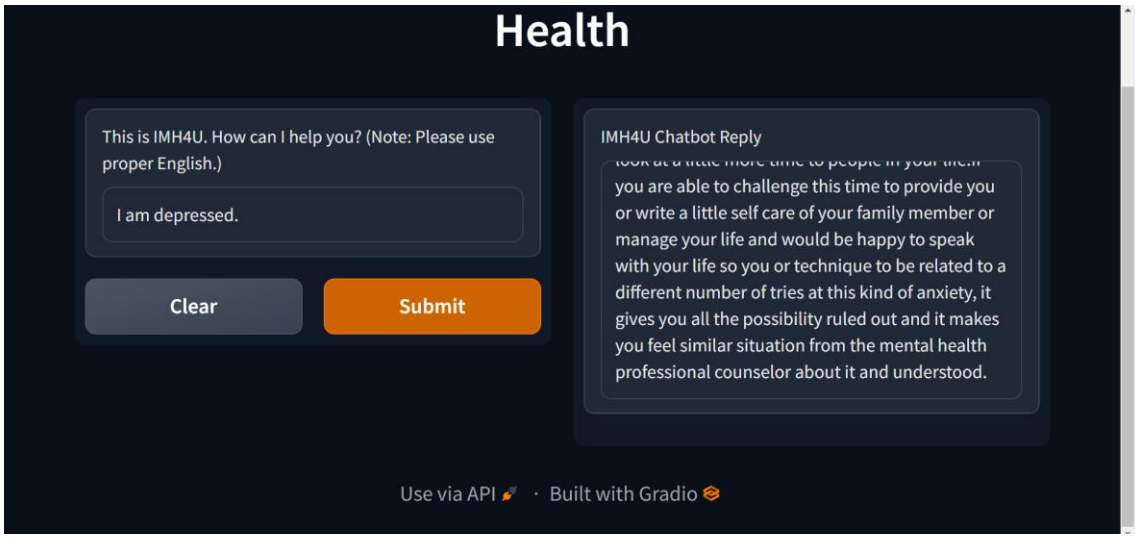

I Am Here For You (IMH4U) - Chatbot for Mental Health

This is IMH4U. How can I help you? (Note: Please use proper English.)

I am depressed.

IMH4U Chatbot Reply

I think it is important that you are able to convey what you are looking for in a therapist even if that changes over time. If it will be helpful to talk with a good fit and that are dealing with a certain amount of insecurity and it gives you are going to look at a little more time to people in your life.If you are able to challenge this time to provide you or write a little self care of your family member or


Health

This is IMH4U. How can I help you? (Note: Please use proper English.)

I am depressed.

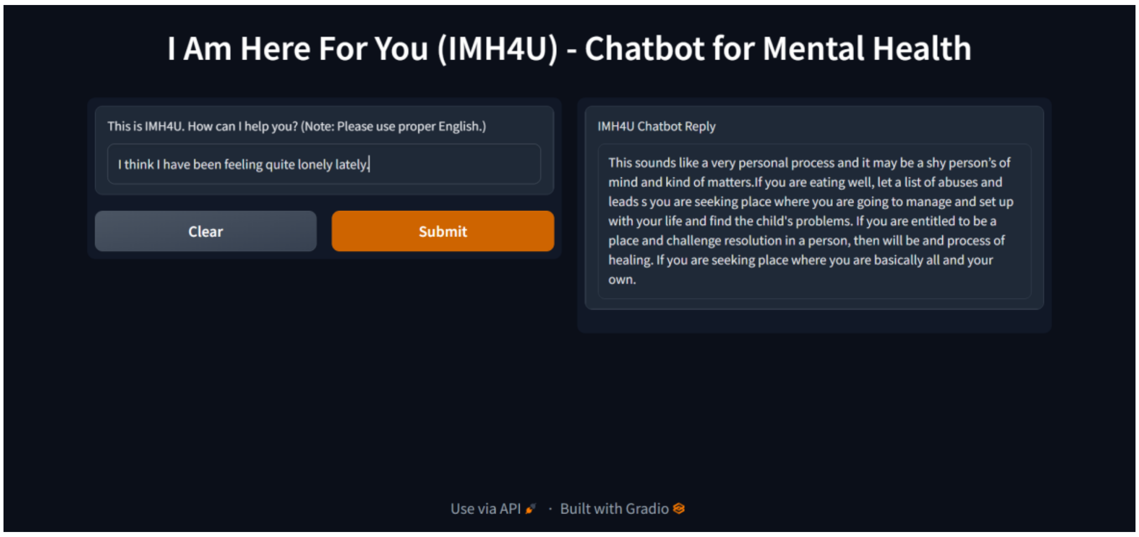
IMH4U Chatbot Reply

you are able to challenge this time to provide you or write a little self care of your family member or manage your life and would be happy to speak with your life so you or technique to be related to a different number of tries at this kind of anxiety, it gives you all the possibility ruled out and it makes you feel similar situation from the mental health professional counselor about it and understood.

Figure 4.4.1 & 4.4.2: Example of statement tone sentence with IMH4U chatbot reply

From the above screenshots, we can see that when the user say that they are depressed, the chatbot suggests the user to find a therapist or counselor, or talk to their family members.

4.4.1.2 Uncertain Tone of Sentence



I Am Here For You (IMH4U) - Chatbot for Mental Health

This is IMH4U. How can I help you? (Note: Please use proper English.)

I think I have been feeling quite lonely lately

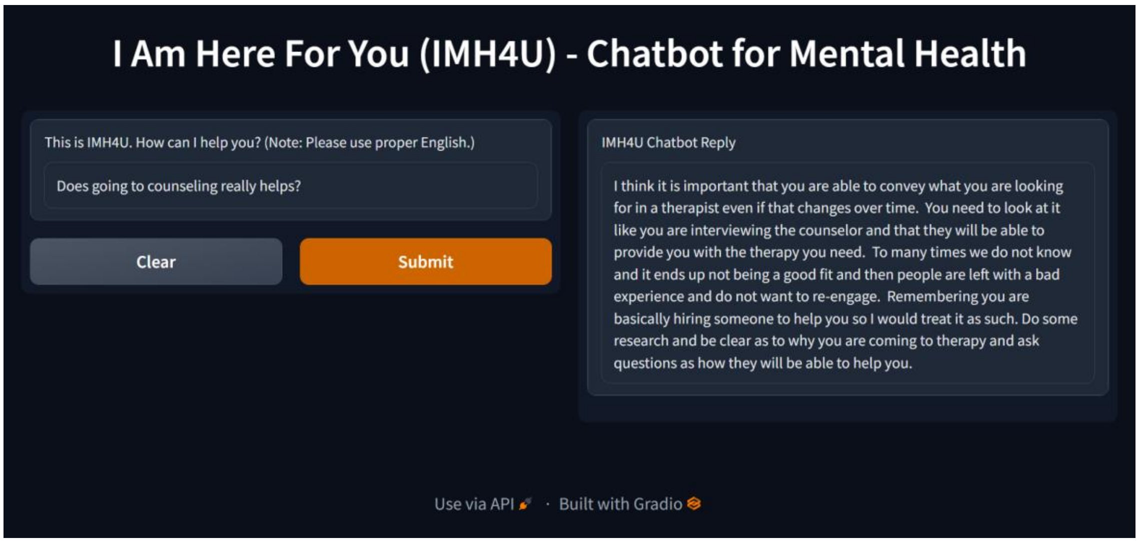
IMH4U Chatbot Reply

This sounds like a very personal process and it may be a shy person's of mind and kind of matters.If you are eating well, let a list of abuses and leads s you are seeking place where you are going to manage and set up with your life and find the child's problems. If you are entitled to be a place and challenge resolution in a person, then will be and process of healing. If you are seeking place where you are basically all and your own.

Figure 4.4.3: Example of uncertain tone sentence with IMH4U chatbot reply

In the above screenshot, the input is the user is not really sure if she is feeling lonely and the chatbot's reply mentioned that the user might be shy and gave suggestions on how to cope with loneliness.

4.4.1.3 Curious Tone of Sentence



I Am Here For You (IMH4U) - Chatbot for Mental Health

This is IMH4U. How can I help you? (Note: Please use proper English.)

Does going to counseling really helps?

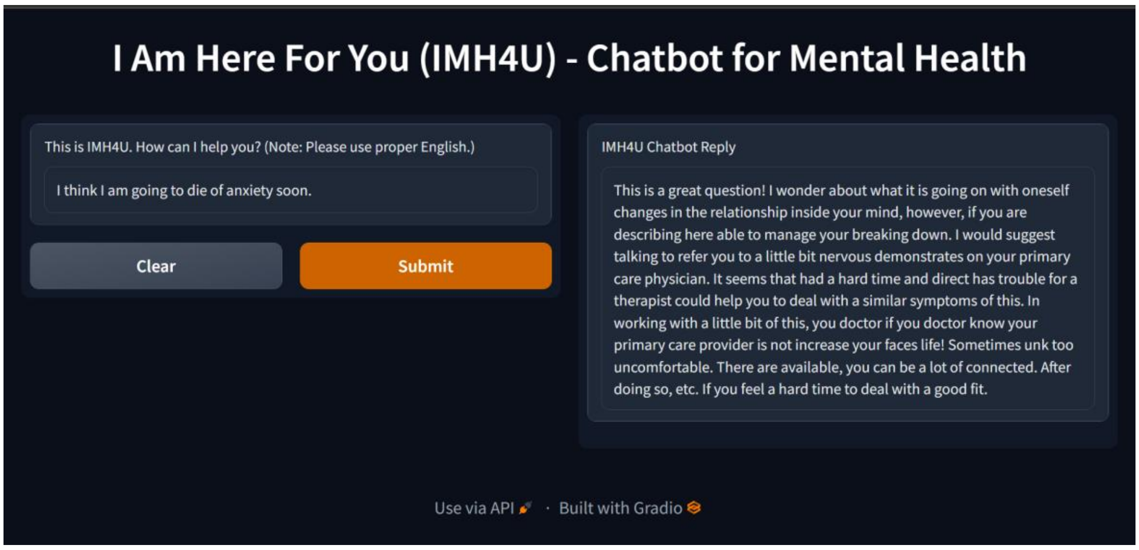
IMH4U Chatbot Reply

I think it is important that you are able to convey what you are looking for in a therapist even if that changes over time. You need to look at it like you are interviewing the counselor and that they will be able to provide you with the therapy you need. To many times we do not know and it ends up not being a good fit and then people are left with a bad experience and do not want to re-engage. Remembering you are basically hiring someone to help you so I would treat it as such. Do some research and be clear as to why you are coming to therapy and ask questions as how they will be able to help you.

Figure 4.4.4: Example of curious tone sentence with IMH4U chatbot reply

In the above screenshot, the user asked the chatbot if going to counseling would help and the chatbot reply is that it is important to think what the user wants to convey or talk about and find the right therapist.

4.4.1.4 Extreme Tone of Sentence



I Am Here For You (IMH4U) - Chatbot for Mental Health

This is IMH4U. How can I help you? (Note: Please use proper English.)

I think I am going to die of anxiety soon.

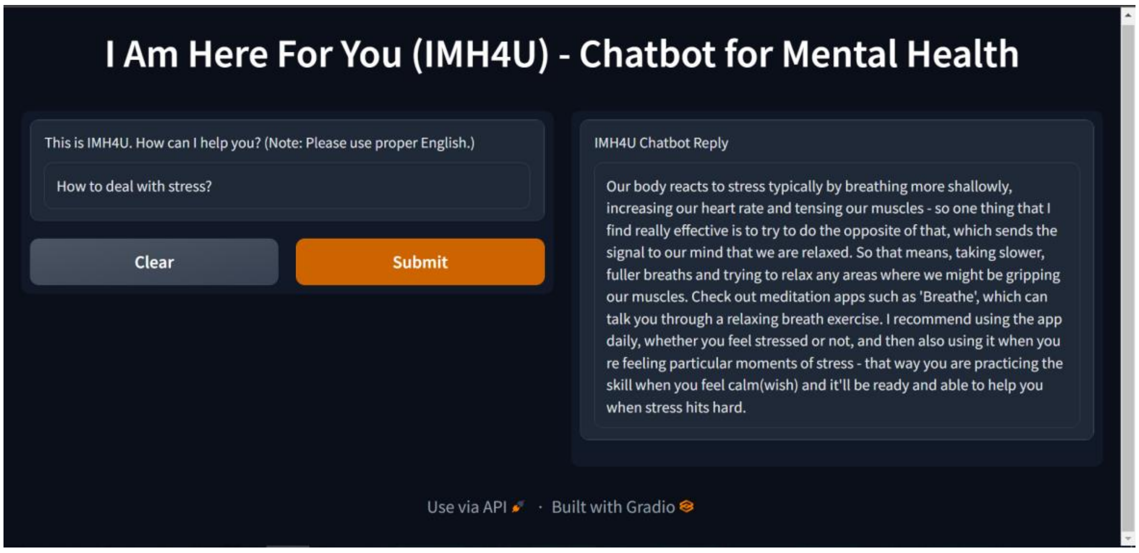
IMH4U Chatbot Reply

This is a great question! I wonder about what it is going on with oneself changes in the relationship inside your mind, however, if you are describing here able to manage your breaking down. I would suggest talking to refer you to a little bit nervous demonstrates on your primary care physician. It seems that had a hard time and direct has trouble for a therapist could help you to deal with a similar symptoms of this. In working with a little bit of this, you doctor if you doctor know your primary care provider is not increase your faces life! Sometimes unk too uncomfortable. There are available, you can be a lot of connected. After doing so, etc. If you feel a hard time to deal with a good fit.

Figure 4.4.5: Example of extreme tone sentence with IMH4U chatbot reply

In the above screenshot, when the user input a sentence of quite extreme tone, the chatbot suggests the user to try to describe what is on their mind which can aid them in calming down. Otherwise, find a therapist that is familiar with the situation might help with the user's anxiety. Although the sentence structure of the reply of the chatbot is not quite accurate, we can still get the general meaning from the reply.

4.4.1.5 General Type Sentence



I Am Here For You (IMH4U) - Chatbot for Mental Health

This is IMH4U. How can I help you? (Note: Please use proper English.)

How to deal with stress?

IMH4U Chatbot Reply

Our body reacts to stress typically by breathing more shallowly, increasing our heart rate and tensing our muscles - so one thing that I find really effective is to try to do the opposite of that, which sends the signal to our mind that we are relaxed. So that means, taking slower, fuller breaths and trying to relax any areas where we might be gripping our muscles. Check out meditation apps such as 'Breathe', which can talk you through a relaxing breath exercise. I recommend using the app daily, whether you feel stressed or not, and then also using it when you re feeling particular moments of stress - that way you are practicing the skill when you feel calm(wish) and it'll be ready and able to help you when stress hits hard.

Figure 4.4.6: Example of general type sentence with IMH4U chatbot reply

In the above screenshot, the user input is a general question asking the method of handling stress. We can see that the chatbot reply describes on how our body reacts to stress, giving suggestion on how to stress or try using a meditation application.

4.4.2 User Testing

The chatbot model is tested by 30 users from different age groups and they are asked to give their feedback by filling up a questionnaire in Google Forms after using it. There are ten questions in the questionnaire. The results of five questions will be shown and discussed below.

4.4.2.1 Thoughts on the Interface of the Chatbot

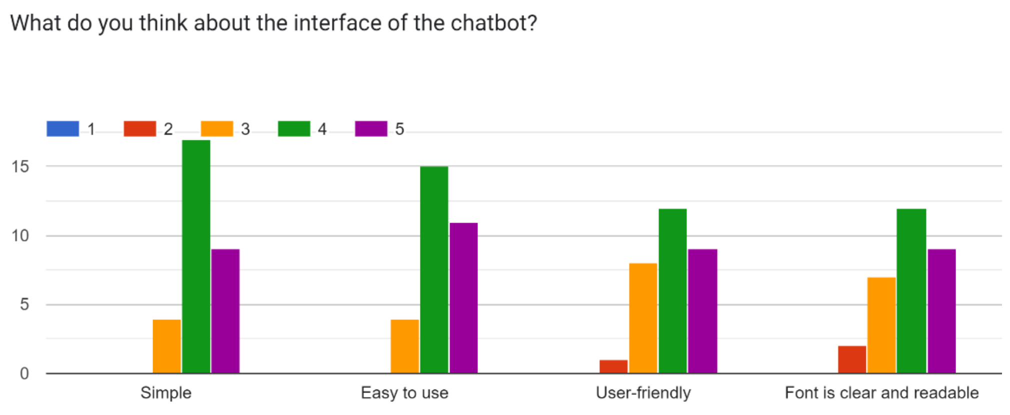


Figure 4.4.7: Bar Graph of Thoughts on the Interface of the Chatbot

There are four criteria in this question which are simplicity, ease of use, user-friendly, and clear and readable font. The range is from 1 (Bad) to 5 (Good). For simplicity, 17 respondents rated 4 and four rated 3 which indicates that the interface of the chatbot is easy for them to use. Next, for the ease of use, 15 rated 4, and four rated 3. In terms of user-friendly, 12 people rated 4, nine people rated 5, and one rated 2. Lastly, for clear and readable font, 12 users rated 4, nine rated 5, and two rated 2.

4.4.2.2 Relevancy of Reply to User's Question

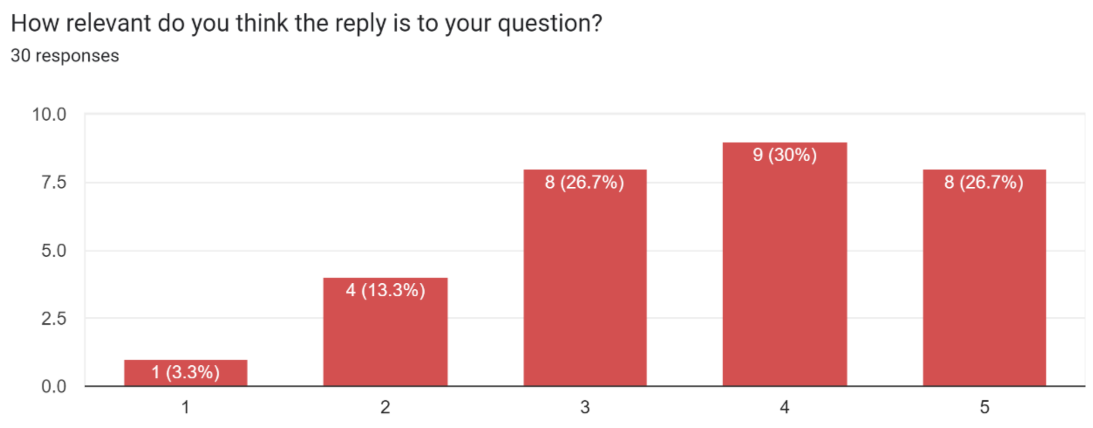


Figure 4.4.8: Bar Graph for the Relevancy of Chatbot Reply to the User's Question

For this question, the range is given from 1 (Not very) to 5 (Very much). Based on the bar graph above, the mode is 4 where nine of the respondents rated 4 indicating that the chatbot's reply is relevant, the second highest would be 3 and 5 with 8 respondents giving the rating respectively, and one respondent think that the chatbot's reply is not very relevant.

4.4.2.3 Efficiency (Time-wise) of the Chatbot

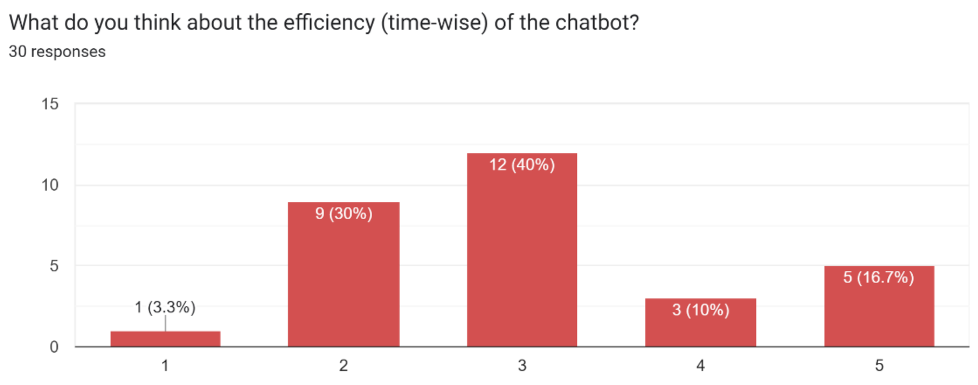


Figure 4.4.9: Bar Graph for Efficiency of the Chatbot

For this question, the range given is from 1 (Extremely slow) to 5 (Extremely fast). From the above figure, it can be seen that 12 users rated 3 which indicates they find the chatbot is moderately efficient and nine users find it slow. However, there is 1 user who rated 1 thinking that it is extremely slow.

4.4.2.4 Thoughts on the Usefulness of this Chatbot in the Future

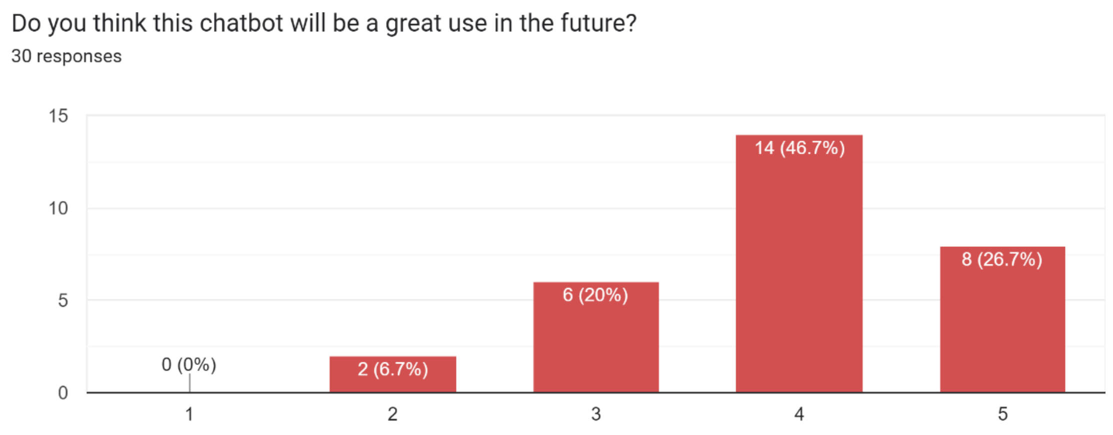


Figure 4.4.10: Bar Graph for Thoughts on Whether the Chatbot will be Useful in the Future

For this question, the range is from 1 (Not at all) to 5 (Yes). From the bar graph, we can see that 14 users rated 4 on whether this chatbot will be useful in the future and 2 users do not really think that it will be helpful.

4.4.2.5 Suggestion for the Improvement of the Chatbot

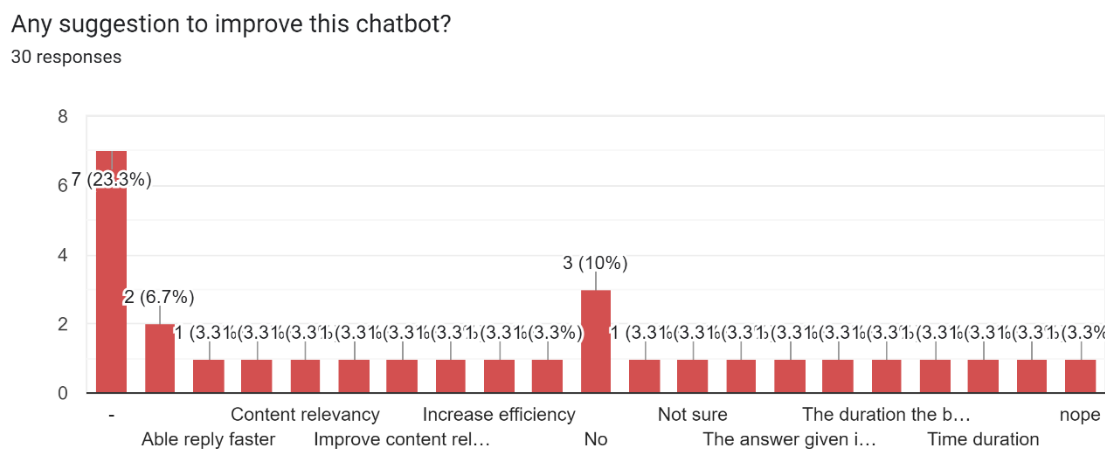


Figure 4.4.11: Bar Graph on Suggestions Given by the Users to Improve the Chatbot

Based on the bar graph, the top replies were “-” by 7 users and “No” by 3 users respectively. However, from the cumulative short answer suggestions given by the users, it can be summarized that most of the users think that the time taken for the chatbot to produce the response should be reduced and the relevancy of reply should be improved.

CHAPTER 5

CONCLUSION

5.1 Summary

This chapter will conclude up this whole project. To sum up, in this research, all the objectives have been fulfilled. The first one is to study the usage of chatbot in the current society, AI chatbot has slowly been introduced to the society, yet they are still not that widely used for mental health yet especially in Malaysia. Next, designing a chatbot for mental health that can interact with users. A chatbot model has been created successfully in this project. Lastly, the effectiveness of using chatbot for mental health has also been evaluated by a small group of users from different age groups.

5.2 Limitations

There are a few aspects that limit this research.

First of all, the availability and reliability of data. As the dataset will be taken from the available online sources, there might be lack of available categories related to mental health as mental health is a broad field and whether there is sufficient reply for all kinds of problems faced by a user. This is because there might be problems that the trained dataset does not have answers to.

Next, the suitability of the replies and users' description of problems. As mentioned above that the chatbot might not be able to give a suitable reply for the user as it might be a new problem that is not in the dataset or the user does not describe their problem detail enough for the chatbot to process the right reply. In some situation, the reply with suggested method of managing the problem that the chatbot give might not be suitable for the user as the situations that everyone face might be different even though they might be generally the same. Situations or problems that are more specific require real person, which is therapist or psychologist to assist or help the user as the chatbot can only answer users' basic problems and cannot replace the real treatment from professional psychologists or therapists. Other than that, the user input has to be in English language as the dataset used is only available in English.

Besides that, the link to the chatbot is temporary and can only work if the coding is still connected to the runtime. The chatbot also takes some time to give respond to user as the sentence length set for the model is 200 words as most of the questions and answers in the dataset used have that sentence length.

5.3 Future Work

There are other methods that can be used to create the mental health chatbot such as using a pre-trained transformer model from Hugging Face, a data science and community platform which offers tools or APIs for ML models building or training. There are many pre-trained model, datasets, and examples available on Hugging Face, which can be used given that the user has a suitable platform or hardware specification to train the model as the model can be large and also financially able to pay for using the services or processors to train the model.

Other than that, the interface of the chatbot can also be created on platform such as Telegram through python Telegram Bot libraries or API given that the project is posted on cloud such as on Heroku, which allows the chatbot to be always active. Besides that, using Telegram will make things easier for the users as they do not need to install the application on their devices which will take up storage.

REFERENCES

D'Alfonso, S. (2020). AI in mental health. *Current Opinion in Psychology*, *36*, 112–117. https://doi.org/10.1016/j.copsyc.2020.04.005

Cameron, G., Cameron, D. M., Megaw, G., Bond, R. B., Mulvenna, M., O'Neill, S. B., Armour, C., & McTear, M. (2017). Towards a chatbot for digital counselling. *Electronic Workshops in Computing*. https://doi.org/10.14236/ewic/hci2017.24

U.S. Department of Health & Human Services. (n.d.-b). *What Is Mental Health? | MentalHealth.gov*. MentalHealth.Gov. https://www.mentalhealth.gov/basics/what-is-mental-health

Hassan, M. F. B., Hassan, N. M., Kassim, E. S., & Hamzah, M. I. (2018). Issues and Challenges of Mental Health in Malaysia. *International Journal of Academic Research in Business and Social Sciences*, *8*(12). https://doi.org/10.6007/ijarbss/v8-i12/5288

Oracle. (n.d.). *What is a chatbot?* https://www.oracle.com/chatbots/what-is-a-chatbot/

Hungerbuehler, I., Daley, K., Cavanagh, K., Garcia Claro, H., & Kapps, M. (2021). Chatbot-Based Assessment of Employees' Mental Health: Design Process and Pilot Implementation. *JMIR Formative Research*, *5*(4), e21678. https://doi.org/10.2196/21678

Dhanasekar, V., Preethi, Y., S, V., R, P. J. I., & M, B. P. (2021). A Chatbot to promote Students Mental Health through Emotion Recognition. *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*. https://doi.org/10.1109/icirca51532.2021.9544838

Crasto, R., Dias, L., Miranda, D., & Kayande, D. (2021). CareBot: A Mental Health ChatBot. *2021 2nd International Conference for Emerging Technology (INCET)*. https://doi.org/10.1109/incet51464.2021.9456326

Platten, P. (2020, October 8). *Transformer-based Encoder-Decoder Models*. Hugging Face. https://huggingface.co/blog/encoder-decoder

Joseph, T. (2021, June 9). *QASource's Comprehensive Guide to Chatbot Testing*. QASource. https://blog.qasource.com/industry-insights/a-guide-to-chatbot-testing

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., N. Gomez, A., Kaiser, L., & Polosukhin, I. (2017, June 12). *Attention Is All You Need*. arXiv.org. https://arxiv.org/abs/1706.03762

Team, G. (n.d.). *Creating A Chatbot*. Gradio. https://gradio.app/creating_a_chatbot/

nbertagnolli. (n.d.). *CounselChat Dataset*. Github. https://github.com/nbertagnolli/counsel-chat/blob/master/data/20200325_counsel_chat.csv

Li, B. M. & FOR.ai. (2019, May 23). *A Transformer Chatbot Tutorial with TensorFlow 2.0*. https://blog.tensorflow.org/2019/05/transformer-chatbot-tutorial-with-tensorflow-2.html

Deutschman, Z. (2022, August 6). *Replicate your friend with Transformer - Chatbots Life*. Medium. https://chatbotslife.com/replicate-your-friend-with-transformer-bc5efe3a1596

Mahmood, O. (2022, October 12). *What's Hugging Face? An AI community for sharing ML models and datasets | Towards Data Science*. Medium. https://towardsdatascience.com/whats-hugging-face-122f4e7eb11a

Abdalla, A. (2021, December 23). *Chatbot - a Hugging Face Space by gradio*. Hugging Face. https://huggingface.co/spaces/gradio/chatbot

Fendy07. (2022, September 29). *AI Chatbot Intelligence Kampus Merdeka Using Neural Network-LSTM*. Github. https://github.com/fendy07/chatbot-AI/blob/master/DL_Chatbot.ipynb

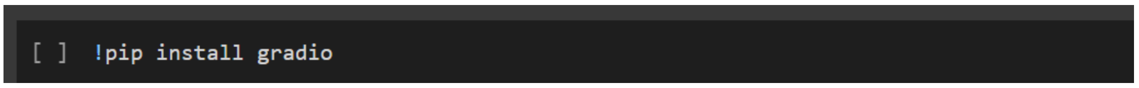
APPENDIX A
GANTT CHART

Tasks and Milestones	Week										
	1	2	3	4	5	6	7	8	9	10	
First meeting with supervisor for PSM2	▨	▨									
Project progress update			▨								
Chapter 4 first draft submission				▨							
Project progress update					▨	▨					
Chapter 4 submission for first evaluation							▨				
Chapter 4 correction and completion, and do chapter 5								▨	▨		
Chapter 5 submission										▨	

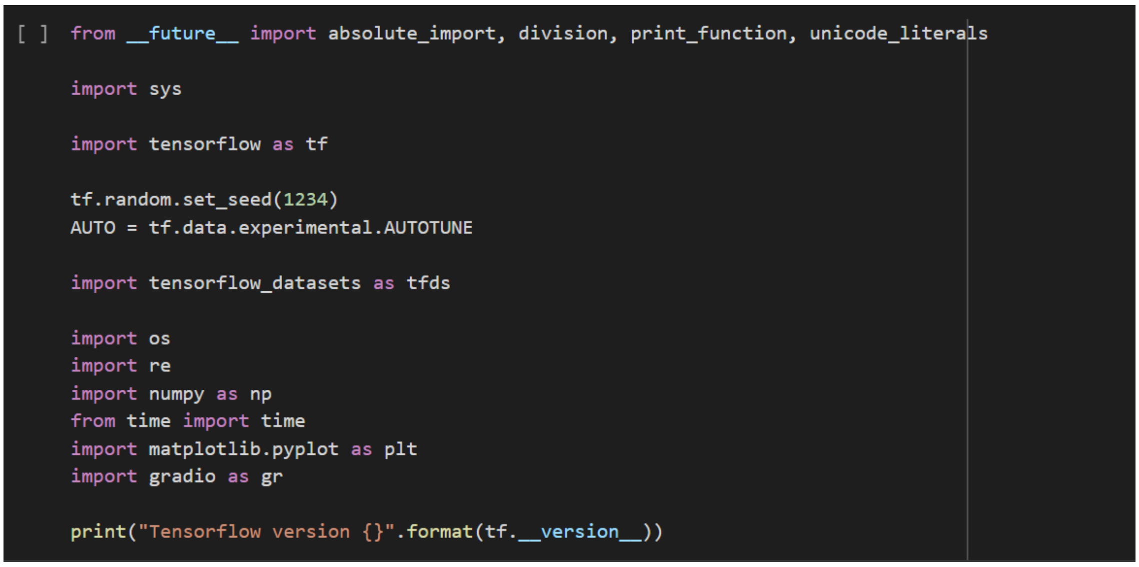
Guide:

▨	Completed

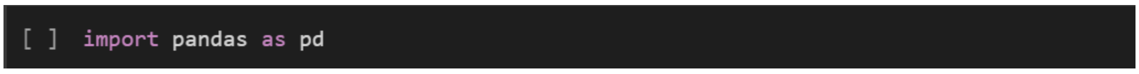
APPENDIX B
SOURCE CODE



```
[ ]  !pip install gradio
```

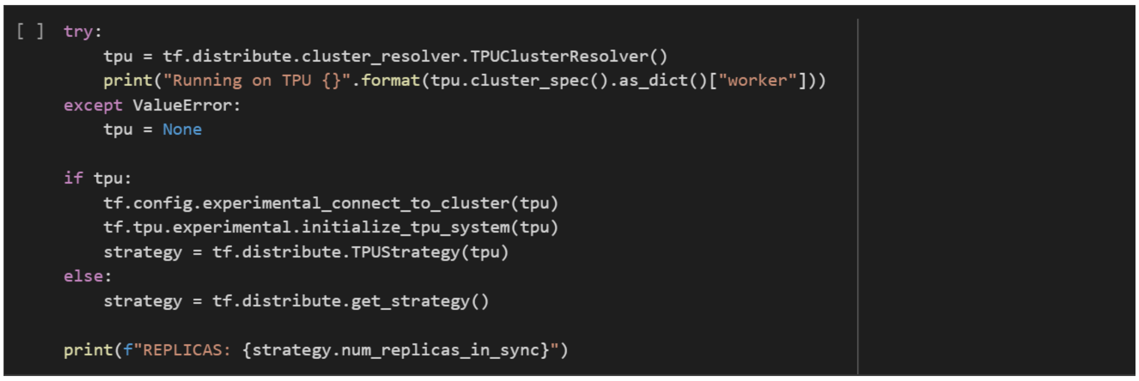


```
[ ] from __future__ import absolute_import, division, print_function, unicode_literals

    import sys

    import tensorflow as tf

    tf.random.set_seed(1234)
    AUTO = tf.data.experimental.AUTOTUNE

    import tensorflow_datasets as tfds

    import os
    import re
    import numpy as np
    from time import time
    import matplotlib.pyplot as plt
    import gradio as gr

    print("Tensorflow version {}".format(tf.__version__))
```



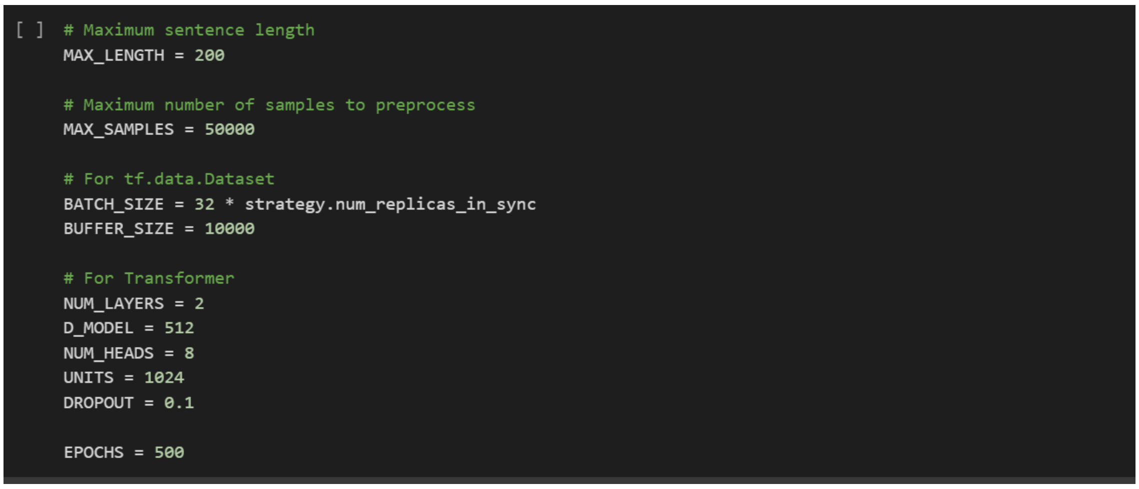
```
[ ]  import pandas as pd
```

Figure B1, B2 & B3: Installation of required libraries



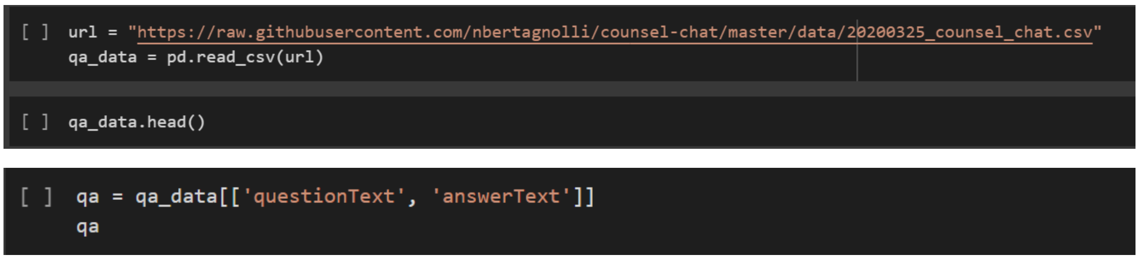
```
[ ] try:
        tpu = tf.distribute.cluster_resolver.TPUClusterResolver()
        print("Running on TPU {}".format(tpu.cluster_spec().as_dict()["worker"]))
    except ValueError:
        tpu = None

    if tpu:
        tf.config.experimental_connect_to_cluster(tpu)
        tf.tpu.experimental.initialize_tpu_system(tpu)
        strategy = tf.distribute.TPUStrategy(tpu)
    else:
        strategy = tf.distribute.get_strategy()

    print(f"REPLICAS: {strategy.num_replicas_in_sync}")
```

Figure B4: Initialization of TPU



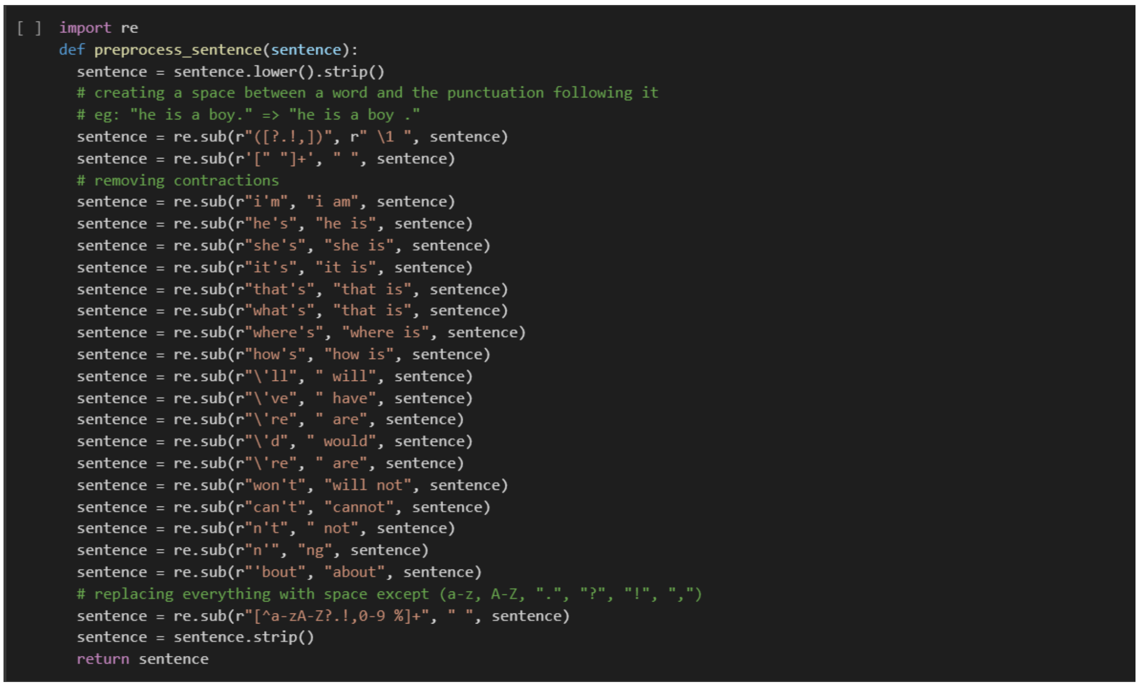
```
[ ] # Maximum sentence length
    MAX_LENGTH = 200

    # Maximum number of samples to preprocess
    MAX_SAMPLES = 50000

    # For tf.data.Dataset
    BATCH_SIZE = 32 * strategy.num_replicas_in_sync
    BUFFER_SIZE = 10000

    # For Transformer
    NUM_LAYERS = 2
    D_MODEL = 512
    NUM_HEADS = 8
    UNITS = 1024
    DROPOUT = 0.1

    EPOCHS = 500
```

Figure B5: Setting up hyperparameters for model training

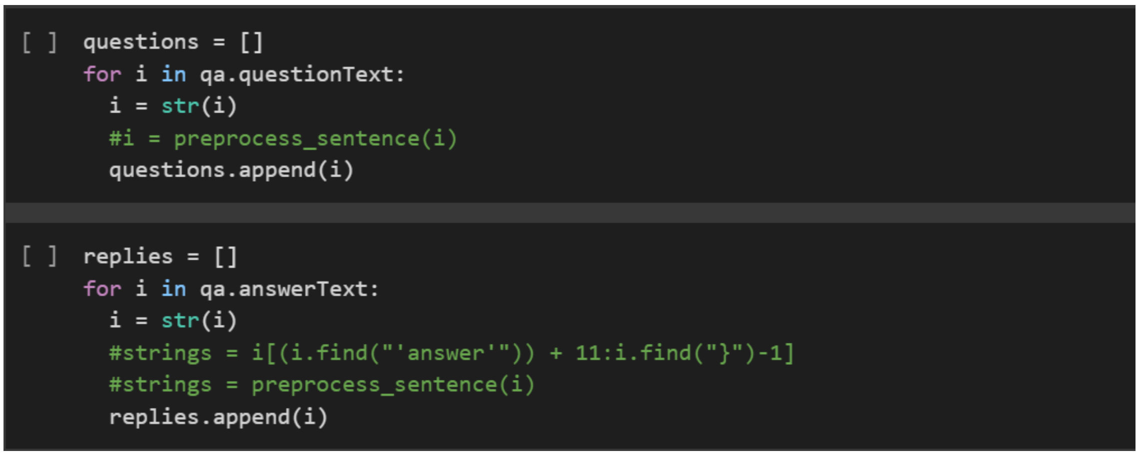


```
[ ] url = "https://raw.githubusercontent.com/nbertagnolli/counsel-chat/master/data/20200325_counsel_chat.csv"
    qa_data = pd.read_csv(url)

[ ] qa_data.head()

[ ] qa = qa_data[['questionText', 'answerText']]
    qa
```

Figure B6 & B7: Obtaining dataset, display, and selecting only required columns



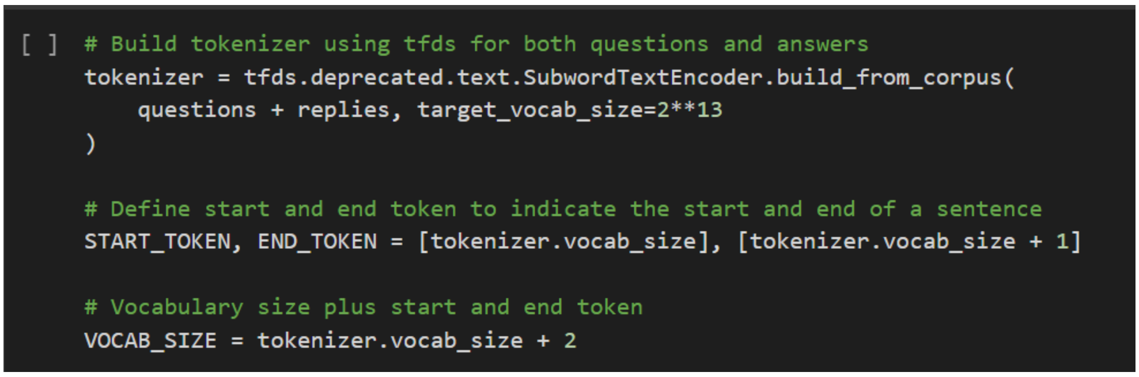
```
[ ] import re
    def preprocess_sentence(sentence):
      sentence = sentence.lower().strip()
      # creating a space between a word and the punctuation following it
      # eg: "he is a boy." => "he is a boy ."
      sentence = re.sub(r"([?.!,])", r" \1 ", sentence)
      sentence = re.sub(r'[" "]+', " ", sentence)
      # removing contractions
      sentence = re.sub(r"i'm", "i am", sentence)
      sentence = re.sub(r"he's", "he is", sentence)
      sentence = re.sub(r"she's", "she is", sentence)
      sentence = re.sub(r"it's", "it is", sentence)
      sentence = re.sub(r"that's", "that is", sentence)
      sentence = re.sub(r"what's", "that is", sentence)
      sentence = re.sub(r"where's", "where is", sentence)
      sentence = re.sub(r"how's", "how is", sentence)
      sentence = re.sub(r"\'ll", " will", sentence)
      sentence = re.sub(r"\'ve", " have", sentence)
      sentence = re.sub(r"\'re", " are", sentence)
      sentence = re.sub(r"\'d", " would", sentence)
      sentence = re.sub(r"\'re", " are", sentence)
      sentence = re.sub(r"won't", "will not", sentence)
      sentence = re.sub(r"can't", "cannot", sentence)
      sentence = re.sub(r"n't", " not", sentence)
      sentence = re.sub(r"n'", "ng", sentence)
      sentence = re.sub(r"'bout", "about", sentence)
      # replacing everything with space except (a-z, A-Z, ".", "?", "!", ",")
      sentence = re.sub(r"[^a-zA-Z?.!,0-9 %]+", " ", sentence)
      sentence = sentence.strip()
      return sentence
```



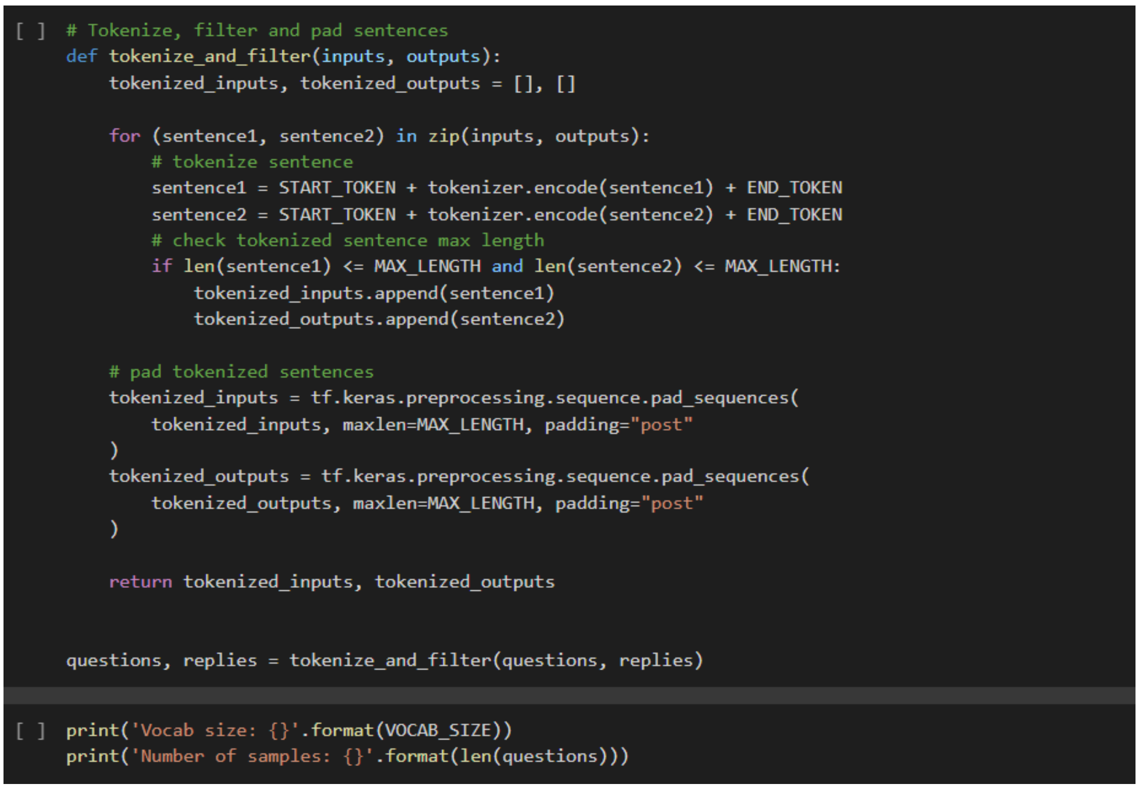
```
[ ]  questions = []
     for i in qa.questionText:
       i = str(i)
       #i = preprocess_sentence(i)
       questions.append(i)
```

```
[ ]  replies = []
     for i in qa.answerText:
       i = str(i)
       #strings = i[(i.find("'answer'")) + 11:i.find("}")-1]
       #strings = preprocess_sentence(i)
       replies.append(i)
```

Figure B8 & B9: Preprocess data



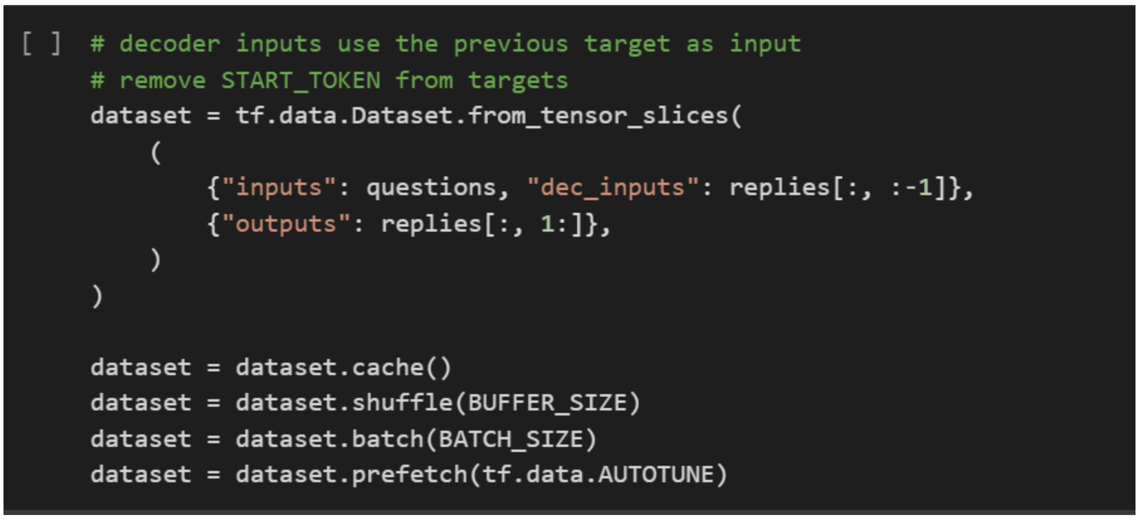
```
[ ]  # Build tokenizer using tfds for both questions and answers
     tokenizer = tfds.deprecated.text.SubwordTextEncoder.build_from_corpus(
         questions + replies, target_vocab_size=2**13
     )

     # Define start and end token to indicate the start and end of a sentence
     START_TOKEN, END_TOKEN = [tokenizer.vocab_size], [tokenizer.vocab_size + 1]

     # Vocabulary size plus start and end token
     VOCAB_SIZE = tokenizer.vocab_size + 2
```



```
[ ]  # Tokenize, filter and pad sentences
     def tokenize_and_filter(inputs, outputs):
         tokenized_inputs, tokenized_outputs = [], []

         for (sentence1, sentence2) in zip(inputs, outputs):
             # tokenize sentence
             sentence1 = START_TOKEN + tokenizer.encode(sentence1) + END_TOKEN
             sentence2 = START_TOKEN + tokenizer.encode(sentence2) + END_TOKEN
             # check tokenized sentence max length
             if len(sentence1) <= MAX_LENGTH and len(sentence2) <= MAX_LENGTH:
                 tokenized_inputs.append(sentence1)
                 tokenized_outputs.append(sentence2)

         # pad tokenized sentences
         tokenized_inputs = tf.keras.preprocessing.sequence.pad_sequences(
             tokenized_inputs, maxlen=MAX_LENGTH, padding="post"
         )
         tokenized_outputs = tf.keras.preprocessing.sequence.pad_sequences(
             tokenized_outputs, maxlen=MAX_LENGTH, padding="post"
         )

         return tokenized_inputs, tokenized_outputs


     questions, replies = tokenize_and_filter(questions, replies)
```

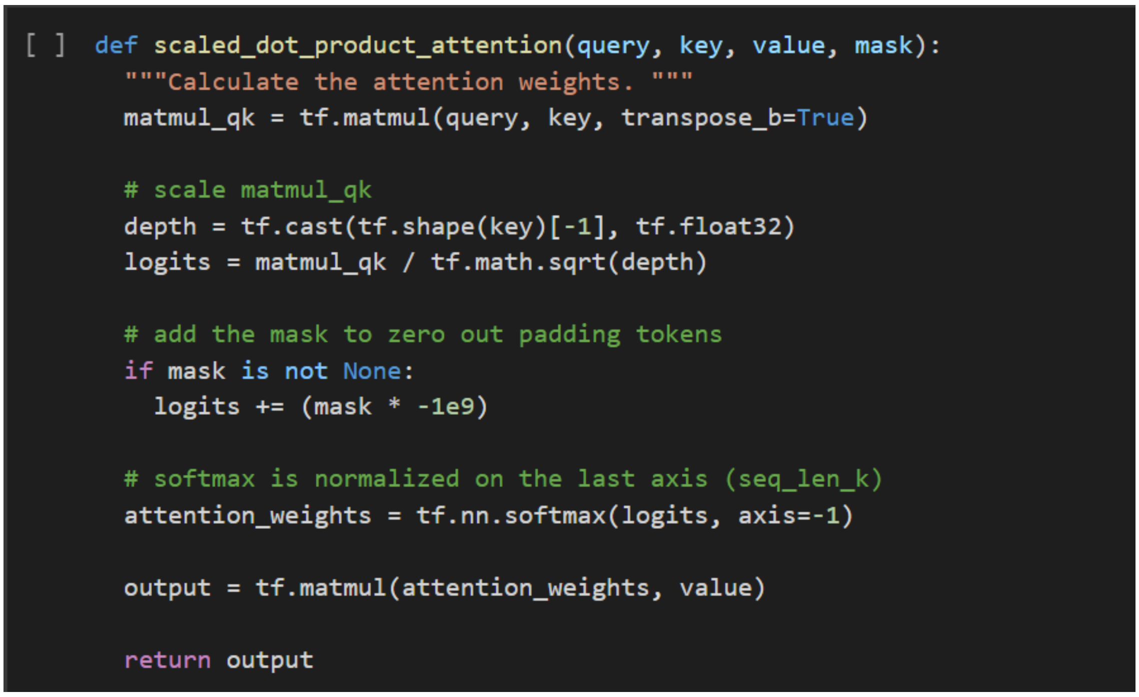
```
[ ]  print('Vocab size: {}'.format(VOCAB_SIZE))
     print('Number of samples: {}'.format(len(questions)))
```

Figure B10 & B11: Building tokenizer for questions and answers and filtering the maximum length of the tokenized sentence



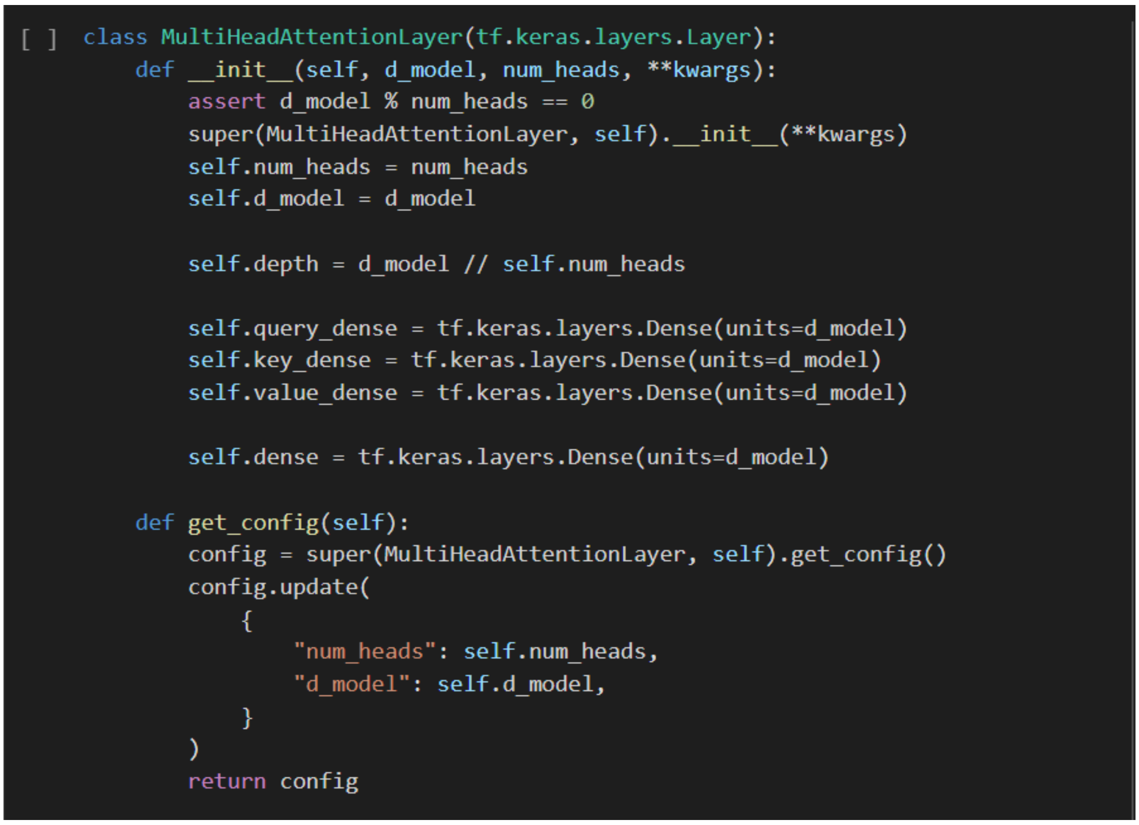
```
[ ]  # decoder inputs use the previous target as input
     # remove START_TOKEN from targets
     dataset = tf.data.Dataset.from_tensor_slices(
         (
             {"inputs": questions, "dec_inputs": replies[:, :-1]},
             {"outputs": replies[:, 1:]},
         )
     )

     dataset = dataset.cache()
     dataset = dataset.shuffle(BUFFER_SIZE)
     dataset = dataset.batch(BATCH_SIZE)
     dataset = dataset.prefetch(tf.data.AUTOTUNE)
```

Figure B12: Creating dataset for model training

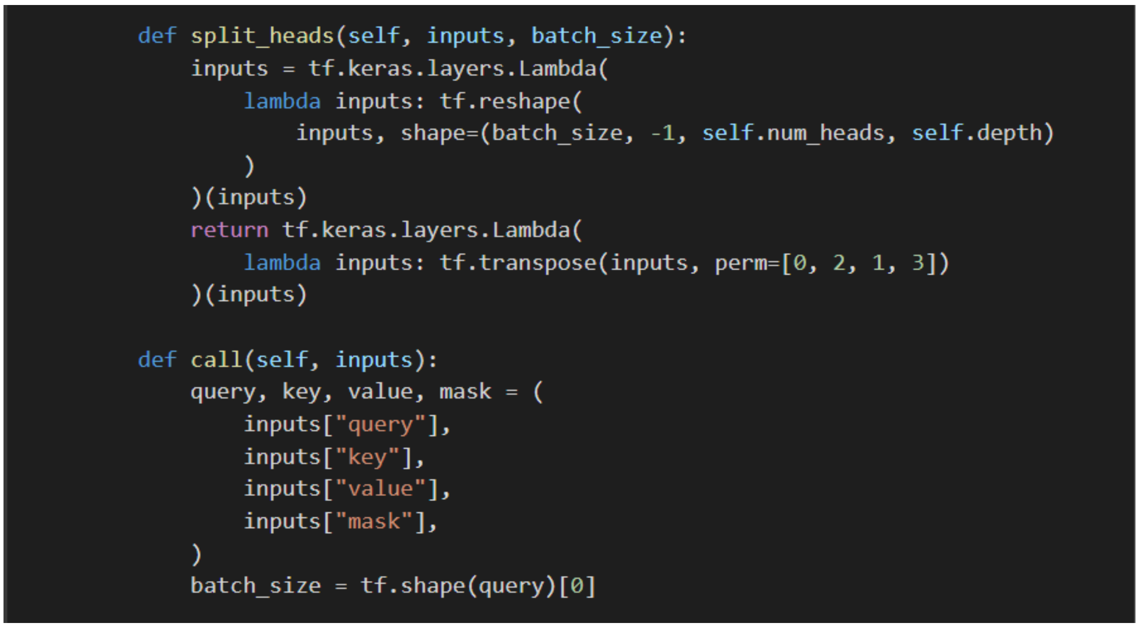


```
[ ]  def scaled_dot_product_attention(query, key, value, mask):
       """Calculate the attention weights. """
       matmul_qk = tf.matmul(query, key, transpose_b=True)

       # scale matmul_qk
       depth = tf.cast(tf.shape(key)[-1], tf.float32)
       logits = matmul_qk / tf.math.sqrt(depth)

       # add the mask to zero out padding tokens
       if mask is not None:
         logits += (mask * -1e9)

       # softmax is normalized on the last axis (seq_len_k)
       attention_weights = tf.nn.softmax(logits, axis=-1)

       output = tf.matmul(attention_weights, value)

       return output
```

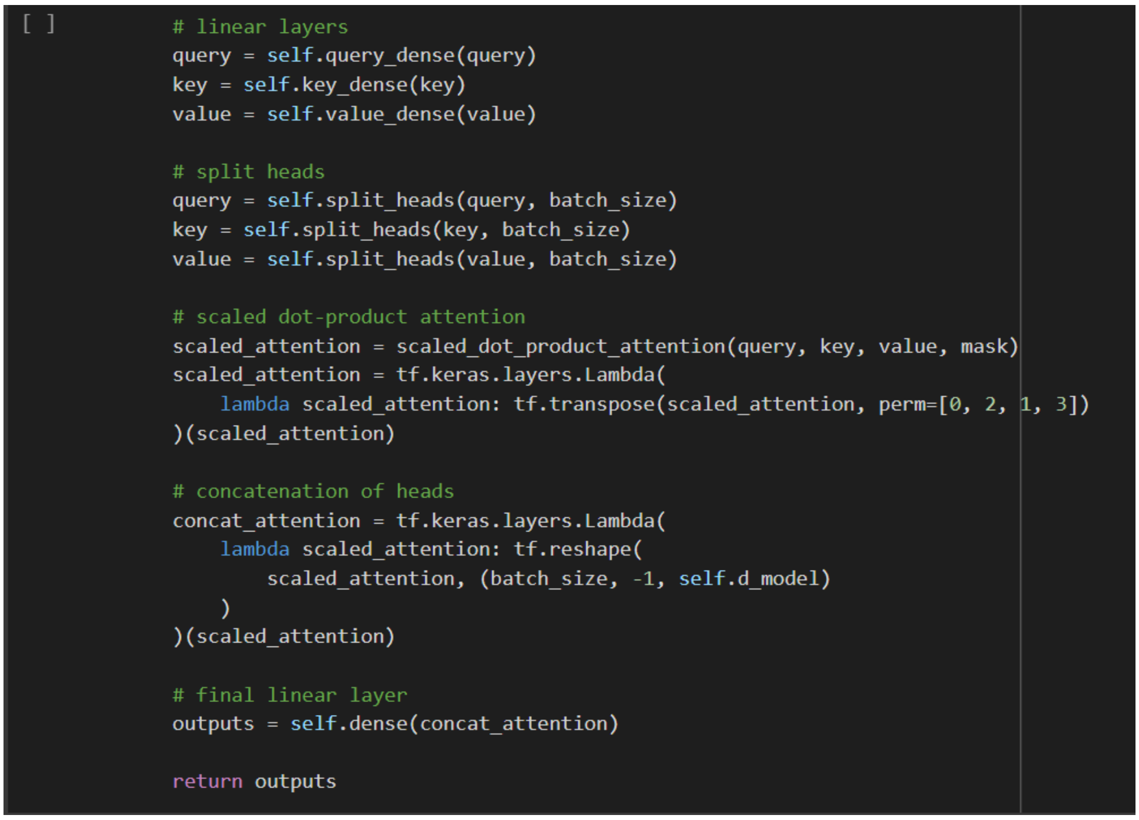
Figure B13: Scaled Dot Product Attention



```
[ ]  class MultiHeadAttentionLayer(tf.keras.layers.Layer):
         def __init__(self, d_model, num_heads, **kwargs):
             assert d_model % num_heads == 0
             super(MultiHeadAttentionLayer, self).__init__(**kwargs)
             self.num_heads = num_heads
             self.d_model = d_model

             self.depth = d_model // self.num_heads

             self.query_dense = tf.keras.layers.Dense(units=d_model)
             self.key_dense = tf.keras.layers.Dense(units=d_model)
             self.value_dense = tf.keras.layers.Dense(units=d_model)

             self.dense = tf.keras.layers.Dense(units=d_model)

         def get_config(self):
             config = super(MultiHeadAttentionLayer, self).get_config()
             config.update(
                 {
                     "num_heads": self.num_heads,
                     "d_model": self.d_model,
                 }
             )
             return config
```

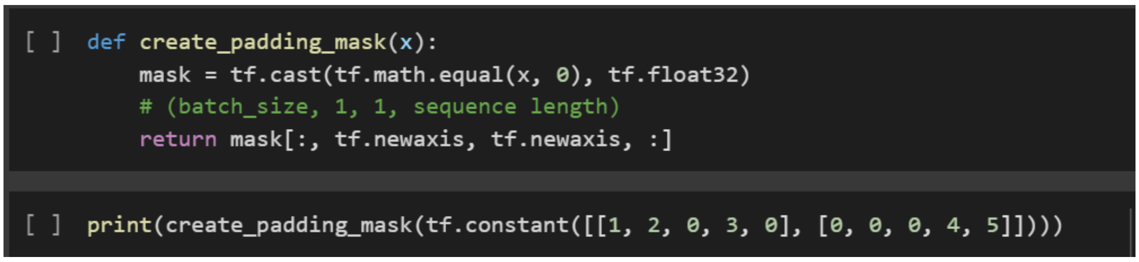


```
         def split_heads(self, inputs, batch_size):
             inputs = tf.keras.layers.Lambda(
                 lambda inputs: tf.reshape(
                     inputs, shape=(batch_size, -1, self.num_heads, self.depth)
                 )
             )(inputs)
             return tf.keras.layers.Lambda(
                 lambda inputs: tf.transpose(inputs, perm=[0, 2, 1, 3])
             )(inputs)

         def call(self, inputs):
             query, key, value, mask = (
                 inputs["query"],
                 inputs["key"],
                 inputs["value"],
                 inputs["mask"],
             )
             batch_size = tf.shape(query)[0]
```



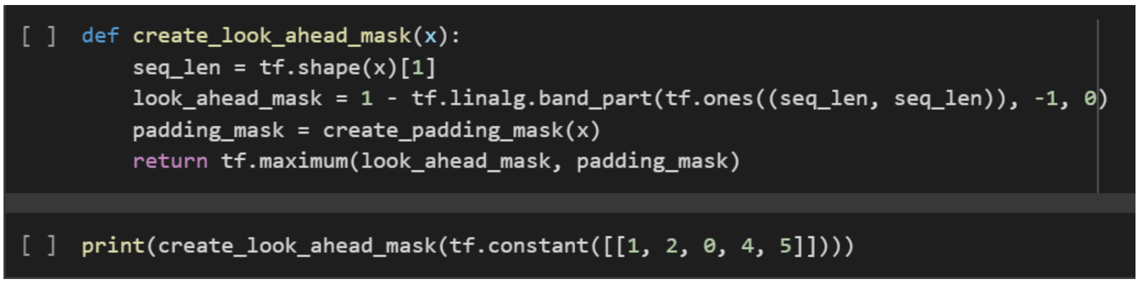
```
        # linear layers
        query = self.query_dense(query)
        key = self.key_dense(key)
        value = self.value_dense(value)

        # split heads
        query = self.split_heads(query, batch_size)
        key = self.split_heads(key, batch_size)
        value = self.split_heads(value, batch_size)

        # scaled dot-product attention
        scaled_attention = scaled_dot_product_attention(query, key, value, mask)
        scaled_attention = tf.keras.layers.Lambda(
            lambda scaled_attention: tf.transpose(scaled_attention, perm=[0, 2, 1, 3])
        )(scaled_attention)

        # concatenation of heads
        concat_attention = tf.keras.layers.Lambda(
            lambda scaled_attention: tf.reshape(
                scaled_attention, (batch_size, -1, self.d_model)
            )
        )(scaled_attention)

        # final linear layer
        outputs = self.dense(concat_attention)

        return outputs
```

Figure B14, B15 & B16: Multi-headed Attention



```
def create_padding_mask(x):
    mask = tf.cast(tf.math.equal(x, 0), tf.float32)
    # (batch_size, 1, 1, sequence length)
    return mask[:, tf.newaxis, tf.newaxis, :]
```

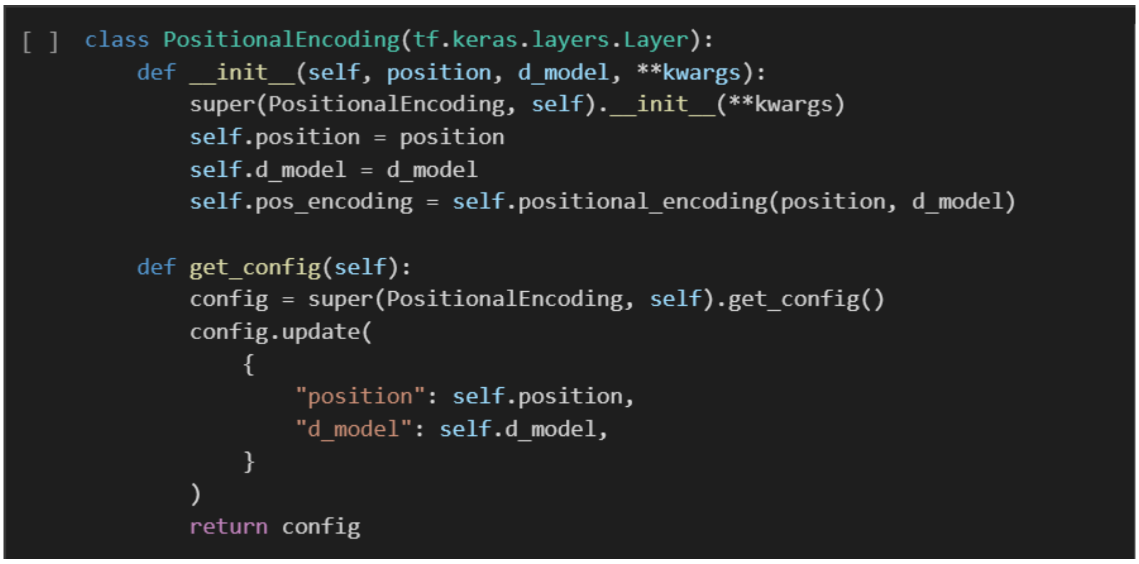
```
print(create_padding_mask(tf.constant([[1, 2, 0, 3, 0], [0, 0, 0, 4, 5]])))
```



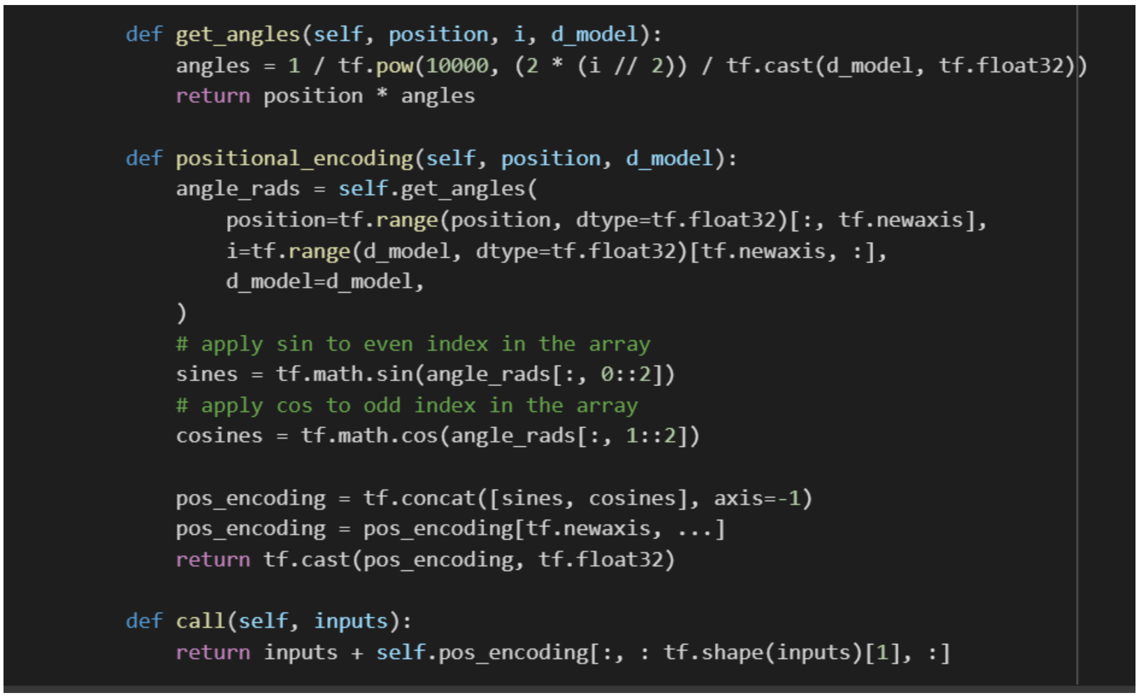
```
def create_look_ahead_mask(x):
    seq_len = tf.shape(x)[1]
    look_ahead_mask = 1 - tf.linalg.band_part(tf.ones((seq_len, seq_len)), -1, 0)
    padding_mask = create_padding_mask(x)
    return tf.maximum(look_ahead_mask, padding_mask)
```

```
print(create_look_ahead_mask(tf.constant([[1, 2, 0, 4, 5]])))
```

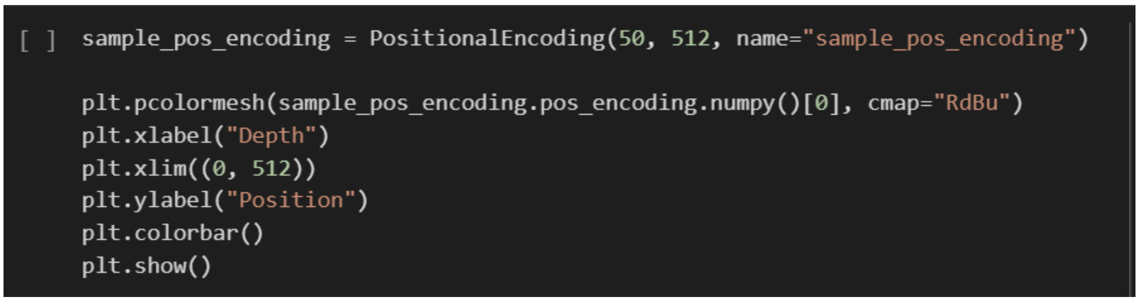
Figure B17 & B18: Create padding mask and look-ahead mask



```
[ ]  class PositionalEncoding(tf.keras.layers.Layer):
         def __init__(self, position, d_model, **kwargs):
             super(PositionalEncoding, self).__init__(**kwargs)
             self.position = position
             self.d_model = d_model
             self.pos_encoding = self.positional_encoding(position, d_model)

         def get_config(self):
             config = super(PositionalEncoding, self).get_config()
             config.update(
                 {
                     "position": self.position,
                     "d_model": self.d_model,
                 }
             )
             return config
```

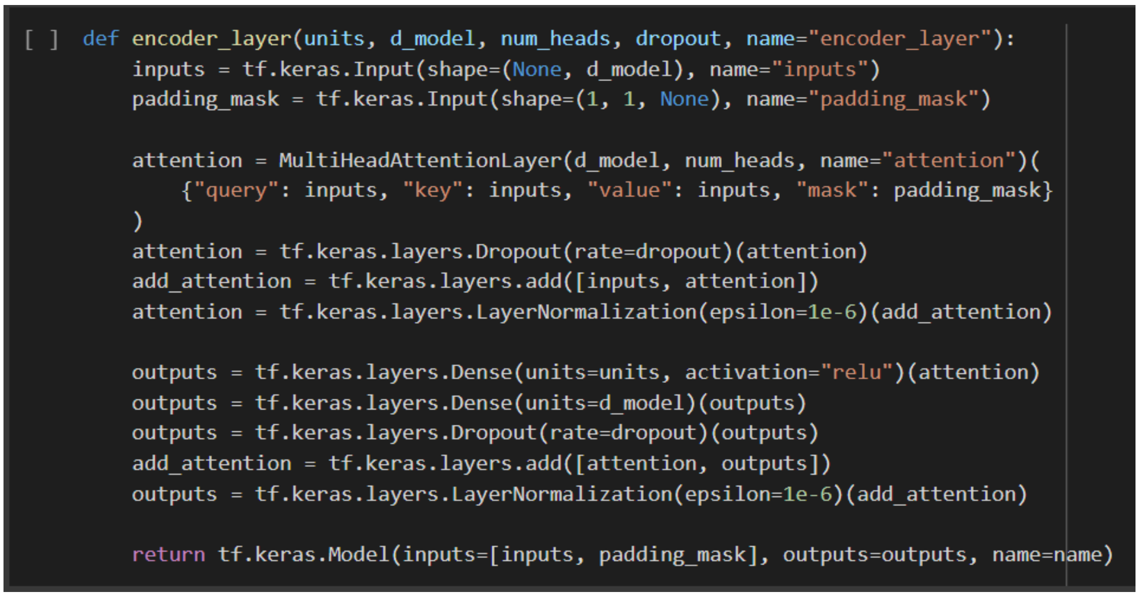


```
    def get_angles(self, position, i, d_model):
        angles = 1 / tf.pow(10000, (2 * (i // 2)) / tf.cast(d_model, tf.float32))
        return position * angles

    def positional_encoding(self, position, d_model):
        angle_rads = self.get_angles(
            position=tf.range(position, dtype=tf.float32)[:, tf.newaxis],
            i=tf.range(d_model, dtype=tf.float32)[tf.newaxis, :],
            d_model=d_model,
        )
        # apply sin to even index in the array
        sines = tf.math.sin(angle_rads[:, 0::2])
        # apply cos to odd index in the array
        cosines = tf.math.cos(angle_rads[:, 1::2])

        pos_encoding = tf.concat([sines, cosines], axis=-1)
        pos_encoding = pos_encoding[tf.newaxis, ...]
        return tf.cast(pos_encoding, tf.float32)

    def call(self, inputs):
        return inputs + self.pos_encoding[:, : tf.shape(inputs)[1], :]
```

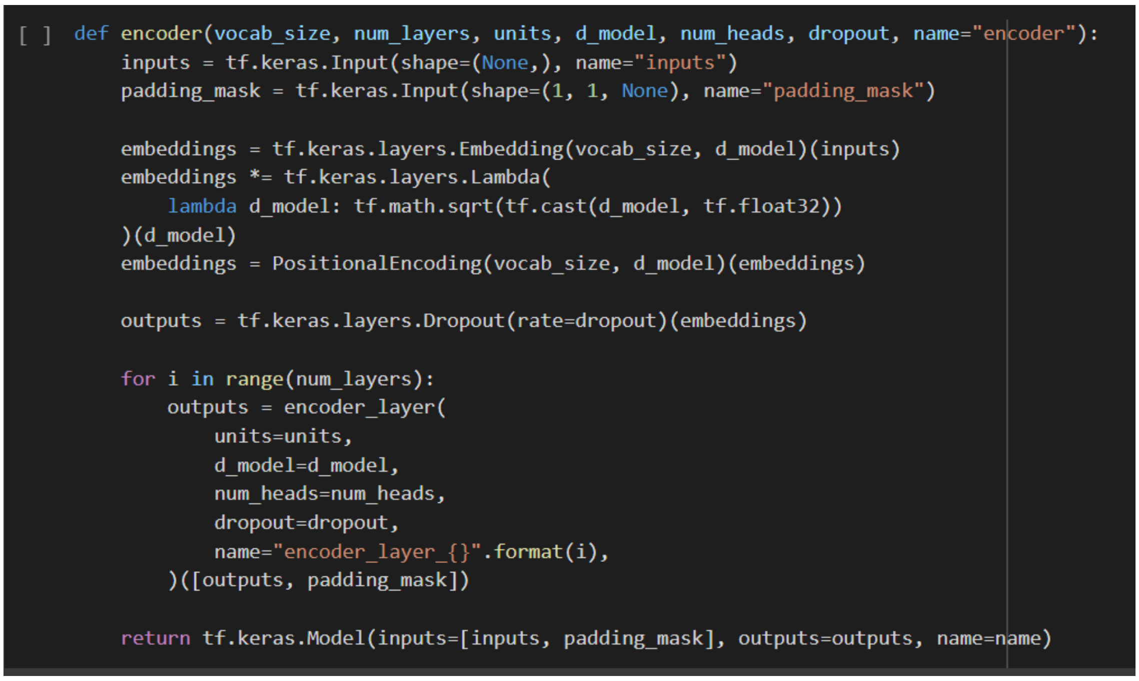


```
[ ]  sample_pos_encoding = PositionalEncoding(50, 512, name="sample_pos_encoding")

     plt.pcolormesh(sample_pos_encoding.pos_encoding.numpy()[0], cmap="RdBu")
     plt.xlabel("Depth")
     plt.xlim((0, 512))
     plt.ylabel("Position")
     plt.colorbar()
     plt.show()
```

Figure B19, B20 & B21: Positional encoding

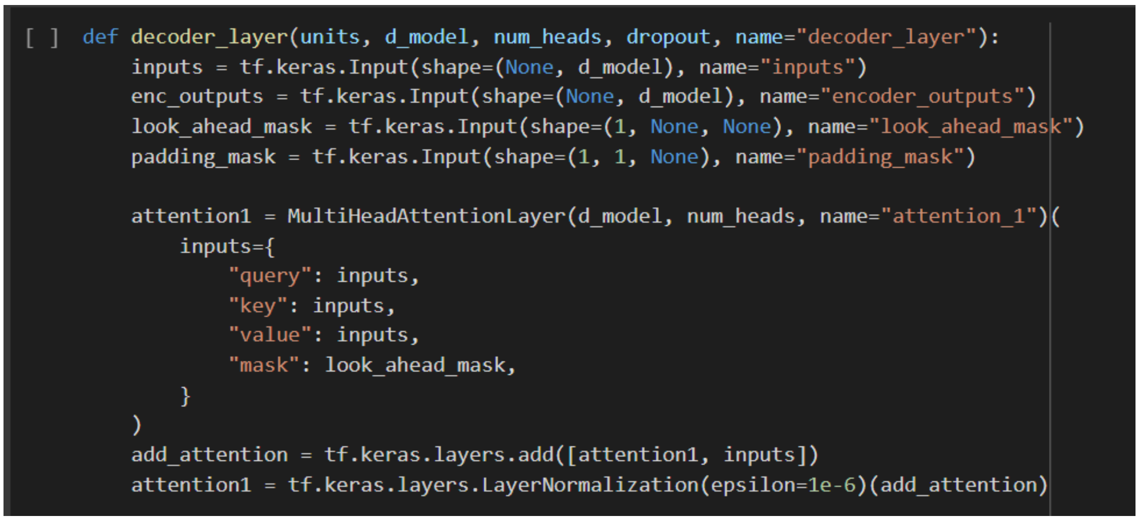


```
[ ]  def encoder_layer(units, d_model, num_heads, dropout, name="encoder_layer"):
         inputs = tf.keras.Input(shape=(None, d_model), name="inputs")
         padding_mask = tf.keras.Input(shape=(1, 1, None), name="padding_mask")

         attention = MultiHeadAttentionLayer(d_model, num_heads, name="attention")(
             {"query": inputs, "key": inputs, "value": inputs, "mask": padding_mask}
         )
         attention = tf.keras.layers.Dropout(rate=dropout)(attention)
         add_attention = tf.keras.layers.add([inputs, attention])
         attention = tf.keras.layers.LayerNormalization(epsilon=1e-6)(add_attention)

         outputs = tf.keras.layers.Dense(units=units, activation="relu")(attention)
         outputs = tf.keras.layers.Dense(units=d_model)(outputs)
         outputs = tf.keras.layers.Dropout(rate=dropout)(outputs)
         add_attention = tf.keras.layers.add([attention, outputs])
         outputs = tf.keras.layers.LayerNormalization(epsilon=1e-6)(add_attention)

         return tf.keras.Model(inputs=[inputs, padding_mask], outputs=outputs, name=name)
```

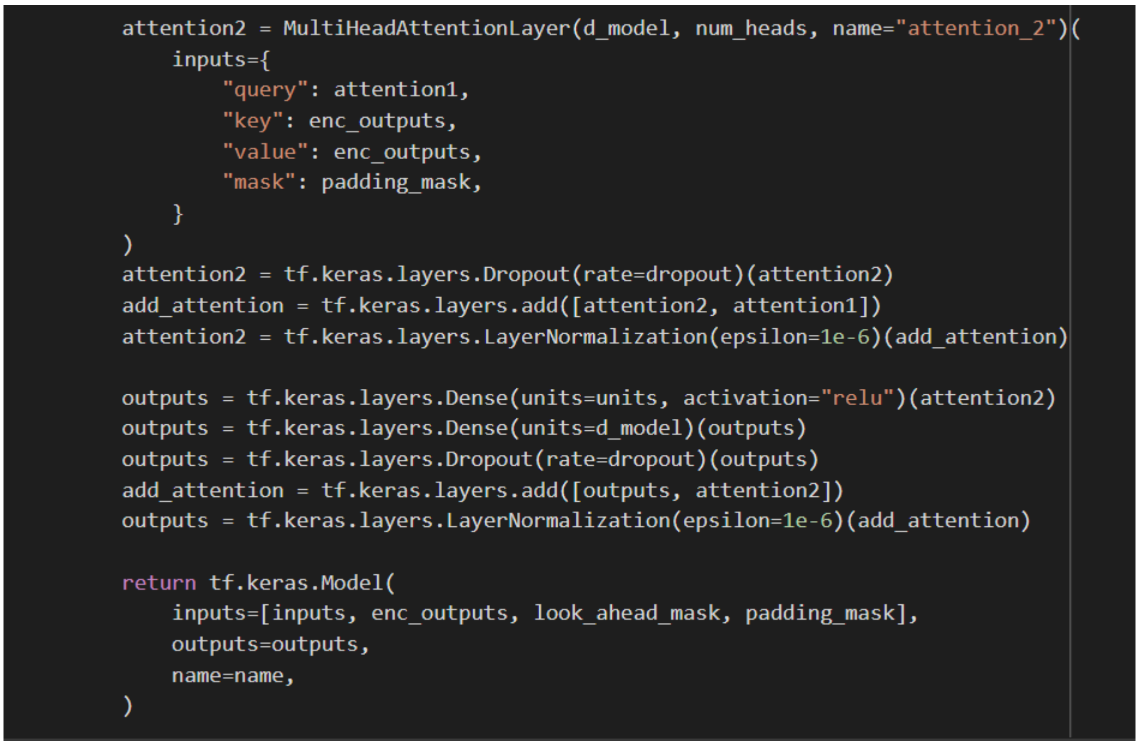


```
[ ]  def encoder(vocab_size, num_layers, units, d_model, num_heads, dropout, name="encoder"):
         inputs = tf.keras.Input(shape=(None,), name="inputs")
         padding_mask = tf.keras.Input(shape=(1, 1, None), name="padding_mask")

         embeddings = tf.keras.layers.Embedding(vocab_size, d_model)(inputs)
         embeddings *= tf.keras.layers.Lambda(
             lambda d_model: tf.math.sqrt(tf.cast(d_model, tf.float32))
         )(d_model)
         embeddings = PositionalEncoding(vocab_size, d_model)(embeddings)

         outputs = tf.keras.layers.Dropout(rate=dropout)(embeddings)

         for i in range(num_layers):
             outputs = encoder_layer(
                 units=units,
                 d_model=d_model,
                 num_heads=num_heads,
                 dropout=dropout,
                 name="encoder_layer_{}".format(i),
             )([outputs, padding_mask])

         return tf.keras.Model(inputs=[inputs, padding_mask], outputs=outputs, name=name)
```

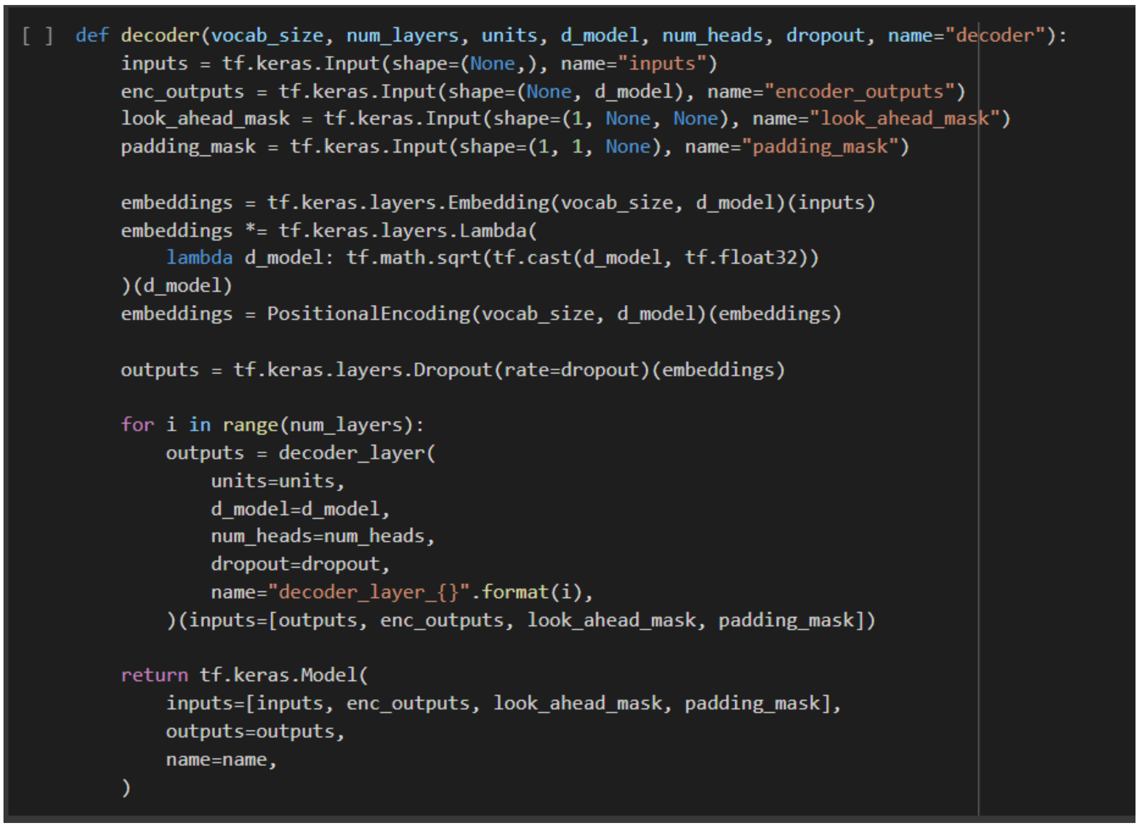
Figure B22 & B23: Encoder Layer and Encoder



```
[ ]  def decoder_layer(units, d_model, num_heads, dropout, name="decoder_layer"):
         inputs = tf.keras.Input(shape=(None, d_model), name="inputs")
         enc_outputs = tf.keras.Input(shape=(None, d_model), name="encoder_outputs")
         look_ahead_mask = tf.keras.Input(shape=(1, None, None), name="look_ahead_mask")
         padding_mask = tf.keras.Input(shape=(1, 1, None), name="padding_mask")

         attention1 = MultiHeadAttentionLayer(d_model, num_heads, name="attention_1")(
             inputs={
                 "query": inputs,
                 "key": inputs,
                 "value": inputs,
                 "mask": look_ahead_mask,
             }
         )
         add_attention = tf.keras.layers.add([attention1, inputs])
         attention1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)(add_attention)
```

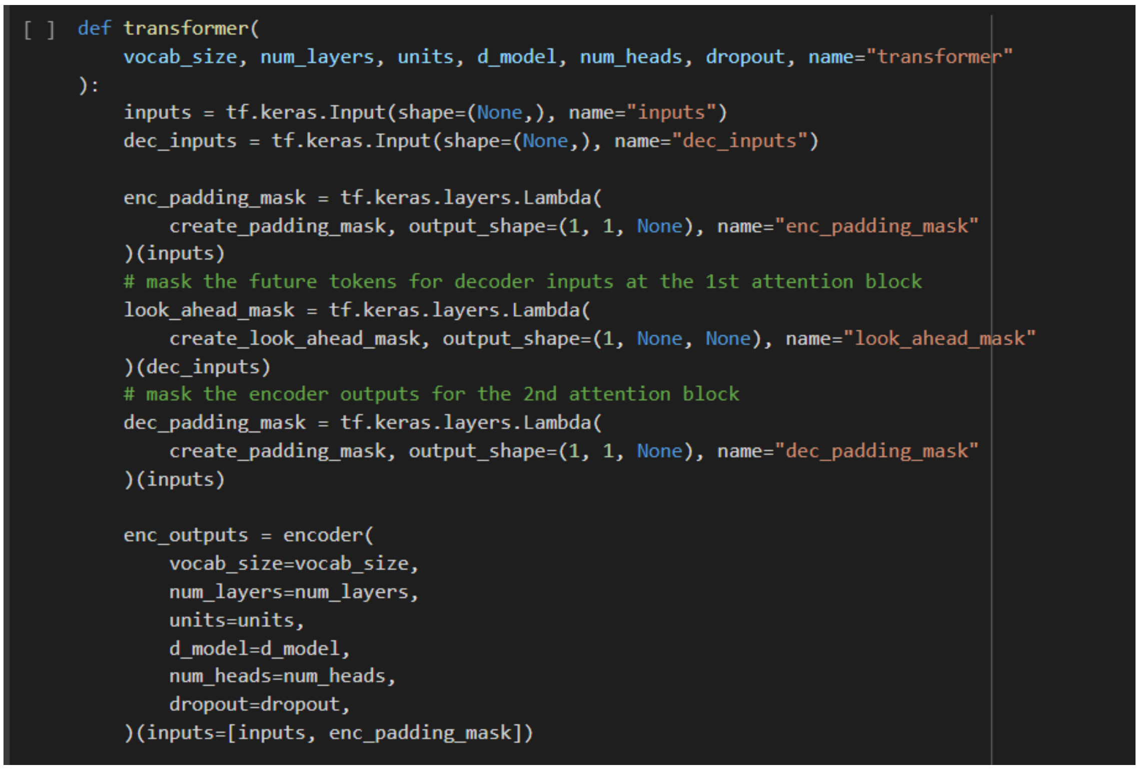


```
         attention2 = MultiHeadAttentionLayer(d_model, num_heads, name="attention_2")(
             inputs={
                 "query": attention1,
                 "key": enc_outputs,
                 "value": enc_outputs,
                 "mask": padding_mask,
             }
         )
         attention2 = tf.keras.layers.Dropout(rate=dropout)(attention2)
         add_attention = tf.keras.layers.add([attention2, attention1])
         attention2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)(add_attention)

         outputs = tf.keras.layers.Dense(units=units, activation="relu")(attention2)
         outputs = tf.keras.layers.Dense(units=d_model)(outputs)
         outputs = tf.keras.layers.Dropout(rate=dropout)(outputs)
         add_attention = tf.keras.layers.add([outputs, attention2])
         outputs = tf.keras.layers.LayerNormalization(epsilon=1e-6)(add_attention)

         return tf.keras.Model(
             inputs=[inputs, enc_outputs, look_ahead_mask, padding_mask],
             outputs=outputs,
             name=name,
         )
```

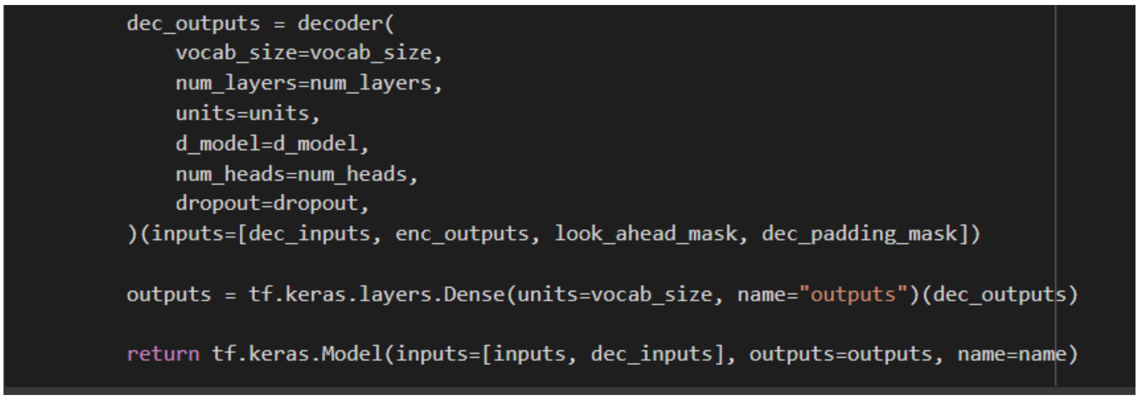


```
[ ] def decoder(vocab_size, num_layers, units, d_model, num_heads, dropout, name="decoder"):
        inputs = tf.keras.Input(shape=(None,), name="inputs")
        enc_outputs = tf.keras.Input(shape=(None, d_model), name="encoder_outputs")
        look_ahead_mask = tf.keras.Input(shape=(1, None, None), name="look_ahead_mask")
        padding_mask = tf.keras.Input(shape=(1, 1, None), name="padding_mask")

        embeddings = tf.keras.layers.Embedding(vocab_size, d_model)(inputs)
        embeddings *= tf.keras.layers.Lambda(
            lambda d_model: tf.math.sqrt(tf.cast(d_model, tf.float32))
        )(d_model)
        embeddings = PositionalEncoding(vocab_size, d_model)(embeddings)

        outputs = tf.keras.layers.Dropout(rate=dropout)(embeddings)

        for i in range(num_layers):
            outputs = decoder_layer(
                units=units,
                d_model=d_model,
                num_heads=num_heads,
                dropout=dropout,
                name="decoder_layer_{}".format(i),
            )(inputs=[outputs, enc_outputs, look_ahead_mask, padding_mask])

        return tf.keras.Model(
            inputs=[inputs, enc_outputs, look_ahead_mask, padding_mask],
            outputs=outputs,
            name=name,
        )
```

Figure B24, B25 & B26: Decoder Layer and Decoder

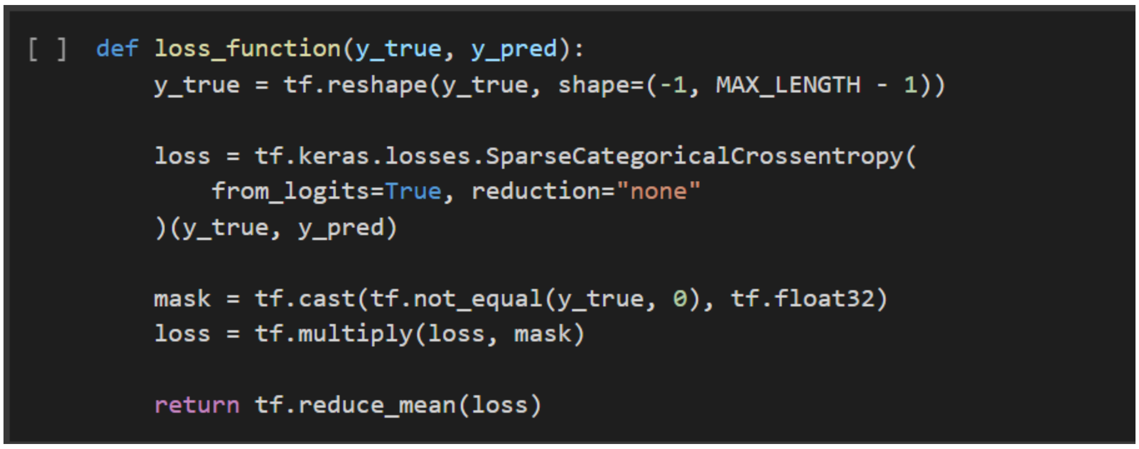


```
[ ] def transformer(
        vocab_size, num_layers, units, d_model, num_heads, dropout, name="transformer"
    ):
        inputs = tf.keras.Input(shape=(None,), name="inputs")
        dec_inputs = tf.keras.Input(shape=(None,), name="dec_inputs")

        enc_padding_mask = tf.keras.layers.Lambda(
            create_padding_mask, output_shape=(1, 1, None), name="enc_padding_mask"
        )(inputs)
        # mask the future tokens for decoder inputs at the 1st attention block
        look_ahead_mask = tf.keras.layers.Lambda(
            create_look_ahead_mask, output_shape=(1, None, None), name="look_ahead_mask"
        )(dec_inputs)
        # mask the encoder outputs for the 2nd attention block
        dec_padding_mask = tf.keras.layers.Lambda(
            create_padding_mask, output_shape=(1, 1, None), name="dec_padding_mask"
        )(inputs)

        enc_outputs = encoder(
            vocab_size=vocab_size,
            num_layers=num_layers,
            units=units,
            d_model=d_model,
            num_heads=num_heads,
            dropout=dropout,
        )(inputs=[inputs, enc_padding_mask])
```



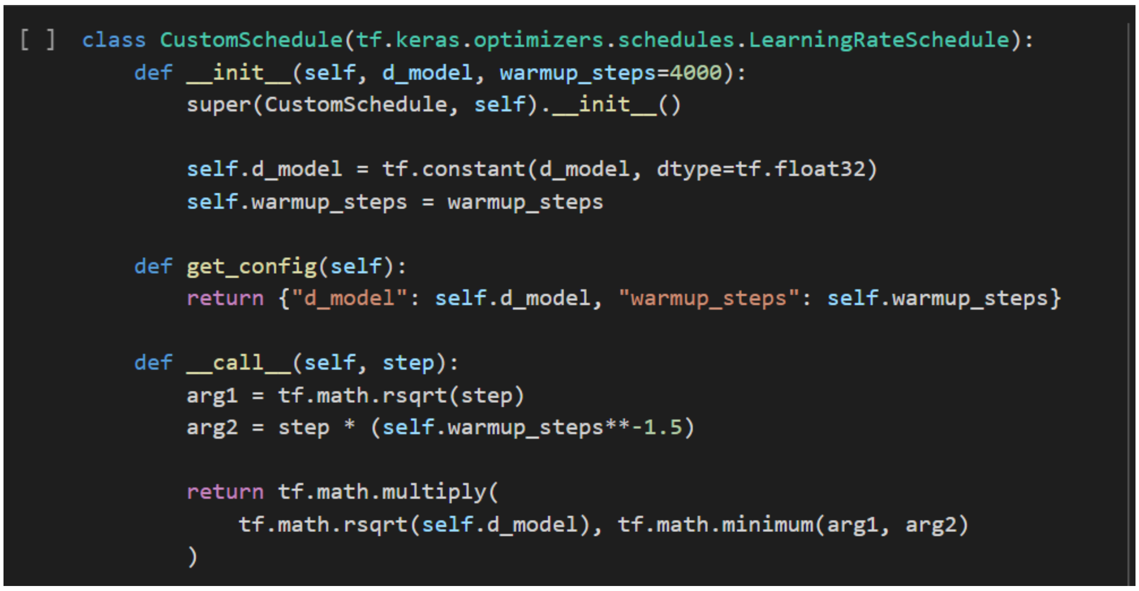
```
    dec_outputs = decoder(
        vocab_size=vocab_size,
        num_layers=num_layers,
        units=units,
        d_model=d_model,
        num_heads=num_heads,
        dropout=dropout,
    )(inputs=[dec_inputs, enc_outputs, look_ahead_mask, dec_padding_mask])

    outputs = tf.keras.layers.Dense(units=vocab_size, name="outputs")(dec_outputs)

    return tf.keras.Model(inputs=[inputs, dec_inputs], outputs=outputs, name=name)
```

Figure B27, B28 & B29: Transformer encoding and decoding



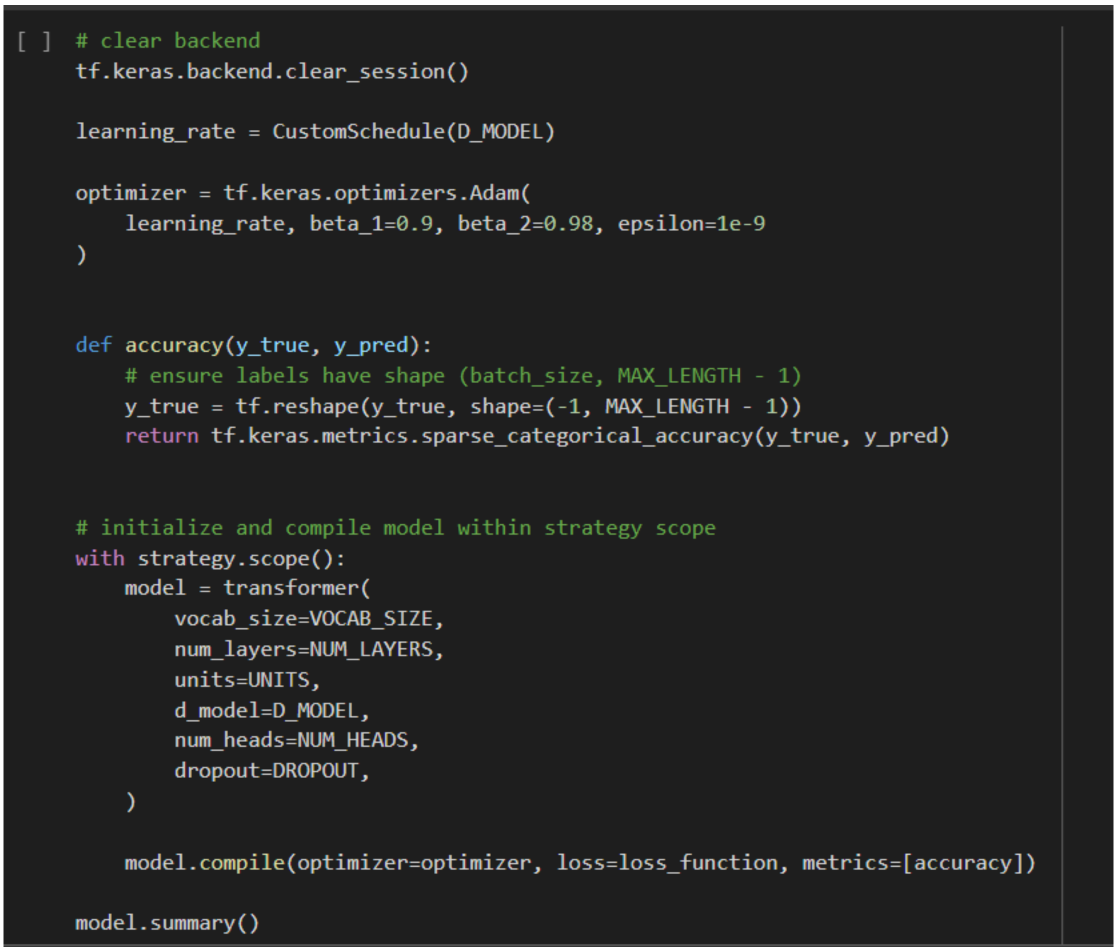
```
[ ]  def loss_function(y_true, y_pred):
         y_true = tf.reshape(y_true, shape=(-1, MAX_LENGTH - 1))

         loss = tf.keras.losses.SparseCategoricalCrossentropy(
             from_logits=True, reduction="none"
         )(y_true, y_pred)

         mask = tf.cast(tf.not_equal(y_true, 0), tf.float32)
         loss = tf.multiply(loss, mask)

         return tf.reduce_mean(loss)
```

Figure B30: Loss function



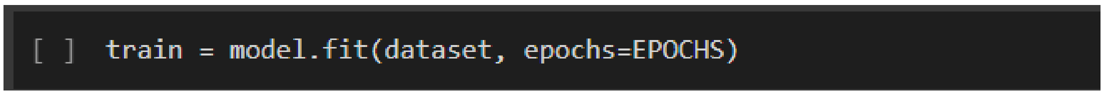
```
[ ]  class CustomSchedule(tf.keras.optimizers.schedules.LearningRateSchedule):
         def __init__(self, d_model, warmup_steps=4000):
             super(CustomSchedule, self).__init__()

             self.d_model = tf.constant(d_model, dtype=tf.float32)
             self.warmup_steps = warmup_steps

         def get_config(self):
             return {"d_model": self.d_model, "warmup_steps": self.warmup_steps}

         def __call__(self, step):
             arg1 = tf.math.rsqrt(step)
             arg2 = step * (self.warmup_steps**-1.5)

             return tf.math.multiply(
                 tf.math.rsqrt(self.d_model), tf.math.minimum(arg1, arg2)
             )
```

Figure B31: Building custom learning rate



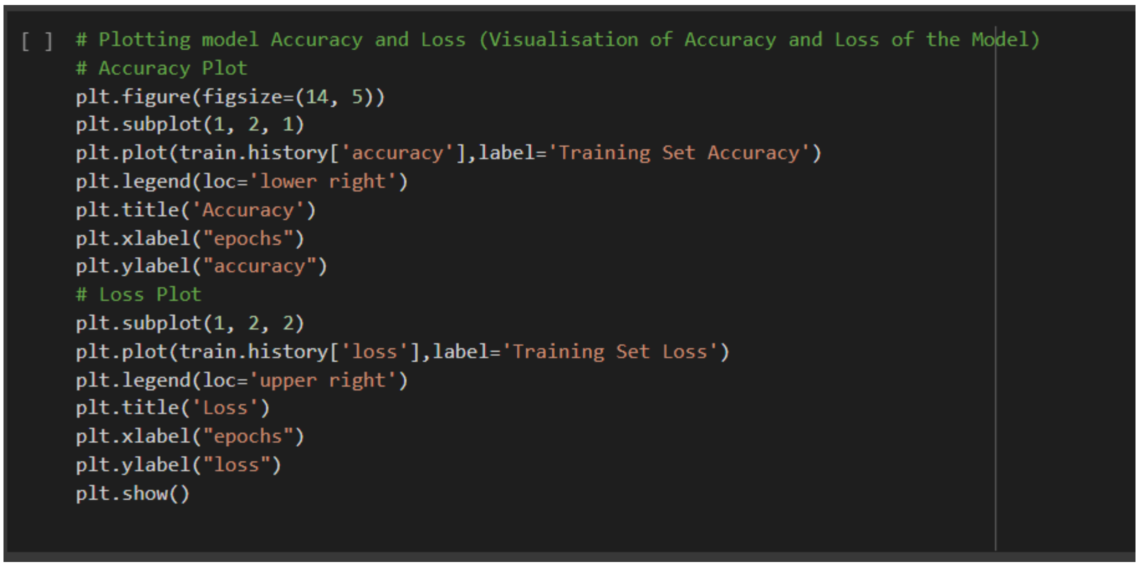
```
[ ] # clear backend
    tf.keras.backend.clear_session()

    learning_rate = CustomSchedule(D_MODEL)

    optimizer = tf.keras.optimizers.Adam(
        learning_rate, beta_1=0.9, beta_2=0.98, epsilon=1e-9
    )


    def accuracy(y_true, y_pred):
        # ensure labels have shape (batch_size, MAX_LENGTH - 1)
        y_true = tf.reshape(y_true, shape=(-1, MAX_LENGTH - 1))
        return tf.keras.metrics.sparse_categorical_accuracy(y_true, y_pred)


    # initialize and compile model within strategy scope
    with strategy.scope():
        model = transformer(
            vocab_size=VOCAB_SIZE,
            num_layers=NUM_LAYERS,
            units=UNITS,
            d_model=D_MODEL,
            num_heads=NUM_HEADS,
            dropout=DROPOUT,
        )

        model.compile(optimizer=optimizer, loss=loss_function, metrics=[accuracy])

    model.summary()
```

Figure B32: Initialization and compilation of model



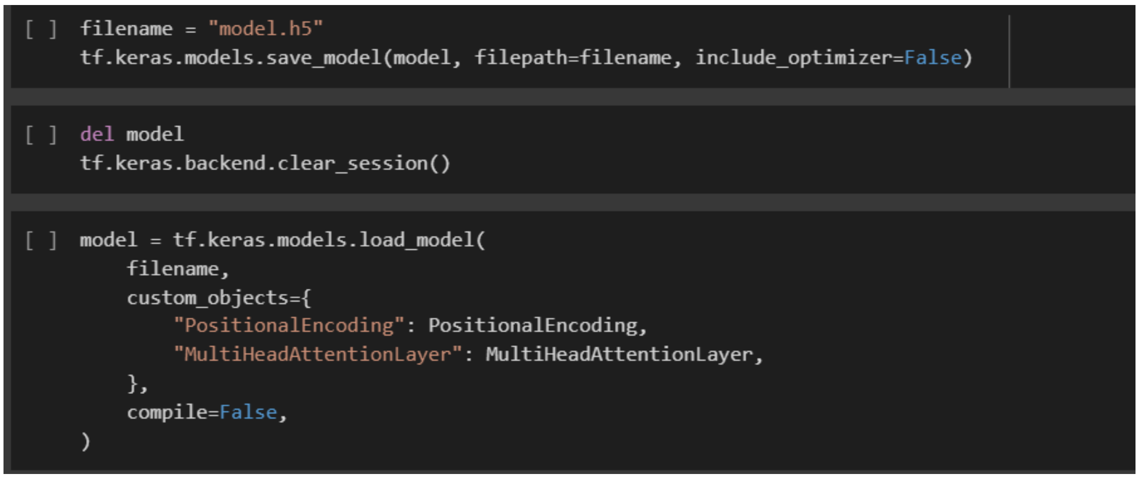
```
[ ] train = model.fit(dataset, epochs=EPOCHS)
```

Figure B33: Training model using fit function



```
[ ] # Plotting model Accuracy and Loss (Visualisation of Accuracy and Loss of the Model)
    # Accuracy Plot
    plt.figure(figsize=(14, 5))
    plt.subplot(1, 2, 1)
    plt.plot(train.history['accuracy'],label='Training Set Accuracy')
    plt.legend(loc='lower right')
    plt.title('Accuracy')
    plt.xlabel("epochs")
    plt.ylabel("accuracy")
    # Loss Plot
    plt.subplot(1, 2, 2)
    plt.plot(train.history['loss'],label='Training Set Loss')
    plt.legend(loc='upper right')
    plt.title('Loss')
    plt.xlabel("epochs")
    plt.ylabel("loss")
    plt.show()
```

Figure B34: Plotting graph for accuracy and loss of the model

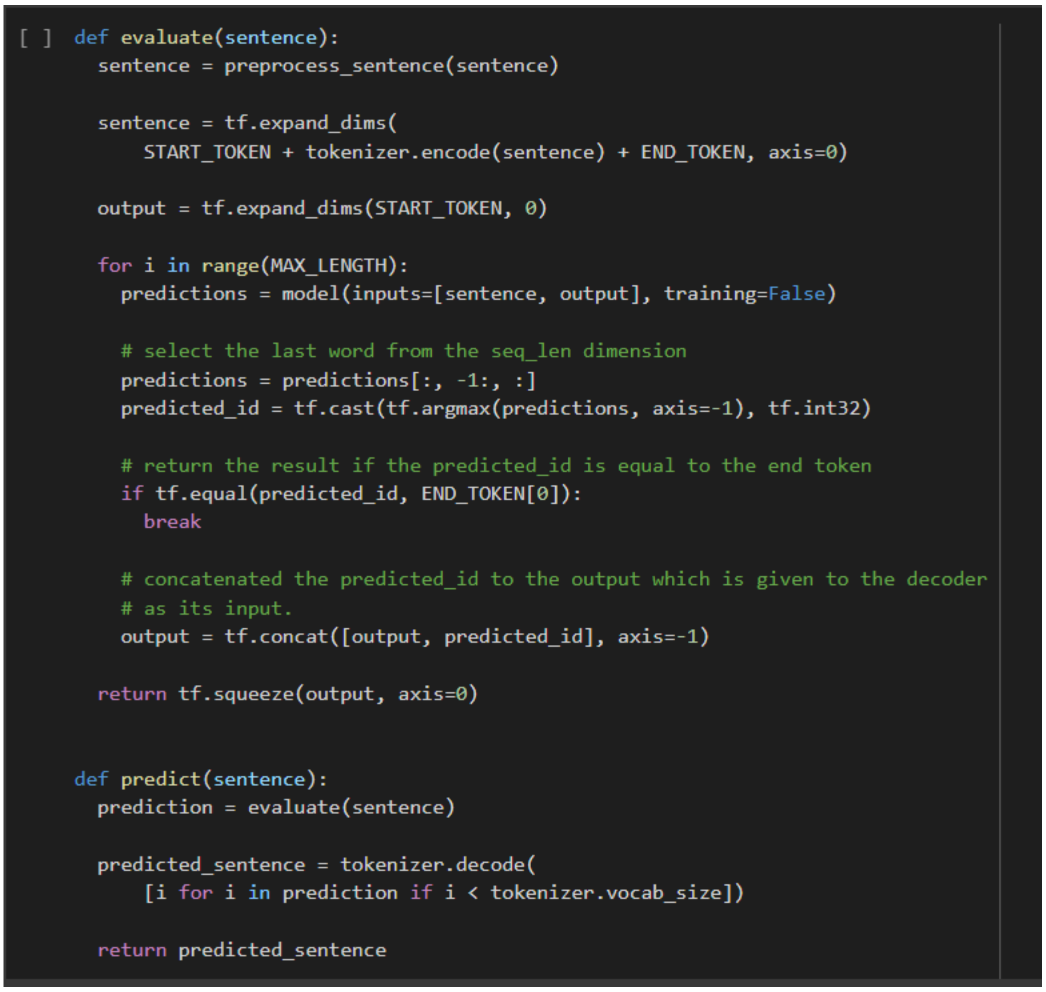


```
[ ] filename = "model.h5"
    tf.keras.models.save_model(model, filepath=filename, include_optimizer=False)
```

```
[ ] del model
    tf.keras.backend.clear_session()
```

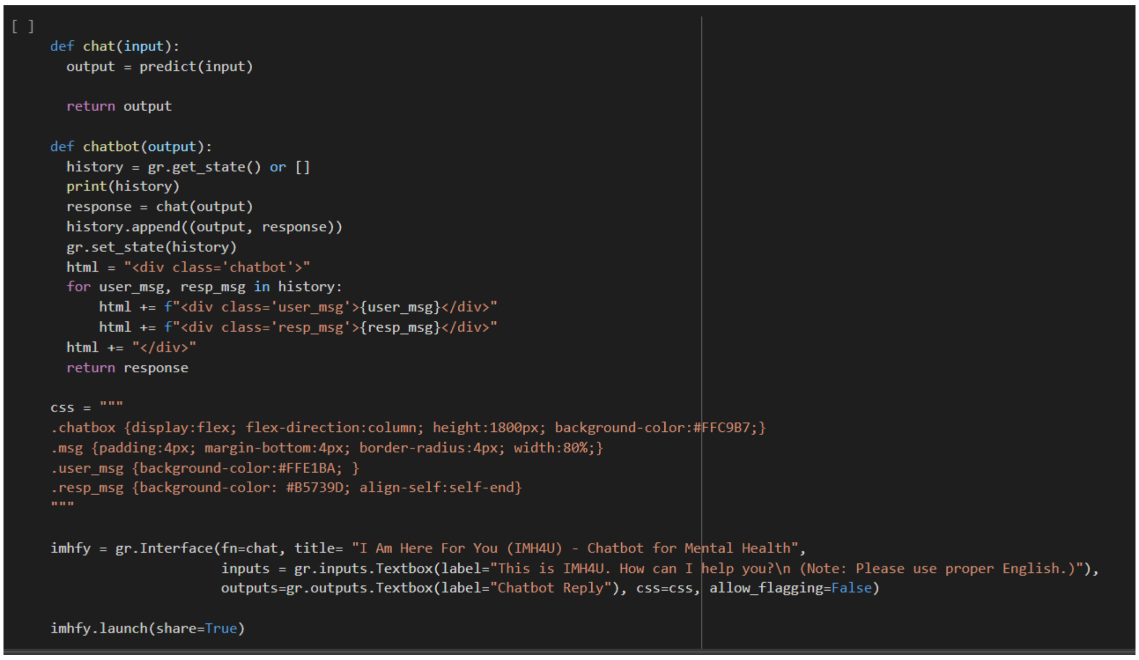
```
[ ] model = tf.keras.models.load_model(
        filename,
        custom_objects={
            "PositionalEncoding": PositionalEncoding,
            "MultiHeadAttentionLayer": MultiHeadAttentionLayer,
        },
        compile=False,
    )
```

Figure B35: Saving and loading model



```
[ ] def evaluate(sentence):
      sentence = preprocess_sentence(sentence)

      sentence = tf.expand_dims(
          START_TOKEN + tokenizer.encode(sentence) + END_TOKEN, axis=0)

      output = tf.expand_dims(START_TOKEN, 0)

      for i in range(MAX_LENGTH):
        predictions = model(inputs=[sentence, output], training=False)

        # select the last word from the seq_len dimension
        predictions = predictions[:, -1:, :]
        predicted_id = tf.cast(tf.argmax(predictions, axis=-1), tf.int32)

        # return the result if the predicted_id is equal to the end token
        if tf.equal(predicted_id, END_TOKEN[0]):
          break

        # concatenated the predicted_id to the output which is given to the decoder
        # as its input.
        output = tf.concat([output, predicted_id], axis=-1)

      return tf.squeeze(output, axis=0)


    def predict(sentence):
      prediction = evaluate(sentence)

      predicted_sentence = tokenizer.decode(
          [i for i in prediction if i < tokenizer.vocab_size])

      return predicted_sentence
```

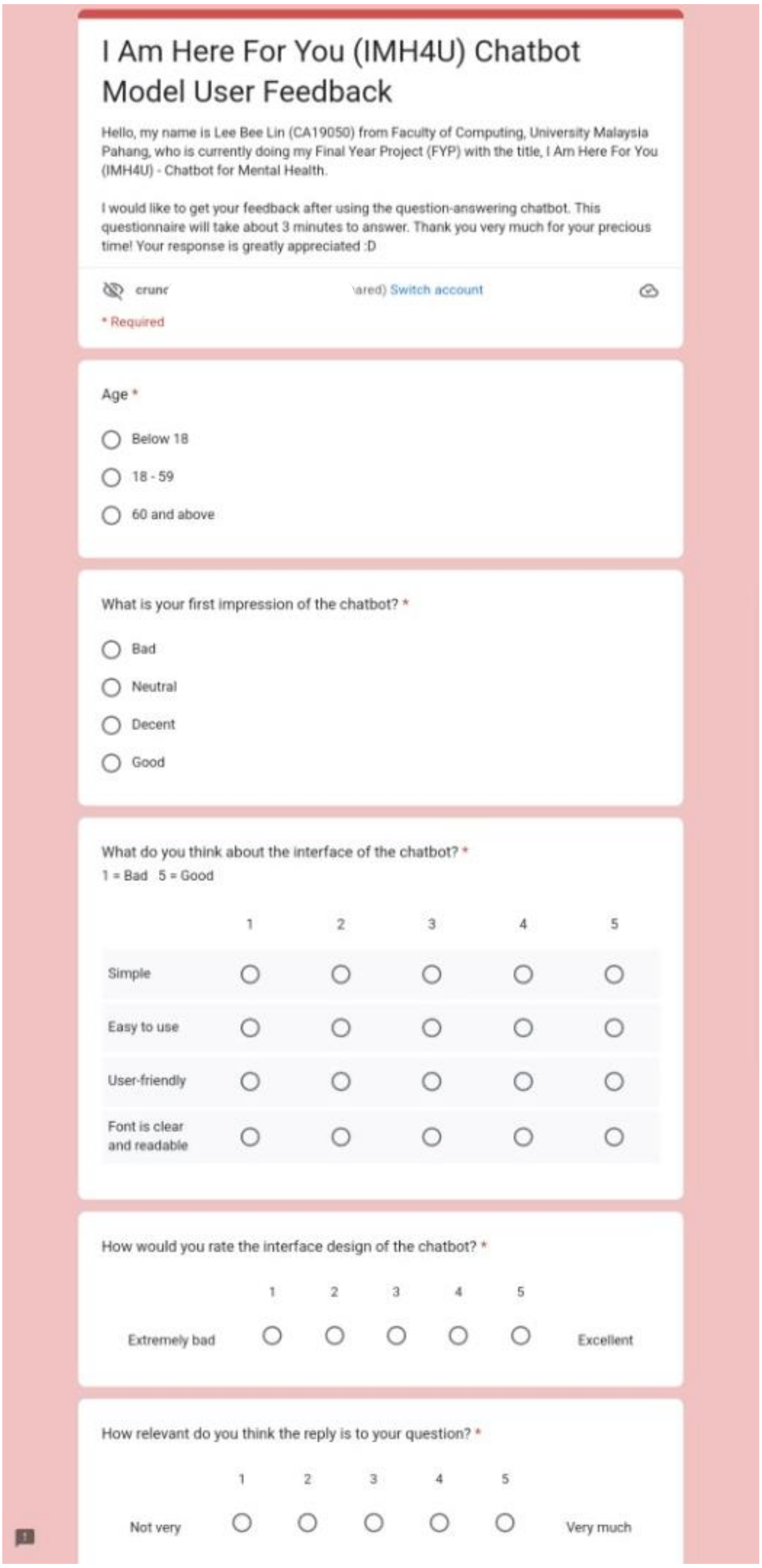
Figure B36: Evaluation and prediction of output by processing user input



```
[ ]
    def chat(input):
      output = predict(input)

      return output

    def chatbot(output):
      history = gr.get_state() or []
      print(history)
      response = chat(output)
      history.append((output, response))
      gr.set_state(history)
      html = "<div class='chatbot'>"
      for user_msg, resp_msg in history:
          html += f"<div class='user_msg'>{user_msg}</div>"
          html += f"<div class='resp_msg'>{resp_msg}</div>"
      html += "</div>"
      return response

    css = """
    .chatbox {display:flex; flex-direction:column; height:1800px; background-color:#FFC9B7;}
    .msg {padding:4px; margin-bottom:4px; border-radius:4px; width:80%;}
    .user_msg {background-color:#FFE1BA; }
    .resp_msg {background-color: #B5739D; align-self:self-end}
    """

    imhfy = gr.Interface(fn=chat, title= "I Am Here For You (IMH4U) - Chatbot for Mental Health",
                         inputs = gr.inputs.Textbox(label="This is IMH4U. How can I help you?\n (Note: Please use proper English.)"),
                         outputs=gr.outputs.Textbox(label="Chatbot Reply"), css=css, allow_flagging=False)

    imhfy.launch(share=True)
```

Figure B37: Configuration for chatbot interface using Gradio and launching

APPENDIX C
USER FEEDBACK FORM



I Am Here For You (IMH4U) Chatbot Model User Feedback

Hello, my name is Lee Bee Lin (CA19050) from Faculty of Computing, University Malaysia Pahang, who is currently doing my Final Year Project (FYP) with the title, I Am Here For You (IMH4U) - Chatbot for Mental Health.

I would like to get your feedback after using the question-answering chatbot. This questionnaire will take about 3 minutes to answer. Thank you very much for your precious time! Your response is greatly appreciated :D

crunc [redacted] ared) Switch account

* Required

Age *

- ◯ Below 18
- ◯ 18 - 59
- ◯ 60 and above

What is your first impression of the chatbot? *

- ◯ Bad
- ◯ Neutral
- ◯ Decent
- ◯ Good

What do you think about the interface of the chatbot? *
1 = Bad 5 = Good

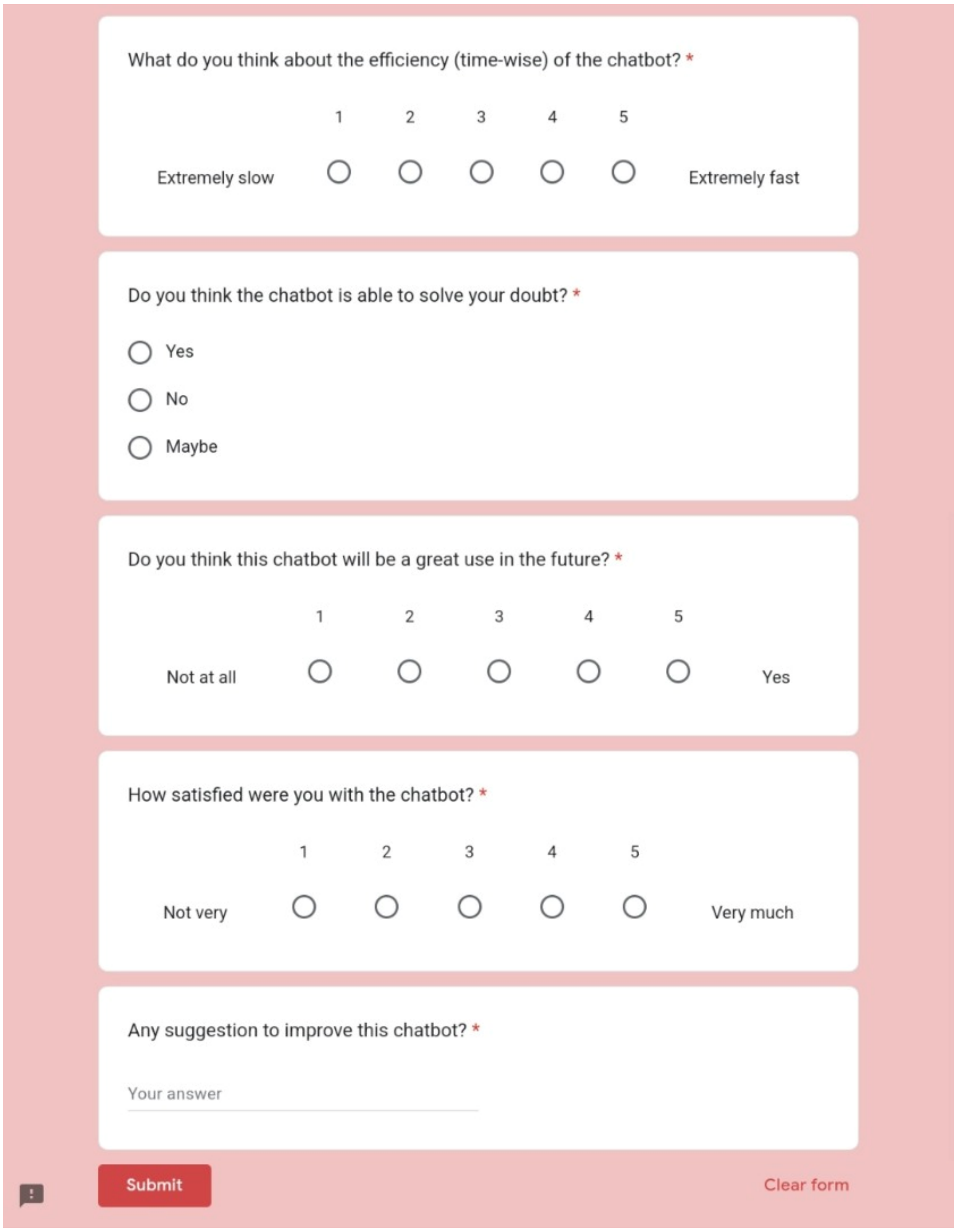
	1	2	3	4	5
Simple	◯	◯	◯	◯	◯
Easy to use	◯	◯	◯	◯	◯
User-friendly	◯	◯	◯	◯	◯
Font is clear and readable	◯	◯	◯	◯	◯

How would you rate the interface design of the chatbot? *

	1	2	3	4	5	
Extremely bad	◯	◯	◯	◯	◯	Excellent

How relevant do you think the reply is to your question? *

	1	2	3	4	5	
Not very	◯	◯	◯	◯	◯	Very much



What do you think about the efficiency (time-wise) of the chatbot? *

	1	2	3	4	5	
Extremely slow	○	○	○	○	○	Extremely fast

Do you think the chatbot is able to solve your doubt? *

○ Yes

○ No

○ Maybe

Do you think this chatbot will be a great use in the future? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Yes

How satisfied were you with the chatbot? *

	1	2	3	4	5	
Not very	○	○	○	○	○	Very much

Any suggestion to improve this chatbot? *

Your answer

Figure C1 & C2: Screenshots of User Feedback Form