



# Edge Hill University

The Department of Computer Science

## CIS3140 Research and Development Project

### Project Report

Development of a Machine Learning Based Chatbot  
to Provide Emotional and Informational Welfare  
Support for Edge Hill University Students

Student - Lewis Cunliffe (23926643)

Supervisor – Huaizhong (Sam) Zhang

Tuesday, 04 May 2021

*This Report is submitted in partial fulfilment of the requirements for the BSc Honours Computer Science Degree at Edge Hill University*

## Contents

Abstract .....	6
Chapter 1 - Introduction .....	7
Overview and Motivations.....	7
Aim and Objectives .....	9
Scope.....	10
Main Contributions .....	11
Report Structure .....	12
Chapter 2 - Background .....	13
Chatbots.....	13
Sentiment Analysis .....	14
Deep Learning and Relevant Techniques in Chatbots.....	15
Evaluation Metrics .....	19
Human Computer Interaction (HCI).....	20
Data Preparation .....	21
Chapter 3 - Literature Reviews .....	23
Overall Chatbot Focuses .....	23
Classification Method Choices for Conversation Modelling and Sentiment Analysis .....	24
Extension - Programming Choices.....	26
Medium Choice for Chatbot.....	27
Dataset Choice for Chatbot.....	29
Feature Review of Existing Chatbots.....	30
Literature Review Conclusion .....	32
Chapter 4 - User Requirement Data Collection.....	33
Interview .....	33
Topic 1 - Employment for Chatbot Service.....	34
Topic 2 - Categorisation of Support.....	34
Topic 3 - Common Dealings.....	34
Topic 4 - Students.....	34
Topic 5 - Conversation timings.....	34
Topic 6 - Medium Choice.....	35
Topic 7 - Interviewee Role.....	35

Topic 8 - Features.....	36
Survey .....	37
Survey Evaluation.....	39
User Data Discussion.....	54
Chapter 5 - Chatbot System Designs.....	55
Use Case Diagram .....	55
Requirements Design.....	56
Back End Designs .....	57
Class Diagram.....	58
Data Flow Diagram.....	59
Front End Design.....	60
Chapter 6 - Implementation and Testing .....	61
Testing Preface.....	61
Stage 1 - Back End Base Conversation Simulation .....	62
Development Choices .....	62
Program Structure.....	63
Development Discussion.....	67
Testing.....	69
Conclusion.....	72
Stage 2 - Emotional Analysis Integration .....	73
Development Choices .....	73
Program Structure.....	73
Development Discussion .....	73
Testing.....	74
Conclusion.....	76
Stage 3 - Support Base Integration .....	77
Parameter Choices .....	77
Program Structure.....	77
Development Discussion .....	77
Testing.....	78
Conclusion.....	81
Stage 4 - Medium Integration .....	82

Development Choices .....	82
Program Structure.....	82
Development Discussion.....	83
Testing.....	84
Conclusion.....	86
Final Product.....	87
Artifact Availability.....	87
Chapter 7 - Evaluations.....	88
Process Evaluation .....	88
Pre-Project Evaluation.....	88
Background Research Evaluation.....	89
Literature Review Evaluation .....	89
User Requirements Evaluation.....	89
Designs Evaluation .....	90
Development Evaluation.....	90
Performance Evaluation.....	93
Goals Evaluation .....	94
Choice Evaluation.....	95
Chapter 8 - Conclusion.....	99
Summary.....	99
Limitations .....	100
Future Work.....	101
References .....	102
Appendices .....	110
Item 1 (Interview Participant Information Sheet).....	110
Introduction.....	112
Study Title.....	112
Principal Researcher .....	112
Study Information.....	112
Interview Information.....	114
Item 2 (Participant Consent Form (Unsigned)) .....	115
Item 3 (Participant Consent Form (Signed for Interview)) .....	116

Item 4 (Interview Participant Email Discussion).....	117
Item 5 (Interview Transcript) .....	119
Item 6 (Initial Survey Design) .....	121
Item 7 (Initial Survey).....	126
Item 8 (Pilot A Conversation).....	130
Item 9 (Pilot B Conversation) .....	131
Item 10 (Pilot Changes).....	132
Item 11 (Distribution Email).....	133
Item 12 (User Testing Participant Information Sheet) .....	134
Introduction .....	136
Study Title .....	136
Principal Researcher.....	136
Study Information .....	136
Testing Information.....	138
Item 13 (Participant Consent Form (User Testing Signed)).....	139
Item 14 (User Testing Participant Conversation) .....	140
Item 15 (Stage 1 User Testing) .....	142
Item 16 (Stage 2 User Testing) .....	144
Item 17 (Stage 3 User Testing) .....	146
Item 18 (Stage 4 User Testing) .....	149
Item 19 (First Attempt Code - Expanded Insufficient Dataset) .....	152
Item 20 (Stage 1 Code).....	158
Item 21 (Stage 2 Code Additions) .....	164
Chatbot Training Additions .....	164
Chatbot Running Additions .....	164
Item 22 (Stage 3 Code Additions) .....	165
Item 23 (Original Psychology Dataset).....	165
Item 24 (Modified Psychology Dataset and Added EHU dataset).....	168
Item 25 (Stage 4 Additions and Modifications).....	172
Item 26 (Proof of medium implementation).....	179

## Abstract

University support systems are an important tool to assure students quality of study is optimal, yet the field is often ignored by both universities and students themselves for a number of reasons. Chatbots are emerging technology that can provide numerous types of service with minimal need for supervision unlike human controlled discussion services. An extensively used service for chatbots is support with current limitation of either informational or emotional focus and lack performance as a unified support tool for these two categorisations. Edgehill University currently uses a human controlled support system that whilst effective, faces many of the issues that are inevitable of a human controlled system. Here we report on development of an automated generative chatbot system that intends to provide informational and emotional support simultaneously for use in the Edgehill University domain as an alternative unmanned system to their current solution. We use existing literature to find the most effective approaches for accomplishing this, then examine the current needs of students and employees of the current system via survey and interview. We then design and develop the proposed system through use of suitable DeepLearning and sentiment analysis techniques. The system produced found that with current resources, particularly data in this domain, the chatbot could not be developed to be sufficient for functional use. Despite this, the project produces a full investigation, and a strong demonstration of how informational and emotional support can be combined in an accessible, automated format where this was previously not investigated.

## Chapter 1 - Introduction

This document is a report to the research and development process of the proposed support chatbot. Initially, background to relevant topics to the chatbot will be discussed to show the research done prior to development. Then, literature reviews and surveys are conducted to discuss issues with current chatbot systems and make informed decisions regarding development based on these. After, the problem will be analysed, taking perspective of potential users and employees of the current system through survey and interview, with each process recorded in this report. Once results are gathered from this analysis, system requirements will be established. Using the system requirements, a set of designs ranging from the system GUI to the inner data transmission of the processing system will be drawn to establish a clear foundation for development. Finally, development will be documented with the stages of the development paradigm to produce a full report of development. The report will close with conclusion of the outcome of the project as well as potential further development discussion.

### Overview and Motivations

Chatbots are software that automate human interaction and cater to a specific domain. A common domain choice for chatbots is customer service, replacing human customer service. According to Adam et al (2019) chatbots have become “increasingly popular” in this domain due to their “cost- and time-saving opportunities”. Constant availability with no wage cost has made them stand out as a replacement for human use. Despite this, Adam does claim that they “still frequently fail to meet customer expectations”.

University presents new challenges constantly to students, not only in academic studies but elsewhere. The exposure to a new environment and systems can be overwhelming and a support system to cater to student needs is necessary.

According to a report by Ciobanu (2013) student support is one of the most important factors of education quality and helps to decrease dropout rate. The report claims that the most important supports are those that meet academic, personal, and emotional needs. It also finds that accessibility and practicality are important factors for support use.

For the project, support has been categorised as emotional or informational. Emotional support refers to helping subjects with issues such as stress or loneliness. This would require some emotional analysis from the chatbot. Informational support refers to helping subjects with issues of knowledge such as directions to locations on campus. Assuring the chatbot functions in a domain specificity would be essential to this. These categorisations are an efficient abstraction of the different issues and whilst issues can be further broken down, doing so provides no benefits for this project.

Schlicht (2016) explains that Chatbots vary between basing their conversational flow on rules or machine learning. The machine learning based approach is chosen due to being less restricted despite potential loss in control. Machine learning itself can also vary between many methods. One subset is Deep Learning, a method that uses neural networks to develop its model with less human intervention than machine learning. According to Cai et al (2018) “Traditional text classifiers based on machine learning methods have defects such as data sparsity, dimension explosion and poor generalization ability, while classifiers based on deep learning network greatly improve these defects, avoid cumbersome feature extraction process, and have strong learning ability and higher prediction accuracy”. The project aims to use text processing in both its function as a chatbot but also in an emotional support component using sentiment analysis. An effective method for text classification is therefore integral. DeepLearning can be further split into individual methods and during the literature review, Seq2Seq was found to be the most ideal with an integration of an additional classification feature for sentiment analysis.

By applying DeepLearning approach to creating a chatbot, as well as additional features such as sentiment analysis an automated support system for students can be developed that handles both emotional and informational issues without the flaws of human controlled support that have motivated the development of this project.

The motivation for completing this project comes from personal experience as a student at Edgehill University. Whilst the existing system provides ample support, the use of human handled support systems raised a variety of concerns such as privacy, response time and inability to be fully anonymous. Due to this, even when support was required and likely easily given, it was avoided. By developing an automated alternative, this problem faced would be resolved. As well as this, current automated support systems lack contextual understanding of what is being discussed and are mostly limited to informational support that is incredibly rigid. This causes them to fail at simulation of human speech convincingly. A proposed improvement would be to include the emotional context human speech has to improve the problem of current automated system's lack of understanding and provide a service as capable as the current human operation for Edgehill University.

The purpose of this project is to develop an automated chat support system for Edgehill University to provide a suitable alternative to the current human operated system in aim of resolving the issues human operation brings. In attempt to provide suitable alternative, the system will combine both informational and emotional support which will establish new understanding of conversation simulation due to current lack of research in combining the two.

## Aim and Objectives

The aim of this project is to develop a chatbot that provides sufficient emotional and informational welfare support to students at Edge Hill University.

1. Conduct background research in development approaches of chatbots.
  - a. Conduct literature survey of deep learning methods to find most effective approach for development of chat simulation and sentiment analysis.
  - b. Conduct literature survey into process of integrating chatbot into accessible mediums (such as websites, apps, or its own software) to find the most fitting medium.
  - c. Conduct literature survey into creation of datasets for chatbots to find effective approach.
  - d. Conduct literature review of existing chatbots into their features and find suitable features for project.
2. Identify and evaluate the user requirements of the chatbot.
  - a. Interview current student chat support staff member to find information regarding current process, such as common topics and issues with current system.
  - b. Survey current students to find their views regarding chat support and needs for a system.
3. Design the chatbot.
  - a. Establish use case diagram and user requirement list, discuss how each will be implemented.
  - b. Design back end of chatbot
    - i. General topic identification for datasets listing identified topics.
    - ii. Class diagram detailing the code design.
    - iii. Data flowchart showing DeepLearning process.
  - c. Design front end of chatbot with user interface design
4. Develop the Chatbot.
  - a. Develop back end of general use chatbot draft using designs.
    - i. Develop conversation simulation functionality and test (function, evaluation, and user) revisiting development stages if necessary.
    - ii. Develop sentiment analysis and integrate into conversation simulation, test sentiment analysis (function, evaluation, and user) and revisit development if necessary.
    - iii. Develop knowledge base integration framework.
      1. Develop back end of domain specific chatbot using designs.
      2. Creation of dataset for knowledge base
      3. Integrate dataset into knowledge base, revisiting back-end development if necessary and testing (function, evaluation, and user)
  - b. Develop front end of chatbot using designs.
  - c. Develop chatbot integration, test (function and user) and revisit development if necessary.

## Scope

This project aims to provide an alternative automated chatbot system for Edgehill university whilst establishing the development process of combining both emotional and informational chatbot support systems in a specific domain, a process that is widely under researched. The aim is to produce a chatbot capable of this combination and investigate the development process to do so.

The scope of this project is limited to Edgehill University and use within. This mainly affects the access to data as chatbot systems function optimally with larger datasets to support them. Given that Edgehill University has no available datasets specific to its support domain, there are certain considerations of limits to the performance of the chatbot, especially when dealing with domain specific issues.

As well, the hardware requirements of the system may be limited at this time. Processing the data and system will be limited to hardware provided by ourselves and the university. Hosting at this time would also be limited without a considerable userbase and funding. Other scope limitations are based on dealing with those surveyed and interviewed as this is limited to students at the university and resolving contact issues is limited by current Covid-19 circumstances. Sample size will be limited due to these factors.

Due to this, the limitations of the scope should be considered when interpreting the results. Certain performance issues or user testing may produce outcomes that are insufficient, yet the project may succeed as a lesser scaled solution to the issues discussed.

## Main Contributions

This section will discuss the methods intended to be used. Some methods used are conditional based on certain stages of the project and will not be defined here.

As a development paradigm, the iterative nature of Agile development makes it suitable for the project. During development, the chatbot is developed in smaller stages and evaluated at each, potentially revisiting development if the testing produces insufficient results. Agile “is the ability to create and respond to change” (Agile Alliance, 2019) and so is very suitable for this. Despite this, development for the project is largely independent so the suitability for teamwork that Agile provides is ignored.

The method of DeepLearning for both the chat simulation and sentiment analysis components of the project will be decided via a literature survey to understand which is most effective for the project. A similar approach will compare methods of integration into accessible mediums, but those options are likely to vary between software integration, web integration or messenger app integration. As well, a literature survey will be conducted into dataset choices for chatbots and find the best format for development. From understanding of a method for DeepLearning and sentiment analysis, a package for processing will be selected during the review stage when one is found to be suitable for the research findings.

User requirements will find the needs of potential users both as features and what the chatbot will cover. This will potentially be used to populate the dataset to build the chatbot from, as well as find features that may be required such as accessibility options. To find these requirements, an interview will be conducted with a current student working as a manual chat support through the university website via the “chat with our student” section. The interview will help find information about a similar service to this one proposed such as what topics are regularly covered. A survey will also be distributed to a small sample of students to discuss their opinions regarding the service with questions such as if they had used it, what they believe could be improved if they had or why they did not use it if not. From this, the general opinions of this sample will be used to direct the designs and development of the chatbot similarly to how the interview did.

Using these requirements, designs of the project will be established to demonstrate how development will proceed. A user requirement list and use case diagram will be made from the results of both the interview and survey as well as any others identified prior to this. A design listing the topics needed for datasets will be also made as well as a class diagram for the program itself and a data flowchart to plan the data processing the chatbot classification will do. Finally a design for the front end of the chatbot will be made as a GUI sketch of the implementation method found suitable.

In development, the agile method will be followed with four major iterative stages. The first three major stages will cover the development of the back-end implementation for the chatbot. The first stage will develop a nondomain specific level that will create a general-purpose conversation simulator and serve as the foundation for the following features. The second stage will build upon the development of the first, incorporating emotional analysis into the classification of the first to allow for conversational simulation based on perceived emotion. Then, the domain specific level will be integrated with a knowledge base with data specific to the aim of providing support to Edge Hill University students. Once the backend of the chatbot is created for use, the fourth and final stage will cover the front-end development to allow for functionality to be accessible to users. Testing of varying types will be conducted at all stages with their respective prototypes. All stages will use function testing to evaluate if each function provided works without error. The backend stages will use evaluation testing to gather metrics regarding the classification and find if they are sufficient. All stages will also incorporate user testing where a third party will use the software and provide feedback.

## Report Structure

As discussed earlier, the report will contain the process used to develop the chatbot artefact in attempt to accomplish the set aim. The report will be split into seven chapters.

The first chapter is this one, the introduction. It has covered the motivation for creating this project as well as given general background of the topic. It has described the purpose for the project as well and defined the aims, objectives, and scope. It has also defined the methods that will be used and the structure of this report.

The second chapter, the background, will cover the background research done to establish an understanding of the project and any relevant topics using existing literature. Each topic researched at this point will be discussed.

The third chapter, the literature reviews, will contain the literature reviews completed to compare existing chatbot solutions in attempt to find the ideal methods to use in this project such as DeepLearning structures and integration mediums.

The fourth chapter, the requirements, will cover the gathering and analysis of the requirements of the project. Initially an interview and survey will be conducted to establish an understanding of what the project requires from different perspectives.

The fifth chapter, the designs, will contain the designs discussed in the methods section, based on previous information provided from prior chapters. It will firstly define the requirements for the project based on the user requirements, literature reviews and previous research. It will then create the backend designs, beginning with the knowledge base and its topics, followed by class diagram and then the data flow for the learning model itself.

The sixth chapter, the development, will contain a documentation of the development process undertaken to implement the designs. It will cover the entire agile cycle, demonstrating development, testing, evaluation, and reiteration until the product is satisfactory. Development will be based upon the designs created and testing will be conducted to seem if the implementation is successful at each stage. Each stage will also analyse if the defined requirements for development have been met.

The seventh chapter, evaluations, will reflect on certain aspects of the project. It will firstly discuss each process undertook to complete the project, discussing issues faced and success of each step. It will then reflect on the performance of researching and developing the process, criticising the approach. A goals evaluation will then be conducted to examine the requirements, aims and objectives of the project and if they had been met. Finally the choices made will be evaluated and alternatives discussed.

The eight and final chapter will conclude the project. It will firstly summarise the project and its findings and then discuss the limitations encountered and their effects. Finally, the future work for the project will be discussed.

## Chapter 2 - Background

### Chatbots

Chatbots are “on-line human-computer dialog system with natural language” (Jia, 2003) designed to simulate conversation to provide numerous support formats in a variety of domains such as support desk tools, automated telephone answering systems, educational aid, business, e-commerce, entertainment and more (Shawar and Atwell, 2007).

Chatbots have three major functions or agents. The first, the dialogical agent, provides comprehension or the ability to understand the users need. The second, the rational agent, which provides competence which is the ability to answer user questions via the support of an external base of knowledge. The third, the embodied agent, provides presence which is the ability to appear more human or to have a persona. (Sansonet et al, 2006).

Chatbots vary between two sources for the rational agent, those being either a set of predefined rules or a machine learning classification. Both methods process user input and compare it to its own base of knowledge to find a relevant output. In the case of rules based chatbots, the knowledge base is defined before use and is rigidly applied to whatever rules are defined. They are ideal for serving specific purposes and domains but fail to function outside of their rules. Machine learning based chatbots classify the inputs and responses via a machine learning algorithm that learns from conversations and can function outside of predefined statements. Whilst they surpass rules based chatbots in their freedom, they can suffer due to this and require large datasets and efficient methods to function well. (Schlicht, 2016)

Machine learning based chatbots are further split into retrieval-based models or generative models (Dua, 2019). Retrieval based models function by using classification methods to find the most relevant response from a predefined dataset. Unlike rules based, the matching process between input and response is defined via machine learning and not by the programmer (Fainchtein, 2020). Generative models generate new responses by using techniques used by machine translation between languages, where the input is considered one language and output another. Generative models can suffer from grammatical errors unlike their counterpart as well as requiring larger training data. They are considered hard to train. Despite this, they benefit from being able to handle unseen cases and handle contextual information, such as names, which massively benefits attempts to satisfy the embodied agent. (Britz, 2016).

A further split to chatbots is their task orientation. A task oriented chatbot (also considered closed domain) focuses on a single goal in a single domain and are limited to this. On the other hand, non-task oriented chatbots focus purely on conversing with the user (Chen et al, 2017).

## Sentiment Analysis

Sentiment analysis, or opinion mining is defined as “the field of study that analyses people’s opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes” (Liu, 2012).

Sentiment analysis obtains a texts sentiment metric, usually a polarity score where negative values represent a negative sentiment, zero a neutral and positive a positive sentiment (Sarkar, 2018). The analysis can be extended into emotional analysis, which returns various scores under different emotional labels rather than simply the texts attitude. These emotional labels usually fall under the categories anger, sadness, happiness, fear, disgust, and surprise. The benefit of emotional analysis is that it gains further knowledge of the emotion behind text and not just opinion but is more complex to accomplish (Hakak et al, 2017).

Sentiment analysis occurs at different levels. Document level classification finds the sentiment of an entire piece of text, sentence level classification of a single sentence and aspect (or feature level) of a single feature (likely to be a word). As the level becomes more specific, certain considerations need to be made. Sentence level needs to define if the sentence is objective or subjective as certain sentences may simply be stating fact and analysing sentiment would produce data of no value. Feature level sentiment analysis likely needs to handle groupings as a single word may have very different meaning than its collocation. (Katrekar, 2019).

Sentiment analysis approaches vary between being machine learning based and lexicon based. Lexical methods take the text at the level being processed and compares the text to a predefined lexicon that contains its own text and a sentiment score attached to said text. Lexicons are essentially dictionaries giving sentiment scores to certain words to be applied. Lexicon based approaches are likely to be accurate but require that the sentiment score be predefined, and the text being used is within the lexicon. Machine learning methods use a variety of classification methods to categorise sentiment not on any predefined rules but example datasets where the sentiment is already classified. This allows the classifier to learn what causes the sentiment score to be certain values and assign new values based on this, potentially to unrecognised text depending on the methods. Both approaches require large datasets to be more effective, yet machine learning allows for use outside of the dataset. (Annett and Kondrak, 2008)

## Deep Learning and Relevant Techniques in Chatbots

DeepLearning is a subset of machine learning. It uses artificial neural nets, a classification model inspired by the human brain in a multi-layered system. In comparison to conventional machine learning, it is more scalable and needs no manual feature extraction. Machine learning requires some manual feature identification such as converting images into pixel values whereas DeepLearning does not and therefore allows for a suitable way to handle complex feature identities such as images or text (LeCun, 2015).

Artificial neural nets loosely follow the biological model they simulate. They are made up of connected units or neurons. Each neuron is a graphical representation of a value and the connections between are weighted by a metric of the importance of their link. As the network learns, the weighting changes to find an ideal value. Neural networks are layered with an initial input layer where the input values are inserted and a final output layer where the classification values are returned. Between those layers are the “hidden” layers, where mathematical operations are performed to find the outcome. The process of obtaining the output via processing the neural net is forward propagation. By taking the dot product of weights of each connection between a layer and the corresponding values of the previous neurons a value is found to represent the transmission between layers. This transmission must then be activated via a chosen activation technique to obtain the next layers corresponding neuron values. Activation techniques vary. This process repeats until the output layer is reached. The weights of connects can produce different classifications, some closer to the correct prediction and others further away. A loss function compares predicted values to the actual values and returns a metric to show the proximity. Weights are altered depending on this, and this is the learning process neural nets use. Loss functions also vary. The weight alteration is done via gradient descent, where the derivative of the loss function is found to direct the change in weight to approach a lower (and therefore more accurate) loss function (Oppermann, 2020). Deep learning uses a variety of different architectures of neural networks for a variety of domains and refers to the number of hidden layers used.

Recurrent Neural Networks are an architecture of Deep Learning that specialise in dealing with sequential data, such as videos or more commonly text. By considering the structure of data itself, certain data can be inferred rather than simply as individual tokens. According to Manideep (2020), RNN's classify using an internal state that holds the memory of the sequence as a whole that is used for its process and distinguishes itself from alternative solutions and updates as the internal state continues. The process follows a typical structure as shown in figure 1.

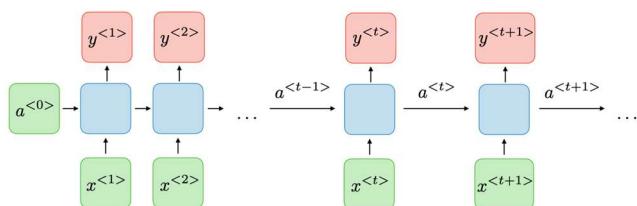


Figure 1 - Architecture of a traditional RNN (Amidi and Amidi, 2019a)

In figure 1, X is the token input. The system begins with an initial token being input (this being  $x^{<1>}$ ). This is then processed, (this being the blue square) and the context is manipulated by this input (this being the A which is changed throughout the process and contributes to the process outcomes). The output of this process is then given (shown as y), which in many cases also contributes to the next process ( $x^{<2>}$ ) depending on the structure and purpose. This process produces a context made up of the processes completed as well as outputs, but some cases may be fully focused on the context itself or use a different input (see seq2seq decoder for an RNN that uses a previously established context as its input). (Amidi and Amidi, 2019b). Manideep goes on to explain that “vanilla” RNNs are considered poor at storing longer sequences and RNNs have multiple more complex subtypes that intend to resolve this.

The Long Short Term Memory model introduced by Hochreiter & Schmidhuber (1997) intended to resolve the issue of storing sequences and long-term dependency issues. Unlike vanilla RNN's single internal layer, LSTMs use many that interact. One layer, the sigmoid layer, outputs a value between 0 and 1 which determines how much of the stored memory of a sequence should continue. The rest of the layers in LSTMs use this, to determine what data is necessary for classification. These can be considered the hidden layers of this model. Whereas RNNs have no control of data to hold, LSTMs decide their significance allowing for longer data dependency and therefore greater performance (Olah, 2015).

The Seq2Seq model uses two RNN structures to classify data, these are the encoder and the decoder. The Seq2Seq model was introduced by Cho et al (2014) intended for linguistic use and finding extensive suitability for translation, transcription, and dialogue systems. The encoder receives an input sentence and processes one token (each word) at a time. The process attempts to convert each token from the sequence into an abstraction of the sentence as a whole, as a fixed size vector containing the sentence context and reducing any unnecessary details. Each token process influences the next and is conjoined from all the sentence data to form what can be considered the context, the summary data of the input sentence. The input of a sentence is tokenized into separate chunks. A process occurs for each chunk, embedding it into the encoder RNN which can then encode each chunk into the context. The context is then sent to the decoder RNN. The decoder then reverses this process, generating a series of tokens based on the input context which are reconverted to normal text in the desired format through the SoftMax layer. It should be noted that the RNN's used can be configured as LSTM's. See figure 2 for a diagram of this.

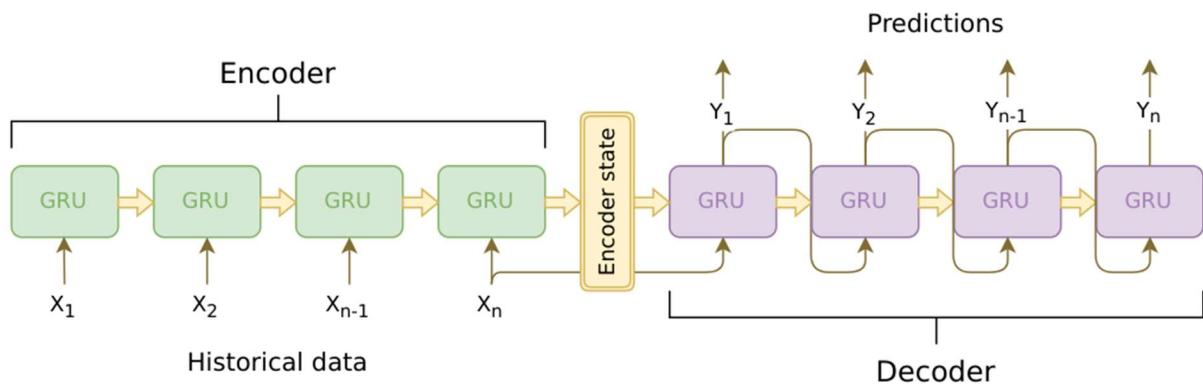


Figure 2 - Seq2Seq Diagram (Kim, 2021)

This model is dependent on use of fixed size vectors, limiting the shape of inputs into the program. The dependency of fixed size vectors does create the major issue that the model cannot effectively handle variable length sentences, despite their common use. An attempt to resolve this is made with padding, by converting potential variable length sentences to fixed length by introducing specific tokens to fill their sizing and make them suitable. The tokens introduced have a variety of functions as well as filling, such as marking the ends or starts of sentences. This solves the issue, but further introduces sentences with large amounts of data due to padding. Bucketing is another solution to this issue, by storing multiple sentence lengths into “buckets” of various sizes. For example, an input of three words would be put into the small bucket as it is suitable to handle the sentence yet a sentence of fifty would be put into a larger bucket as it is capable of handling it. This avoids the smaller sentence having the inefficiency of being included in larger sentences whilst allowing for some variability of sentence sizes.

Another issue is the data format of the inputs for the model. Text needs to be converted to be readable by computers. There are multiple methods, and each method of representing this data is important to ensuring efficiency. One such method is word embedding. This distinguishes words graphically with their semantic meaning controlling their position on the graph. An example would put “garden” and “hose” close to one another based on their likelihood of being used together due to similar topic in meaning. This is typically an initial process of Seq2Seq and assures that data is accessible for classification. Another is One Hot encoding where words are converted to representations of a dictionary with occurrences of specific words being marked as one. This is far less efficient than the embedding process.

Seq2Seq suffers from the unsuitability for larger sentences. Even with use of LSTM's, the model can struggle to hold long lasting data. The attention mechanism is a feature that can be introduced to resolve this. It works by providing weighting to the context data being processed so that as each token is processed, the data held as the context is judged. When the decoder process begins, the decoder can now examine the weights of the input context to understand the location of certain specific needs and align elements of the output to elements of the input and use a weighted sum of input contexts to produce the output instead of usually giving all input contexts the same amount of focus, hence the name attention. The mechanism gives attention to specific important components of the input and ignores lesser components. This is useful for implementation for long text usage such as document translation (Ramamoorthy, 2016).

The implementation method for attention varies between two major techniques named after their creator. Bahdanau's attention (Bahdanau et al, 2014), also known as additive attention functions by combining the assignment scores and finding the weights from this. The weights are then applied to the encoders output to produce the context as an input into the decoder. Luong's attention (Luong et al, 2015), also known as multiplicative attention, functions in the decoder instead and uses alignment scores directly within the decoder layer. Luong's alignment scores are also calculated differently, usually by multiplying the hidden state of the encoder with the hidden state of the decoder. (Loye, 2019)

Certain aspects of deep learning structures can be changed between use such as hidden layers, epochs, batch size, learning rate and activation functions. Hidden layers are defined as the layers between input and output that direct data with activation functions that perform non-linear transformations on the data to produce the intended output (DeepAI, 2019b). In consideration of the previous discussion of DeepLearning, they can be considered the “neurons” within the layers of an LSTM computational structure. Each unit in the hidden layers perform calculation on the data to classify it. Balancing this value is important to assure that the model fits ideally (lower units may cause underfitting where the trends of the data are not captured and higher may cause overfitting where the trend of the data is caught too closely and takes consideration of unwanted data such as outliers (Al-Masri, 2019)) as well as balancing any hardware considerations with this. According to Eckhardt (2018), the most common framework for choosing the number of hidden layers is k-fold cross-validation using the equation  $Hl = \frac{Samples}{(SF * (In + O ))}$  where samples is the number of samples of training data, SF the scaling factor (usually between 2 and 10), In the amount of input nodes and On the amount of output nodes. A simpler format that produces an amount nodes simply finds two third of the amount of input and output nodes. Eckhardt does go on to say that these methods are only a guide, and that trial and error produces the greatest results.

Between the hidden layers, the neurons are connected, and their relationships determined by the mathematical function processing between. The function performed is the activation function, which “determine the output of neural network like yes or no” and “maps the resulting values” (Sharma, 2017). There are many potential activation functions that can be used. For the project, the SoftMax function will be used as this takes an input selection of categories and distributes a probability (summing to one) between the classes according to the training process (Mahmood, 2018). This is very appropriate for processing multi-category data, like the projects own where potential words for the chatbot can be considered classes.

An epoch is defined as a “cycle through the full training dataset” (DeepAi, 2019a) and by extension the number of epochs is the number of cycles through the training dataset. According to Brownlee (2018) the number of epochs is usually large and is intended to run until the original models’ error is resolved and the trend of data established.

Batch size is defined as the “number of examples from the training dataset used in the estimate of the error gradient” (Brownlee, 2019a). It controls the amount of data from the dataset stored and used for the training process at each time. Batch size therefore affects memory usage as well as generalisation. Batch size has three formats according to Brownlee (2019a), batch gradient descent (where the total number of examples is used), stochastic gradient descent (where batch size is set to one) and mini batch gradient descent (where a value between one and the max is used). Stochastic gradient is commonly used for data streams instead of sets of data. Similarly to the number of nodes, higher batch size can lead to too close a trend established and therefore overfitting. Lower batch size introduces noise. An advantage of mini batch gradient descent is that it allows for the model to “start learning before having to see all the data” (Shen, 2018) by separating the data into segments as well as reducing memory usage. In a study by Shen (2018), use of mini batch was found to be superior in almost all compared categories and the smaller of the compared sizes performed best.

Another aspect to consider is the learning rate. The learning rate is a “hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated” (Brownlee, 2019b). This value needs balancing like the others as too large allows the model to train too quickly to a suboptimal result yet too low and the process may become too slow. To handle this parameter, an adaptive learning rate optimizer like RMSProp can be used to manage the value as the process runs.

## Evaluation Metrics

For evaluating machine learning processes, two major metrics exist. These are loss and accuracy. Loss is the distance of a predicted value from the actual value (Kumar, 2018). As described earlier, loss is used to correct classification but also finds use in evaluation. The most common function for calculating loss is categorical cross entropy loss where the probability of a prediction belonging to a specific category is analysed and the distance (from the 1 denoting certainty) of belonging to the correct category is examined and used as loss (Peltarion, 2020). In the project, these categories will denote the correct word for the chatbot to select as its output. The Peltarion documentation (2020) goes on to state that the SoftMax activation function is the recommended choice for conjunction with this function. Loss is used to determine the direction that classification should take, acting as a correcting metric for the data. This makes it also suitable to evaluate the performance of the project as well. Due to this, it will be used in evaluation testing.

Accuracy is the measure of correct predictions over total prediction, returning a percentage of those that were deemed correct. A higher accuracy denotes a more successful model. Given its measure of a model's performance, this will also be used in evaluation testing.

Whilst not a metric itself, using validation data is a method of evaluating chatbots. Desmerais (2018) explains that validation data is the use exclusion of a portion of data from the training process to use as a comparison for each epoch with the accuracy allowing for an understanding of how closely a system can predict values without being trained for those specifically. Validation produces accuracy and loss that can be compared to the training data's accuracy and loss to find how well the model predicts for data that is not included. Although this is useful, in this project the aim is to produce open ended outputs and the data would likely not reflect the chatbots performance clearly. User testing the chatbot to find how users react to outputs that are not predefined would likely produce a better testing method.

## Human Computer Interaction (HCI)

HCI is the study of interaction between humans and computers and to what extent the interaction between the two is considered successful. HCI consists of three parts: the human (or the user), the computer (any technological interface, a website can be considered the computer) and the interaction (the link between the computer and user). The intention of studying HCI is to provide usable, safe, and functional systems. Attempting this requires understanding of factors affecting HCI and for features to be developed to achieve the aims. It requires focusing on the users and their needs. A key concept of HCI is usability which quantifies the systems ease, effectiveness, efficiency, safety and enjoyability to use under a single term. Usability is essential to providing user satisfaction. Some factors that contribute to HCI and usability are:

- Organisation Factors – Relevant to the organisation the computer is used within, handling the training, job design, politics, roles, and work organisation.
- Environmental Factors – Relevant to the environmental effects of use of the computer such as noise, heating, lighting, ventilation and health and safety
- User Factors – Relevant to the user themselves, the capabilities of the user and processes they undertake to use the computer as well as motivation, enjoyment, satisfaction, personality, and experience.
- Comfort Factors – Relevant to the user's physical presence such as seating, equipment used and the layout of the system.
- User Interface – Relevant to the user's interface with the computer such as the input and output devices used, the dialogue structures, use of visual display (colours and icons) and instructions (commands, navigation, and natural language) as well as support.
- Task Factors – Relevant to the task that the user and computer perform such as the ease of said task, the amount of time, the skills it requires and monitoring of said task.
- Constraints – Relevant to the limitations of the computer and its development, such as cost, timescales, budgets, staff, equipment, and buildings
- System Functionality – Relevant to the functionality of the computer such as the hardware, software and application needed for the interaction.
- Productivity Factors – Relevant to the productivity results of the interaction such as its effects on output, quality, costs, errors, and innovation

HCI impacts a wide variety of disciplines in a variety of ways and is made to have aspects that apply to all of these disciplines. In many cases, some factors of HCI do not apply or some apply more drastically than others.

(Beale, 2019)

## Data Preparation

According to Couper et al (2001) "Research on self-administered surveys suggests that the design of the instrument may be extremely important in obtaining unbiased answers from respondents". As such, special consideration into the design of the projects survey should be taken and research conducted to assure the design is of high quality.

According to Loughborough University Library (2019) there are a series of stages to designing a successful survey. Firstly the objectives of what is to be known should be defined and the subjects being asked should be considered carefully. The survey should begin with clear instructions that designs should:

"Maximise your response rate by providing clear information and instructions as follows:

- State who you are.
- Outline what the purpose of the survey is and why their response is important.
- Explain how answers will be treated with confidentiality and anonymity (unless agreed with the respondent)
- Provide clear instructions as to how each question should be answered –e.g. whether you are expecting one or more answers, or whether answers should be ranked –and, if so, is 1 high or low?
- How to return the questionnaire and by what date" (Loughborough University Library, 2019)

The order of questions is also a concern as "The placement of a question can have a greater impact on the result than the particular choice of words used in the question" (Pew Research Center, 2011). Separating the questions between their types (closed or open) and topics (demographic, opinion e.g.) is integral to effective design. To structure a survey, the ideal structure may be:

- Intro - Explain the rules and instructions as well as the survey as suggested by Loughborough University Library (2019)
- Questions
  - Closed Ended
    - Demographics
    - Survey Questions
  - Open Ended
- Thank the survey taker.

As shown, questions can be either "open-ended question, where respondents provide a response in their own words, or a closed-ended question, where they are asked to choose from a list of answer choices" (Pew Research Center, 2011). Closed ended questions gather discrete data from the surveyed and allow for statistical analysis such as finding averages and other metrics. Open ended gather the full opinion of the user but require interpretation which can cause bias.

Questions can also cover the persons demographic, to find personal criteria that may impact the outcome of the research. Demographics cover attributes such as age and gender. According to Jovancic (2018) the eight most common demographic questions cover:

- Age - Usually as ranges
- Gender - Usually as Male, Female, Other and prefer not to say.
- Ethnicity
- Education - Usually asks for highest degree.
- Employment - Usually asking for hours (Part time, unemployed, etc)
- Location - Usually a dropdown of specific information
- Marital Status
- Household income

Many of these demographics are omitted due to irrelevance to the question at hand or due to unnecessary reasoning to ask to a specific sample (such as if the survey were addressed to men).

Closed questions vary in many formats such as simply responding yes or no to a statement, agreeing, or disagreeing with a statement or the Likert scale, “scales where respondents express their strength of agreement with each of several statements” (Wuencsh,2015) and is usually a scale of integrals from 1 - 10 where 1 is least agree. Open questions usually ask a question and then allow the user to write their response. Both open and closed have many more potential formats but these are the most common and likely to be used.

The delivery mechanism of surveys varies but currently the most effective may be web based for ease of both the writer and user. Google forms is a commonly used and effective tool for this and will likely be used. (Google, 2019)

Interviews are structured similarly to surveys only intended for smaller audiences (usually either one to one or one to a focus group). They are ideal for obtaining detailed information and where surveys gained breadth on a topic, interviews gain depth. They can be structured (where the interview covers only predefined topics), unstructured (undefined) and semi-structured (somewhere in between). The delivery mechanism varies between phone, face to face and via an online platform and cover a recorded conversation between interviewer and interviewee(s). A structure for interview to follow may proceed as:

- Introduction - Introduce self and purpose of interview, as well as rules and details of the use.
- Questions - Beginning with the simplest (usually demographic)
- Closing Statements - Ask the interviewee for additional comments any questions and thank them for their time.

Once conducted interviews should be transcribed and analysed. Contents of such should be verified with the interviewee.

## Chapter 3 - Literature Reviews

### Overall Chatbot Focuses

Chatbots have multifaceted use and can be applied to a range of fields (Dahiya, 2017) making them particularly suitable for functioning as university support, which is a domain with many requirements ranging from logistical support to emotional (Ciobanu, 2013). Despite this, the development of chatbots in this domain function with an almost complete focus on logistical support and serving as a more accessible FAQ for the university (Patel et al, 2019; Ranoliya et al, 2017; Santoso et al, 2018). Chatbots, specifically those for universities, lack the attempt of providing emotional support to students despite its importance.

Chatbots can function in a variety of ways, namely through rules based or machine learning based functions that alter the behaviour between rigid yet unnatural or natural yet uncontrolled respectively (Schlicht, 2016). Within universities, chatbots largely use a rules-based approach (Patel et al, 2019; Ranoliya et al, 2017) or a hybrid between the two, with leaning towards rules based (Santoso et al, 2018). The regular use of rules-based approaches can lead to an unnatural conversation simulation that whilst suited to the FAQ use of these chatbots, is not suited to other potential uses. The regular use of rules-based approaches causes a loss of potential for uses that are harmed by rigid control. Whilst retrieval-based models would provide sufficient embodiment for this purpose, generative models would enhance this further and help to provide a suitable persona for giving emotional support to users due to its lack of repetition and more human appearance and sincerity. The function of the emotional component should also consider sentiment analysis, or specifically that sentiment analysis is insufficient and emotional analysis would be needed to provide a quality system. Despite sentiment analysis being sufficient when opinions scale between positive and negative, this emotional support requires a system that handles the spectrum of emotions and not a single scale.

The accessibility is an important factor for use. University websites are a common traffic avenue for students and other potential users and so integrating into the website of the university used for would be an effective method (Patel et al, 2019; Santoso et al, 2018). However, this does rely on the ability to be able to integrate with the website and so without this, other avenues of access should be explored that have a similar traffic to the website.

In conclusion, chatbots catered towards university lack use outside of logistic focus and too regularly use rules-based approaches to fit this consideration. A machine learning based chatbot that exploits the multifaceted potential of chatbots would improve upon this issue. The regular use of integrating into university websites is effective for the chatbots function but potentially limiting for development.

## Classification Method Choices for Conversation Modelling and Sentiment Analysis

Generative conversation modelling will be used in the development of this project, as found in the chatbot focuses literature review. On top of DeepLearning being superior to traditional machine learning (LeCun, 2015), generative chatbots require DeepLearning methodology to function and are largely limited to the deep recurring neural net model, Seq2Seq (Adamopoulou, 2020). This model is widely considered to be the most effective deep learning model for chatbots at the current time (Cahn, 2017; Csaky, 2019; Nguyen and Scherbakov, 2018; Yin et al, 2017; Ramamoorthy 2016). Due to these limitations and strengths, this model will be used.

Recurring neural networks are necessary for chatbots due to their handling of sequences of data. Unlike regular neural network models, recurring neural networks allow for sequence input due to their unravelled neuron model. They also account for historical information, data from previous iterations. This makes these network models very suitable for handling text. (Amidi and Amidi, 2019b). Given that the previously discussed LSTM model of RNN's performs better due to its weighting of features to retain, the implementation should use LSTM RNN models for its function.

Building from RNNs, the Seq2Seq model further enhances their suitability for text handling. The model makes use of two RNNs, an encoder and decoder model. The encoder takes input text and outputs it in a vector to abstract the text for processing. The decoder then takes this abstracted data and produces text from it. This is the output or response of the chatbot. Despite Seq2Seq's universal use for generative chatbots, there are issues. Firstly is the potential difference in sizes between encoder inputs and decoder inputs. Padding is needed to assure consistent sizing that is suitable for the model. The words input also need to be formatted into an acceptable data type. The method of doing so for the words is Word Embedding where the semantic meaning of words is displayed graphically with similar words sharing proximity. This proximity is used in the classification process to find similar wording. Another issue to consider is the length loss of data in Seq2Seq. An attention mechanism may be needed to abstract previous data in the Seq2Seq model to handle this issue (Ramamoorthy 2016). Despite the presence of the issue, it is not a concern for this specific project as chatbots are reduced in length in comparison to uses where this may be useful. According to Ludwig (2018) attention is only beneficial for uses where there is a "long span context, such as in the case of text summarization or translation of set of sentences" and that this is not the case for Seq2Seq chatbots. He goes on to say that attention can in fact be detrimental to Seq2Seq chatbots and so should not be used.

In DeepLearning models, overfitting exists as an issue where the data is too closely fit to its training and is incapable of predicting newer data. Prickett et al (2018) discuss the use of dropout layers. It works by removing some of the units of data in each pass to reduce chances of overfitting. They claim, "Dropout increased the model's scope of generalization from novel syllables to novel segments, demonstrating a human-like behaviour". This is integral to the aims and objectives of this project and so will be implemented. To implement a Seq2Seq model that handles these issues, a model using LSTMs, Embedding and dropout should be implemented.

To integrate the emotional component for the chatbot, emotional analysis will be conducted separately on the text and the classification produced input into the classification process of the conversation simulation with the encoder inputs. The emotional analysis will find specific emotions (anger, sadness, happiness, fear, disgust, and surprise as considered standard by Yam (2015)) in its own method. Further evaluation of the emotional analysis structure will be conducted to find the implementation method for this.

In conclusion Seq2Seq has been chosen as the model for conversation modelling and a method specific to emotional analysis will be decided at development. The output of the emotional analysis will be passed to the chatbot to impact the chatbot results as an additional feature.

### Extension - Programming Choices

As an extension to this literature review, discussion, and research of the potential choices of programming support for the Seq2Seq model will be conducted. In comparative reports between programming languages for machine learning, Python stands out as the best performer. According to a recent survey, “77% of ML developers and data scientists currently use Python” (Developer Economics, 2020). Competitors to Python in this field, suffer drastically less usage and as such, also suffer from less community support, an integral feature for this project’s development. One report suggests Python’s outperformance of alternatives is due to it offering “an easier and faster way to build highly performing algorithms” (Voskoglou, 2019). They go on to commend python’s “extensive collection of specialised libraries that come with it” (Voskoglou, 2019), furthering the initial point that python should be used due to its extensive community support. Specific to Seq2Seq usage, a search of GitHub found that the vast majority of Seq2Seq chatbot programs use python and all of the most popular do as well (Dong, 2021; Duan, 2021; Ivanov, 2021; Ramamoorthy, 2021; Wu, 2021). It is clear that Python is the most widely used programming language for the purpose of this project and therefore should be used in development.

With Python chosen as the projects programming language, a package suitable for Seq2Seq should also be chosen. Despite consistent usage of Python as a programming language for this, the package choice is disputed. The three major choices are PyTorch, TensorFlow and Keras. Whilst no study has been conducted into the package’s performance comparisons specifically to Seq2Seq, some reports have discussed the differences in the general scope. According to the reports, Keras is designed as a lighter solution to allow for “fast experimentation” with the software itself (Bhagwat, 2020; Sayantini, 2018; Terra, 2020) specifically via its “ease of use and syntactic simplicity, facilitating fast development” (Sayantini, 2018). TensorFlow focuses more on scalability with a suitability for creation of larger programs (Bhagwat, 2020; Sayantini, 2018; Terra, 2020) and is focused on “dataflow programming across a range of tasks” (Terra, 2020). PyTorch is built for Natural Language Processing (Bhagwat, 2020; Sayantini, 2018; Terra, 2020) and has a “reputation for simplicity, ease of use, flexibility, efficient memory usage, and dynamic computational graphs. It also feels native, making coding more manageable and increasing processing speed.” (Terra, 2020). In comparison, Keras is suitable for rapid prototyping of smaller projects supported by smaller datasets. TensorFlow is suitable for larger dataset projects and finds specific use for object detection. PyTorch has found suitability for flexible programs that require a small training duration (Bhagwat, 2020; Sayantini, 2018; Terra, 2020) . Overall, given the scale of this project, Keras will be used. Despite the other packages potentially being capable of outperforming Keras, the project is early in concept. Keras functions as an API of TensorFlow, allowing for scalability as well as simplicity which is beneficial at this stage without hindering potential expansion at later dates. The cooperation between the two packages is another clear reason that Keras is suitable and why this is the chosen package.

With Python chose as the programming language, the emotional analysis component could also be selected. Whilst creating an independent DeepLearning model for this is a possibility, the package Text2Emotion (Sharma, 2020) performs this. The package appears to be lexicon based rather than machine learning based. Using the package will make development far more efficient than providing an independent developed solution. It produces its own dictionary of five category weighting for the emotions anger, fear, happiness, sadness and surprise and is claimed to have experience in use with chatbots (Band, 2021). This makes it a suitable choice for implementing the emotional classifications.

## Medium Choice for Chatbot

This literature review investigates the choice of medium for user access to the chatbot and aims to find the most effective medium for the project. This will consider the HCI implications of the choice. There are three main categories of choice for mediums, application based (Dahiya, 2017; de Jesus, 2019), website based (Jia, 2003; Patel et al, 2019; Ranoliya et al, 2017; Santoso et al, 2018) and messenger based (Adam et al, 2020; Baron, 2017; CNN, 2020; Schlicht, 2016).

Application based chatbots are those that run in a dedicated program format, specific to the chatbot. They require the user to download the chatbot itself. The biggest reasonings for this medium use are for dedicated purposes where the chatbot itself is the user attraction and not as an aside to another service such as consultation chatbots (de Jesus, 2019) or pure simplicity (Dahiya, 2017). The largest benefit of this medium is its complete focus on the chatbot itself, meaning there is no development factors of integrating the service into others to consider. As well, an application based chatbot is only contained by the app itself and so it faces less performance constraints, allowing the chatbot to perform better.

Development of an application for the chatbot also allows complete control of many HCI factors such as user interface as well as providing potential user control of the device used to access the chatbot giving more comfort factor control.

Despite this, the independence of application based chatbots causes some HCI issues, namely task and system factors. An app based chatbot requires the user to seek the chatbot out and install it. This can discern potential users due to the effort required as well as potential system requirements to run the application. As well, the security implications of downloading a chatbot versus accessing it through a trusted messenger app or website is considerable. Users are very likely to avoid having to download an application for a chatbot due to inconvenience. In summary, application based chatbots are strong when the chatbot is the sole domain of the issue and production simplicity is prioritised over user access.

Website based chatbots are those that run as part a webpage and are accessible through user's browser engines. They do not require the user to download the chatbot itself and are heavily used as a secondary feature of existing sites to serve the same domain (Jia, 2003; Patel et al, 2019; Ranoliya et al, 2017; Santoso et al, 2018). The greatest benefit of website based chatbots is its accessibility to users interacting with an existing service. Many chatbots exist as small windows in websites that a user may use for services such as FAQ's (Ranoliya et al, 2017). In terms of task factors for the user, they do not require any extra effort from user's accessing the site to use and are very effective as side support to existing services. As they exist within the website, it is likely there is complete control of user interface design yet if the chatbot serves as a secondary feature of the website, it is likely to be constricted to a smaller size. Another benefit is that the chatbot is reliant on the server hosting rather than the user's hardware. The only user requirement is a sufficient browser and internet connection. Issues with this medium can be faced in the organisation factors aspect of HCI. Implementing an entire chatbot into a website is likely to cause conflict between the website designers and the chatbot designers, potentially impacting the sites design. Another issue is performance, as integrating into a website may remove the potential for complex chatbots due to poor hosting. As well, integrating chatbots into websites can cause their own development issues. Overall website based chatbots are most effective for those that serve as secondary features to existing domains without being too complex. They are easily accessible and have no hardware requirements.

Messenger based chatbots run within existing messenger services such as Facebook messenger which had “300,000 developers building on the Messenger Platform” (Facebook for Business, 2019). The medium is widely preferred by users as found in a study by Facebook for Business (2016). These chatbots use the existing messaging framework to contain their own communications and do not require the user to download the chatbot, only visit the messaging platform used. They are widely used (Adam, 2020; Baron, 2017; CNN, 2020; Schlicht, 2016). The benefits of messenger based chatbots are that they balance between the development ease of application based chatbots and accessibility of website based. Existing frameworks are available to implement chatbots into the services and potential issues based on a websites own development are avoided whilst staying somewhat accessible. Users are more likely to have access to these services already or more willing to use them than an untrusted third party chatbot. Unlike web based chatbots, there is no potential for issues with the web designers as the chatbot would be completely independent. Issues with hardware are likely to be less considerable and are not user reliant due to use of hosting provided rather than user hardware. The biggest issue is a lack of developer control, especially in consideration of the user interface and the factors of HCI that apply. As the messenger interface is used, developers have no control of features. This does mean that implementing certain features is not needed however as the messenger service likely already has many (such as accessibility features). The limitation can be beneficial for this aspect. Overall, the messenger based chatbot serves as a strong in-between on many of the issues that the application based chatbot and website-based face. Whilst not requiring users to download software they may not trust, accessibility still takes more effort than website integration. As well, development does not require website implementation but does require some messenger API integration.

In conclusion, further analysis with user surveying should be considered before a choice is made. Despite this, messenger based chatbots seem to be the strongest contender as other than the issue of lack of GUI control, the issues it has are not as severe as those impacting alternative choices. Messenger based chatbots look the most suitable due to the ability for development to be independent of website requirements. This is ideal for the project to keep simplicity yet access through notable mediums such as Facebook would keep the chatbot easy to use for the user base. The next stage of the project will survey and interview potential users who should have an impact on this decision.

## Dataset Choice for Chatbot

The chatbot intends to create realistic conversation within a generative model catered towards the university domain. To create realistic conversation, larger datasets are needed. To create domain specific conversation, domain specific datasets are needed. After searching for a dataset to fit these conditions, none were found as expected. A dataset specific to the domain (Edgehill university support) will have to be made but considering the size requirement, manual creation of a suitably sized chatbot would be too inefficient. Csaky (2019) describes neural network chatbots with knowledge base integration, an extra component to a chatbot that allows for the chatbot to handle domain specific information as well as perform generatively. This is essentially a second dataset, specific to the domain and usually smaller in size. The example given (Yin et al, 2017) demonstrates a chatbot used in airports where the location and times are taken from the text and treated specifically through named entity recognition. It can recognise named entities in text and find specific data by searching. This is a suitable implementation for task oriented chatbots. In this project, the domain specificity intends to provide data as if it were a normal part of the conversation and so the named entity recognition would not be completely necessary. It may support the domain specificity but the added complexity for minimal benefit is decided as unnecessary. Instead the knowledge base will be used to classify from directly. This will allow for a general dataset that provides conversation simulation and a specific dataset to the university.

An alternative approach suitable for domain specific purposes is Facebook's bABI dataset (Facebook Research, 2017). Originally used in the restaurant domain (Bordes and Weston, 2016) the dataset allows for use of a rules-based system without loss of natural language from the generation of text. In the case of Bordes and Weston, the task the chatbot aims to complete is divided into 5 sub tasks where performance is measured for each and classification depends on the performance. Joshi et al (2017) describes such a manipulation of the bABI dataset, demonstrating the possibility of doing so in a specific domain and the effectiveness of this approach. Joshi describes it as a "starting point towards the development of sophisticated personalized dialog systems in domains such as restaurant reservation, customer care or digital assistants". Whilst there are issues present due to the relatively new nature of this approach, it is the most specific to solving the issue of creating a suitable domain based generative chatbot and is well suited to the chatbots aim of supporting university students.

In conclusion, to fit the iterative development cycle of the process, the knowledge base implementation will be used as a separate stage to development. Whilst bABI appears useful for chatbot implementation, it structures the entire chatbot around domain specificity. The project aims to provide natural conversation as well as domain specificity so separating the two will be more suitable for the process structure. This allows for development to proceed in separate stages to adhere to an Agile process and allow for separate analysis at each stage. With knowledge base use, a foundation dataset must be selected for use. There are many choices available for this option. An article (Nguyen, 2019) summarises the best chatbot datasets based on needs. The article discusses datasets suitable to both conversation and support. The chosen dataset is intended to provide the conversation component individually so the dataset will be selected with conversational data focused on. The article describes the Cornell movie dialogue corpus, which "contains a large metadata-rich collection of fictional conversations extracted from raw movie scripts: 220,579 conversational exchanges between 10,292 pairs of movie characters involving 9,035 characters from 617 movies" and has a larger data size than alternatives suggested. At this time, this will be the dataset used for the project with conjunction of a knowledge base developed for domain specificity.

## Feature Review of Existing Chatbots

This literature review will examine documentation regarding existing chatbots and compare the features that they have chosen to include or exclude as a method of evaluating the features that should be included in the project. Most of the examined chatbots will be generative but there may be examination of features others of varying types offer to understand the wider range available.

The first feature of existing chatbots found was the use of Knowledge Bases as shown in various examples (Wen et al, 2017; Zhao et al, 2017; Santoso et al, 2018). The uses vary in many ways, but all essentially provide a secondary dataset format to provide domain specific functionality. According to Csaky (2019) "KBs store information discretely, their integration with neural network-based encoder-decoder models is not trivial". The simplicity of their integration to provide domain specific functionality to chatbot models would be a very strong feature for the project that intends to provide realistic speech but have a domain-oriented approach. In Zhao's chatbot (2017) it works by replacing some aspects of text with general tokens for recognition, in their example to recognise locations. By separating certain aspects of text that are likely to be domain specific such as this, the aim of catering to a specific field can be achieved. However, the previous investigation evaluated that name entity recognition such as this will not be utilised for minimal benefit to the project specifically.

Additional input features are another feature and are very specifically needed for this project, specifically the emotional component. An existing chatbot utilises this to create a personal profile that provides further classification (Li et al, 2016) or in a more relevant case, classify using emotion as well as text (Zhou et al, 2018). By taking a classification value for these inputs and integrating them into the text processing function, they can impact the process of the chatbot classification as well, allowing for reaction to non-text specific features. Integrating an emotional component will be vital to emotional text processing.

Secondary features shown in existing chatbots also include attention, which expands the decoding process of each word based on previous steps to aid holding the relevant information and keep "attention" of what is being processed. There are two main uses of this function (Bahdanau et al, 2014; Luong et al, 2015) and both vary in their approach. Despite their inclusion in many chatbots, it has been recommended against in chatbots with similar purpose to the projects own (Ludwig, 2018). Due to this, attention will not be included as it may be detrimental to the performance.

Similarly to attention, pretraining is another widely used feature with varying implementations (Akasaki and Kaji, 2018; Mikolov et al, 2013). Pretraining involves "initializing the parameters of a model randomly the model is first pretrained on some data or some other task that is different from the main task that the model needs to be applied to" (Csaky, 2019). One aspect of the model that can be pretrained is the embedding process for the foundation dataset. Without pretraining, the model would have to establish the words semantic graphical distances itself which would add to the complexity and process time. Global Vectors for Word Representation (GloVe) would provide these weights (Pennington et al, 2014). The service provides pre-trained word vectors based on its own learning algorithms and data. In situations where hardware availability is lacking, this is a strong consideration as a feature to use. The embeddings provided would be accurate without the inefficiency of use on insufficient hardware. Despite this, there is potential for missing data and the context that Glove uses would be separate from self-trained, impacting the weightings. With hardware limitations, this is a strong feature to consider.

Outside of these features specific to generative models are some provided in other contexts. De Jesus (2019) describes various emotional support chatbots that provide external tools for tracking the user and certain things they may discuss. Whilst this is not a priority, this feature may be considered as an extension to the project if available. As well, for rules based chatbots database backing (Patel et al, 2019) and AIML checks (Ranoliya et al, 2017) are used. Whilst they are contradictory to the nature of the generative chatbot, these features have a small possibility of being integrating in lighter formats for issues such as knowledge base integration.

In conclusion, many of these features seem useful. Knowledge bases and additional inputs are key to developing certain requirements of the project. Attention is considered detrimental to the context of this chatbot by Ludwig (2018) and will be avoided due to this. Pretraining, specifically using GloVe, would provide an efficient method of implementation for certain elements of the chatbot that are not needed to be controlled by the projects own training process. This is particularly suitable when there may be hardware issues. The embedding of the words into representative vectors would be considered for this project if hardware is insufficient as if the costs can be spent, use of pretraining is somewhat detrimental due to lack of contextual reasoning and potential for missing scores. However, if the costs cannot be afforded, pretraining is a strong feature to include. Non – generative features such as external functions as well as rules-based implementations are not directly applicable but could be considered for extension to the project.

## Literature Review Conclusion

In conclusion of the literature reviews conducted to find information regarding the project, some pre-emptive decisions can be made. The overall chatbot focuses have been defined as creating a chatbot that supports both informational and emotional desires of university students with a focus on creating realistic conversation via a generative model. The generative model requires sophisticated classification methods through DeepLearning and the recurring neural network structure seq2seq is ideal for this. The Seq2Seq model developed will use the LSTM RNN implementation for its encoder and decoder as well as an embedding layer to provide vector representation of text. A dropout layer will be used to attempt to avoid overfitting. Emotional analysis will make use of the “text2emotion” package (Sharma, 2020) due to its simple integration. The programming language Python will be used for this and the package Keras imported to the Seq2Seq model. The medium choice cannot be fully made at this time, but the messenger-based method appears most suitable. Surveys and interviews will make this choice before development. The dataset is decided to be used as a conjunction of both a regular foundation dataset, the Cornell movie dialogue corpus (Danescu-Niculescu-Mizil and Lee, 2011) as well as a developed knowledge base at this stage. The feature reviews of existing chatbots have shown that secondary classification of emotions and knowledge bases are key and that pretraining should be incorporated to provide better performance and development efficiency if hardware available proves insufficient.

## Chapter 4 - User Requirement Data Collection

### Interview

The interview conducted will be a one-to-one conversation with a “chat with our students” worker. The interview will follow a semi structured approach and be face to face. This section will detail the design and preparation of the interview itself.

Certain documents had to be established for the interview. Below is a list of them with the location of such documents in the appendix listed.

- Participant Information Sheet (item 1)
- Participant consent form (unsigned) (item 2)
- Participant consent form (unsigned) (item 3)
- Interviewee email discussion (item 4)
- Interview Transcript (item 5)

This design serves as a brief plan of what should be discussed during the interview itself.

- Introduction - Discussion about the interview, introducing it and its purpose. Verbal consent will be asked for here as well as how the interviewee can be addressed.
- Closed questions regarding their role.
  - What course they studied / study at the time of working for the system
  - What hours they worked there
  - How much they were paid.
  - How often they would be requested for support.
  - What times they would provide support
  - How quickly they would respond.
  - What year were most of the students they spoke to in?
- Open questions topics
  - Categorisation of support
  - Common topics of support sessions
  - Potential issues encountered with students.
  - Language used during conversation (slang, formal, etc)
  - Own issues with the service
  - Feedback for the service
  - Discussion about availability and access for support
  - Course specificity
  - Importance of human interaction

With the designs and interview scheduled the interview could begin. See appendix item 5 for the transcript of the interview. This transcript has been authorised as valid by the interviewee as shown in item 4 of the appendix.

To evaluate the interview, the points the interviewee had about each of the topics and interviewer's own reaction to them will be discussed. This will only cover issues in the interview about the project itself and not issues regarding demographics or consent although they may be considered for certain points. Evaluations made about the interview have been authorised by the interviewee as valid as shown in item 4 of the appendix.

#### Topic 1 - Employment for Chatbot Service

- Interviewee still works there.
- Paid around 10 pound an hour.
- Works 1 - 2 hours a month, 3 - 4 when an event is on
- Usually works Wednesdays and Thursdays, between 9am to 9pm
- Receives outside hour messages, regularly from overseas students.
- Can work outside hours.
- Response time within hours is usually within the hour, until next morning outside of hours

Evaluation of this topic shows that there is benefit to the project proposed as it has no wage requirement, works constantly (when functional) and has near instant response.

#### Topic 2 - Categorisation of Support

- Interviewee agreed that support can be considered informational or emotional.

#### Topic 3 - Common Dealings

- Most dealt with "university life" and course specifics
- Interviewee course was "computing and security, networking and forensics".
- Majority are course related.
- University life is related to what is available on campus such as recommendations on where to go for drinks and such.

Evaluation shows that university life and course specifics are largely a subset of informational support. Given that they are the most common, they should be focused on.

#### Topic 4 - Students

- Mainly students finishing with a level.
- Commonly first years needing help on campus.
- Sometimes second and third years

Evaluation shows that the system is largely used by those unfamiliar with campus and that the system should potentially focus on these groups.

#### Topic 5 - Conversation timings

- Dependent on topic
- Example given of a long lasting was 15 minutes.

Evaluation shows that the conversations would likely max out around 15 minutes based on the interviewee giving that as an example.

### Topic 6 - Medium Choice

- Interviewee felt that affiliation with university was important.
- Messenger or third-party use would need certification / endorsement.

Evaluation shows that interviewee was most receptive to web integration purely due to clarification that it would be university affiliated. The other two methods would need endorsement, although messenger would likely need less due to innate safety of messenger app use.

### Topic 7 - Interviewee Role

- Students send questions.
- Response via links to solutions or conversation
- Limited to university discussion
- Emotional support limited due to this rule, has to be relevant to university such as dropping out or stress.
- If topics became too involved, trained supervisor would be included.
- Only happened once.
- Informational easier category to handle due to existing information.
- No specific techniques for support of both categories

Evaluation shows that the system the interviewee uses is catered towards responses coming later using links like an FAQ system. The limit to university discussion is a concern for the chatbot as it is likely the university enforces a rule to ensure that the chat service is not used for issues they may be considered liable for. Enforcing this rule in a chatbot would be difficult. For project purposes, this rule will not be implemented due to issues that could arise but should be considered if future university implementation is considered, especially considering the certification of the university was important to the interviewee. As well, the emotional support on offer is heavily limited by this rule and for the chatbot to provide it and the aims to be achieved, the rule should not be considered. Another consideration is that the support service redirects to fully trained staff when the conversation becomes too much, potentially a recommendation service for the chatbot when certain topics that the bot would be incapable of handling should be considered. This situation was rare so prioritisation may not be entirely important. The interviewee discussed that informational support is easier to handle as well due to the existing information. Use of existing information to provide support should be considered. The lack of techniques also mean that no framework would be followed, potentially something the chatbot could do but would mean designing a framework that could be complex and potentially limit conversational simulation. This would not be considered at this point but maybe in the future after evaluating the conversations the chatbot could provide.

### Topic 8 - Features

- Hyperlinks
- Uses access platform.
- FAQ section with pictures and comments
- No recommendations or criticisms from students
- Interviewee believes that the platform should allow for easier cross support staff communication.
- Displays previous conversations.

Evaluation shows that the “access platform” should be investigated for this. Links from the chatbot should be considered, would likely be an extension to the knowledge base. FAQ would be an external feature that may be beneficial in the future but not considerable as a part of this project. The suggested feature would have no implication given the automation of the system. Display of previous conversations allows for reminder of topics which could be a potential extension feature for the chatbot.

After the interviewee revealed that the current system uses the access platform framework, further analysis of the access platform was undertaken. After examining the site (The Access Platform, 2021) the system used has very similar design and functionality to most messenger apps such as Facebook messenger with the aforementioned features as well as an administrative control panel for monitoring conversations. Given that the product intends to allow for unmonitored conversation and that the aforementioned features are going to be considered, this further research found very little.

Despite this, the similar GUI design to messenger apps does support that as the mode of integration.

## Survey

The survey conducted is planned in brief below, with certain elements decided in text to be implemented.

- Google Forms the delivery mechanism and distribute via email messaging service.
- University Students at edge hill university would be surveyed.
- Small sample taken from people who study at edge hill contacted via email messaging services.
- Around 20 survey takers predicted, small value may cause issues so data should be handled with care.
- Introduction - Brief introductory text stating who I am, the purpose of the survey and the reason their response is needed, explain the data handling and clear instructions on how to answer.
- Questions
  - Closed
    - Demographics - Question the surveyors age, course they are taking and gender.
    - Survey Questions - Multiple choice, Likert scale (1-7 used to allow for neutral agreement)
  - Open
    - Text input, based on previous multiple choices such as explaining why they disagree.
- Conclusion - Thank them and leave contact information for further questions.

The introduction to the survey is an important component for assuring results gathered are high quality by providing sufficient information to those taking the survey. Below is the introduction used in accordance with advice from Loughborough University Library (2019).

*I am running a survey on current Edge Hill students use of the university support systems,  
specifically the "Chat with our students" service.*

*The data gathered will be used in the project to develop an automated chatbot support system for  
Edge Hill University Students. Your response regarding your interaction with the current system will  
be important to understand the user requirements of this system and how it should compare to the  
existing.*

*The information gathered will be anonymous and only used in statistical context with no identifying  
factors to attribute responses to individuals.*

*The survey is made up of various question formats. For multiple choice questions, click the circle  
that applies most. For text questions, type your answer in the box labelled "Your answer". For  
dropdown questions, click the box labelled "Choose" and select an answer from the potential  
responses. For scaling questions, a question or statement will be asked, and 7 choices given. These  
choices usually scale between a very negative response at 1 to a neutral response at 4 and a very  
positive response at 7. Please chose the value you believe applies to the statement. Many questions  
are mandatory and marked with a red asterisk (\*) to show this. Those that are not, are usually  
dependent on previous questions as stated in the question itself. Please leave these blank if they do  
not apply.*

*This survey will take around five minutes. Thank you for participating and if you have any questions  
or feed please contact me at [23926643@edgehill.ac.uk](mailto:23926643@edgehill.ac.uk).*

The survey is made up of seven sections. The first introduces the survey with the designed introduction text. The second asks some demographic and introductory questions, these are age, gender and course followed by questions regarding the support system. The third section covers some topics depending on if the user has not used the current support system such as if they were aware of its existence. The fourth covers questions depending on if they had used it. The third and fourth section are accessed depending on if the user answers if they had used it. Section five asks some questions on what the user prefers in terms of features and access such as their preferred medium. Section six makes some opinionated statements regarding chat support and asks the user to agree or disagree. Section seven closes the survey with a box asking for potential feedback.

The design for the survey itself is shown in the following appendix items.

- Initial Survey Design (Item 6)
- Initial Survey (Item 7)

Pilot testing was conducted to find feedback prior to survey use. The designs were distributed to two pilots for their feedback.

Pilot test A suggested addressing the users own communication ability via a chatbot as well as the addition of more scenario-based questions. A question will be added addressing the format of users seeking support. A scenario-based question will not be added as introducing a scenario may distract from the opinions intended to gather and introduce biases that may intrude on the data.

Pilot tester B criticised the use of questions that assumed the user had used the “Chat with our students” support system. They suggested that these questions only be available to those who had used the system (like the chat with our students use questions section). Given that this would restrict a large portion of questions from being answered, the questions have been reworded to be applicable.

The conversation and changes discussed are demonstrated in the following appendix items.

- Pilot A Conversation (Item 8)
- Pilot B Conversation (Item 9)
- Pilot Changes (Item 10)

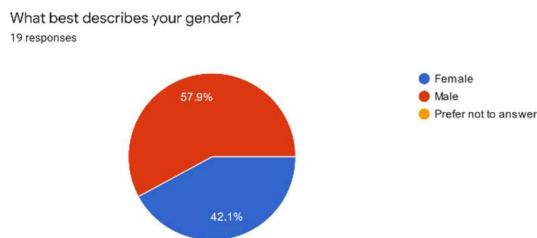
Now that the survey was complete the next step was distribution which would be done via email to a chosen sample. The survey with the pilots’ changes implemented were sent. Of the 21 contacted, 19 responded.

The distribution email sent is shown in the following appendix item.

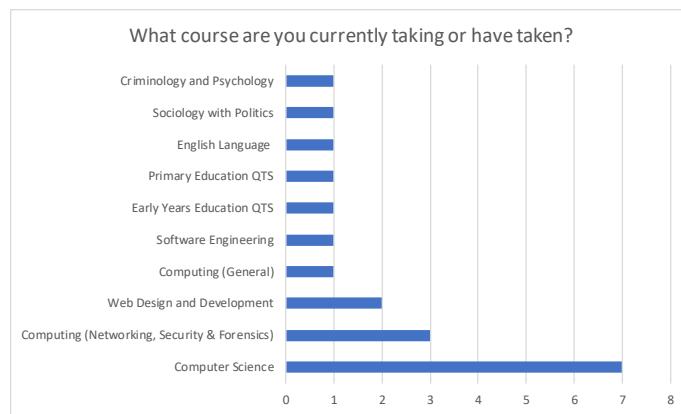
- Distribution Email (Item 11)

### Survey Evaluation

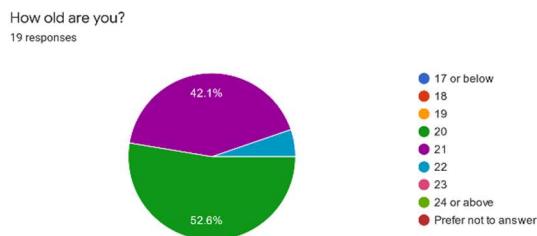
The first group of questions within the survey are simple demographic ones. Whilst no data can directly be gathered from these, certain assumptions can be made about other questions and their responses. Figure 3 shows respondent gender with an imbalance towards male responders although only by about 20%. The balance is somewhat suitable for making judgements about both male and female responses. Figure 4 shows the respondent's course. A large bias is shown here towards responders taking computing related courses with 14 of them. Outside of this, there is a large spread of course types and variance so whilst the data may be skewed towards opinions of one type of student, many student types are considered. Figure 5 shows the ages of respondents. Most asked were in the same university year so an age range of between 20 to 22 was to be expected and as the survey is aimed at university students, this age range is suitable.



*Figure 3 - Gender Demographics*



*Figure 4 - Course Demographics*



*Figure 5 - Age Demographics*

This question asks if the user agrees with the categorisation of either support into informational or emotional. As figure 6 shows, all surveyed agreed that this categorisation applied. This supports the original definition and purpose of the chatbot as it aims to apply itself to both of these. Note that whilst the image below shows the question condensed, respondents could see the entire question "For my project, I have defined two categories of support. These are informational (such as support regarding directions to locations on campus) and emotional (such as support regarding concerns like stress). Would you agree with this categorisation?". Note also that a follow up question was asked for respondents who disagreed to provide their categorisation types. As none disagreed, none answered the follow up question.

For my project, I have defined two categories of support. These are informational (such as support regarding directions to locations on campus) and e...tress). Would you agree with this categorisation?

19 responses

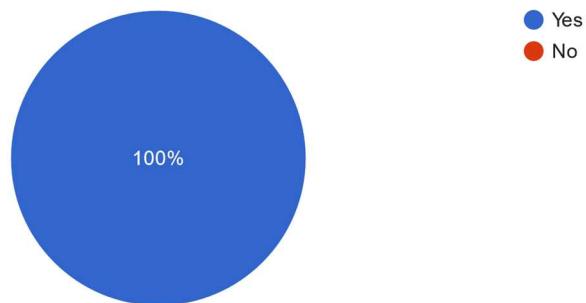


Figure 6 - Categorisation Question

The next question asked if users had used the current system before. As seen in figure 7 only 10% had used the system. Whilst this may show potential issue with the system, use of support systems is expected to be low. From here, the survey splits between users and non-user with questions catered towards each to investigate potential issues or views about the system.

Have you ever used the "Chat with our students" service Edge Hill University provides?

19 responses

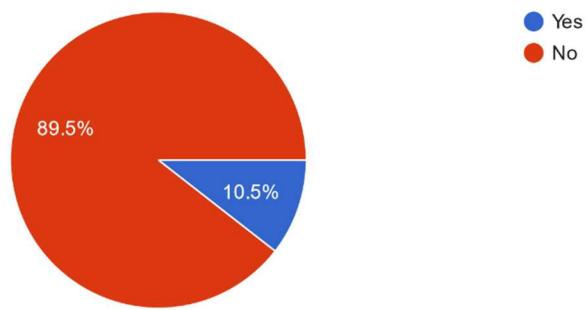


Figure 7 - Current System Users

The respondents that had not used the systems were redirected to two questions. The first (figure 8) asks if they were aware of the systems existence, around a third were not showing that a contributing factor to the non-use was the lack of awareness and distribution. This is not an issue that the project aims to tackle however it should be considered, especially when regarding the medium use. The current system is implemented into the university website which all respondents will have accessed at some point. If the web implementation (which is likely to have the most awareness due to traffic towards it) has low awareness, then the other mediums are likely to be even worse than this. The second (figure 9) is intended for those who were not aware of the system however the respondent amount does not match so there may be some responses that were not intended. The question asks that if they were aware, would they have used it and found that 37.5% would have. This shows that awareness is an important factor for the userbase as many potential users were lost due to lack of it. Despite this, the majority still would not have used it and so it can be assumed that awareness of the system is not the only issue and is not the most major one.

Were you aware of the existence of the service before taking this survey?

17 responses

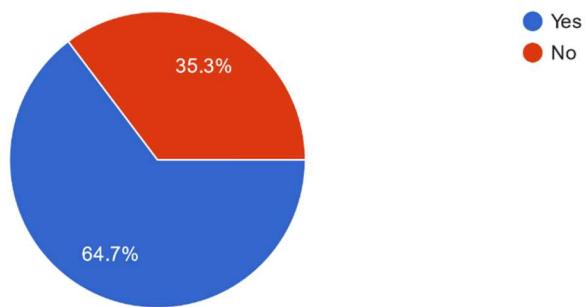


Figure 8 - Awareness Question

If not, would you have used the service? (chat with our students is a discussion service where you can ask specific students questions about Edge Hill University)

8 responses

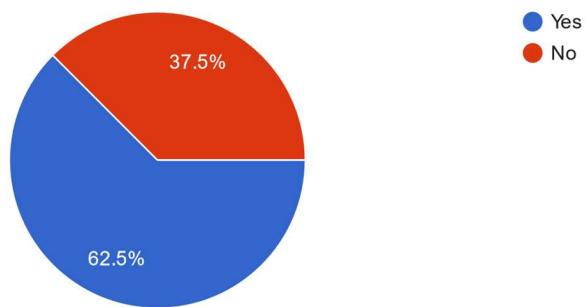


Figure 9 - Hypothetical Use

The respondents that had used the system were questioned on the nature of their use. Figure 10 asks how many times that the user had used the service and found that reuse was low. One user had used the system once and another between 2-3 times. Regular use seems unlikely, potentially due to their issues being resolved or not wanting to use the system after the initial uses. Figure 11 asks when the users had used the system and found from the two answers before first year and second year. Earlier year use was expected to be most common due to inexperience yet the response of second year shows that use in later years is still a potential. Figure 12 asked what categories their use would fall under. Despite the previous agreement of informational or emotional, one user answered, "Progression steps onto a postgrad course". This could be categorised as a sub type of informational support so the responses can be treated as if all respondents used informational support. The extra detail of nature of support given is helpful, however. It can be assumed that the second-year user used this for postgrad support. This and the informational support before first year show that the users have a trend of using the current support service as a preparation tool for upcoming stages such as undergraduate study or postgraduate study. Given that 100% of respondents can be assumed to be looking for this type of support, a focus on this is potentially useful.

#### How many times have you used the service?

2 responses

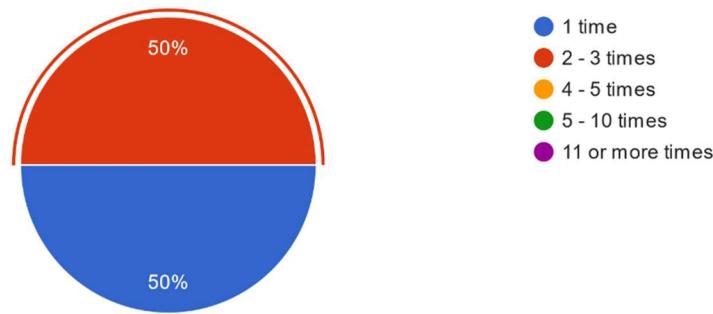


Figure 10 - Use Amount

#### What time of your study did you find you used the service most?

2 responses

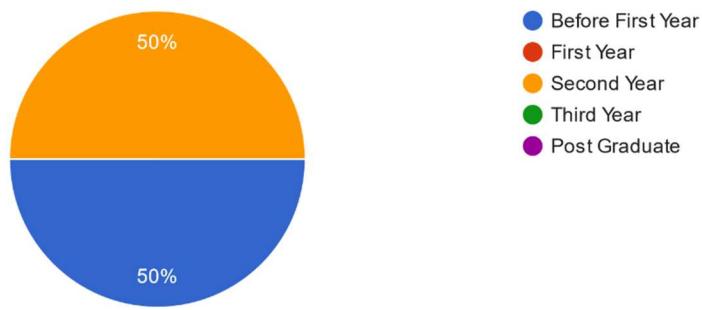


Figure 11 - Use Timing

Referring to the previous categorisations, which do you believe you used the service for more? If you do not agree with the categorisations, please ...cribe what you found you used to the service for 2 responses

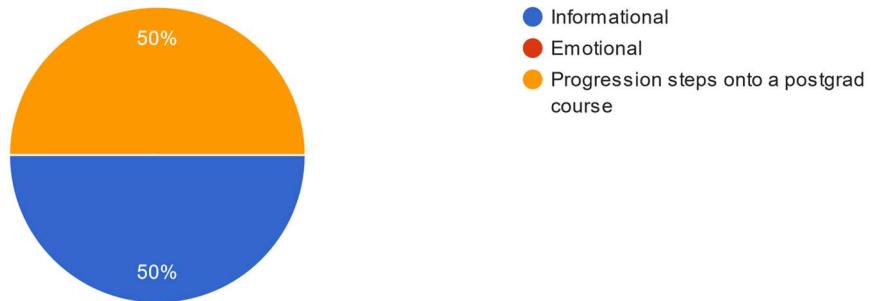


Figure 12 - Use Categorisation

Respondents were now answering the same questions again. The first in the group of this were three Likert scale formats based on their opinion regarding the potential mediums.

Figure 13 shown response to web implementation, with a positive outlook in the mean. The mode shown that commonly, users were neutral to this. Despite this, the median shown a slight lean of positivity. The max and min both shown the most extreme responses and the standard deviation shown a wide spread of responses.

Figure 14 shown similar responses to messenger implementation. The mean was slightly positive with a mode agreeing with the implementation. The median also followed the mode showing a lean towards agreement however the max value shown that the extreme responses were not in full agreement (min still shown an extreme disagreement). The standard deviation shows a considerable spread throughout responses.

Figure 15 shown a strong trend of dislike for software implementation. The average response was negative with a mode and mean of also negativity. Despite this, the extremes still shown complete polarity with the max being at full agreement and min being to full disagreement. The standard deviation also shows a widespread.

In comparison of the three questions, only web implementation and messenger can be considered. A strong aversion to independent software was shown by those surveyed so this shows that this medium is almost agreed to be unwanted by all potential users. Web implementation and messenger implementation shown similar results with some distinctions. Web implementation shown that the majority of users were neutral to the idea yet those that were supportive were in stronger agreement. Messenger implementation shown that the majority were supportive yet to a lesser strength than those supporting web implementation. The spread of both were similar. In conclusion, web implementation would be the stronger choice to provide more satisfaction and messenger implementation would be the stronger choice to cater to more users. Catering to more users is more important than catering more to specific users for this project so this data supports use of the messenger implementation.

How willing would you be to access a chat support service for Edge Hill university via their website?

19 responses

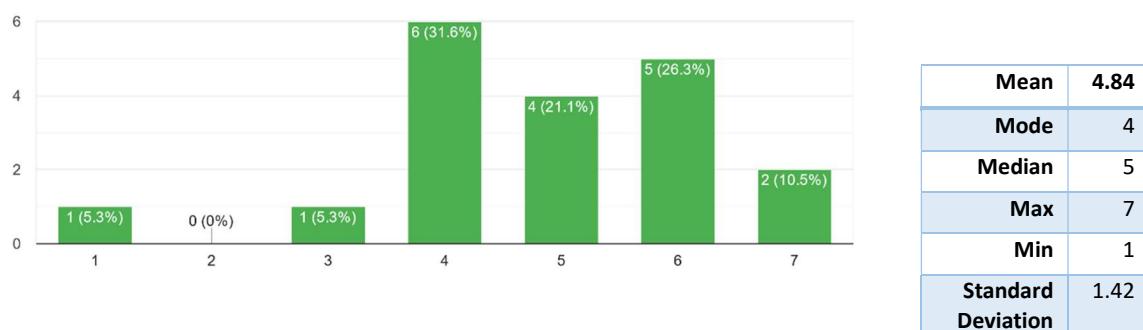
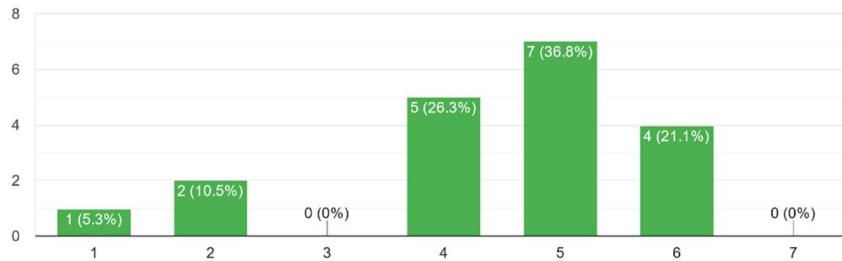


Figure 13 - Web Implementation Question and Metrics

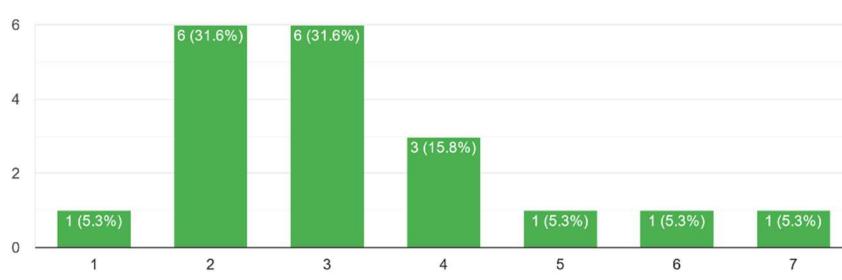
How willing would you be to access a chat support service for Edge Hill university via an existing IM service (such as facebook messenger)?  
19 responses



<b>Mode</b>	5
<b>Median</b>	5
<b>Max</b>	6
<b>Min</b>	1
<b>Standard Deviation</b>	1.39

Figure 14 - Messenger App Implementation Question and Metrics

How willing would you be to access a chat support service for Edge Hill university via independent downloadable software?  
19 responses

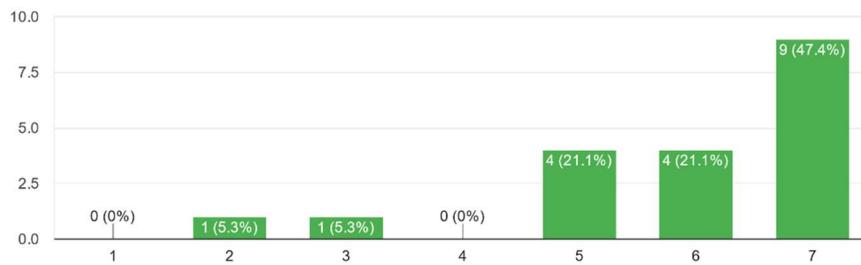


<b>Mean</b>	3.21
<b>Mode</b>	3
<b>Median</b>	3
<b>Max</b>	7
<b>Min</b>	1
<b>Standard Deviation</b>	1.47

Figure 15 - Independent Software Implementation Question and Metrics

Figure 16 graphs responses to a question asking the importance of device accessibility. This found that most responders felt that the chatbot should be accessible to both PC and mobile, stressing the importance of this feature. With the mode being in complete agreement and many following, this would be an important consideration.

How important is having the chat service be accessible to both mobile phones and PC's to you?  
19 responses



<b>Mean</b>	5.89
<b>Mode</b>	7
<b>Median</b>	6
<b>Max</b>	7
<b>Min</b>	2
<b>Standard Deviation</b>	1.41

Figure 16 - Device Accessibility Response and Metrics

Figure 17 shows the results of a question asking how confident respondents were at expressing issues through text chat. Whilst the majority found that they could, 20% found they could not effectively do so. This shows that special consideration may be necessary to assure that all users are capable of effective use of the service however given that the majority are capable, any features implemented to do so should be considered extensions. A follow up question asked for feature suggestions to help those that were incapable, the responses were:

- Phone call
- Facetime
- Zoom
- In person
- Video call

A common factor between the implementations were the ability to communicate via voice. Certain features would be unnecessary or incompatible with the nature of the project, automation and so this abstraction of all the requests is the chosen extension feature to be implemented if given the time.

Can you effectively communicate issues you are having via text chat?

19 responses

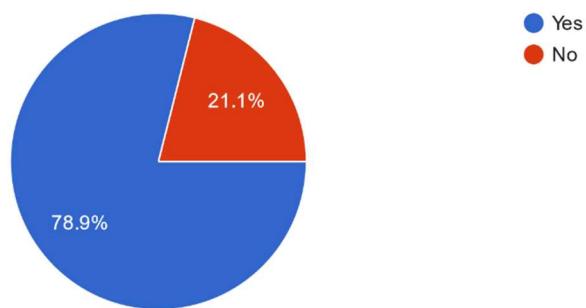


Figure 17 - Effectiveness of Communication Question

Similarly to device accessibility, figure 18 shows a unanimous agreement that accessibility features are also important only to a lesser extent. With a mean of five, the average user believed it was important however standard deviation was high showing a disparity caused by some users who felt it would be unnecessary although they are the minority. A follow up question asked what features they felt should be included. They were:

- Contrast Modes
- Human Contact number
- Voice messenger
- Ability to send images and videos (Map for example)
- Readers (Text to speech?)
- Option to send text messages.
- Vibration API

Many of these features are included in some implementation methods. Contrast modes would be included to a device itself so implementation into the software display would be unnecessary. A human contact number may be considered as an extension feature yet would remove some of the benefits of an automated chatbot. At this current stage, this will not be considered but if the chatbot were to be used commercially in the future, the feature might be necessary. Voice messaging, referring to the previously mentioned extension feature of voice communication. Image sending and videos would be natively included with messaging implementation only a way for the classifier to recognise the images would need to be implemented. This is not a current goal and should be considered an extension. The readers response was vague yet can be interpreted as a text to speech function. Once again, this would likely be included in the implementation device itself (computer or phone) and so implementing this would be unnecessary. The option to send text messages is also another unclear response so it can be assumed that it means communication via phone specifically via text instead of internet. This would be difficult to implement and unnecessary at this stage but may be considered as an extension requirement. The vibration API would be only implementable on phones and provide little accessibility. Overall, these features should be considered as extension with varying priority. Using the messenger service, particularly one with many accessibility features, would help serve many of these issues with little effect on development.

How important are accessibility features for you?

19 responses

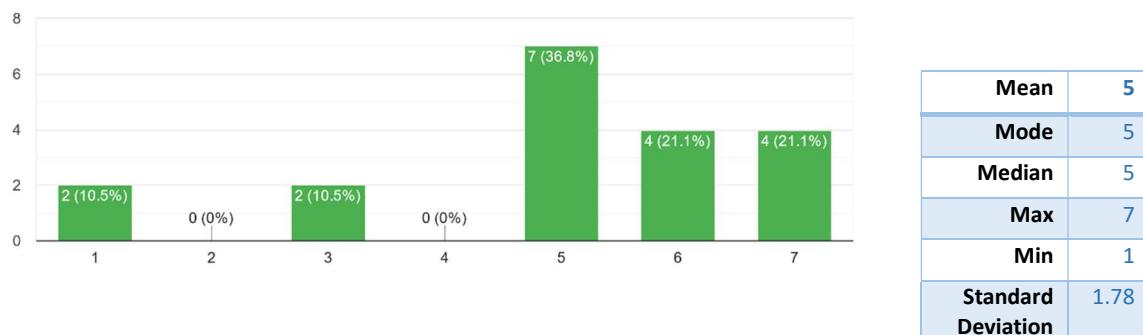


Figure 18 - Accessibility Feature Responses and Metrics

This section handles questions regarding respondent opinions regarding certain topics.

Figure 19 shows the response opinion to accessing the current system. The majority found that currently accessing the system is easy with a mean of agreement and a mode of strongly agreeing. This may suggest that the web integration of the current system is the strongest medium to use.

Figure 20 shows respondent opinion regarding the usefulness of chat support. Similarly, a strong positive response was given with no respondents actively disagreeing showing that the project at its core is justified.

In figure 21, when asked if they preferred discussion to resolve issues rather than finding the information plainly, they agreed largely with the most common response being in full agreement. This shows that most users would want to use a chat support system over normal support yet given that they did not with the current support system (only 10% did) this likely shows potential need for improvement over the current system.

In figure 22, when asked about privacy concerns the opinions were largely split. No respondents felt neutrally, and standard deviation was high. Despite the split, there is a slight lean towards lack of privacy concerns. Despite this, the split shows that privacy is important to around half of the potential users and the benefit supplied via use of a chatbot is not unnecessary.

Similarly, when asked about comfortability asking for emotional support to human figure 23 shows a similar split with negative leanings this time. The lack of neutrality and spread show both of these are contentious issues. Given that the lean is slightly negative towards feeling uncomfortable getting support for emotional issues from humans it is clear that a chatbot would provide a necessary service to some people to avoid potential of emotional support from another human.

On the other hand, when asked the same question about informational support, figure 24 shows a near complete agreement to be comfortable discussing these issues. This shows that for a chatbot to distinguish itself, providing informational support is not a particularly wanted trait.

Figure 25 shows that respondents had a varied preference to discussing issues with a human rather than a bot regardless of conversation quality. This varied massively between respondents however so this would not completely stop users from using the service for many users, but it is a concern.

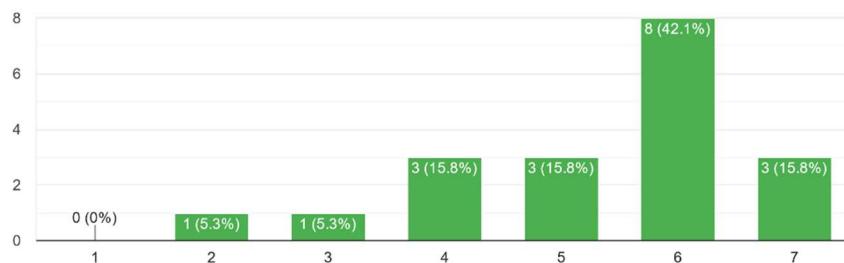
A similar split to that when asked about comfort in emotional support and privacy is also found when asked about the concern of having to wait for support. In figure 26 the mean shows an average agreement although the standard deviation shows disparity in concern. Catering to those who do find this to be an issue would distinguish chatbots as a suitable support provider alternative, yet this appears to be only important to around half of potential customers.

Figure 27 shows many a unanimous agreement that anonymity is important to users with a high average response opinion and very little disagreement. The ability to stay anonymous would be an easy implementation to a chatbot as storing the user's identification would actually be more complex than not. As such, this may be a good way to distinctively serve user desires and make development more efficient simultaneously.

Figure 28 shows a similarly unanimous agreement that users strongly felt constant availability is important. This is supplied by the use of automation and is a strong reason to justify this project from responses given.

Finally, figure 29 looks for respondent priority between support quality and conversation simulation quality. It finds that the vast majority would prefer support quality than conversation simulation. This should be considered during development, to assure that these needs are met.

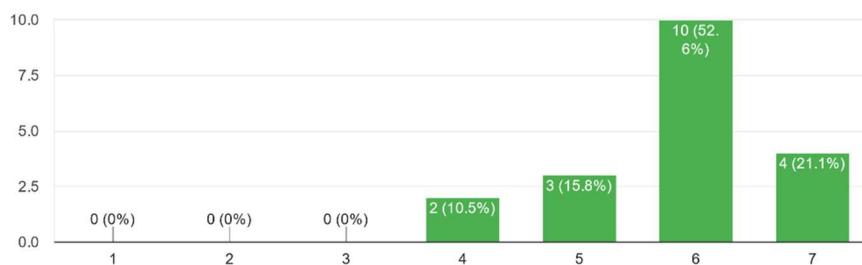
If I were to use the "Chat with our students" service, I would find accessing it easy  
19 responses



<b>Mean</b>	<b>5.32</b>
<b>Mode</b>	<b>6</b>
<b>Median</b>	<b>6</b>
<b>Max</b>	<b>7</b>
<b>Min</b>	<b>2</b>
<b>Standard Deviation</b>	<b>1.34</b>

Figure 19 - Access Opinion Response

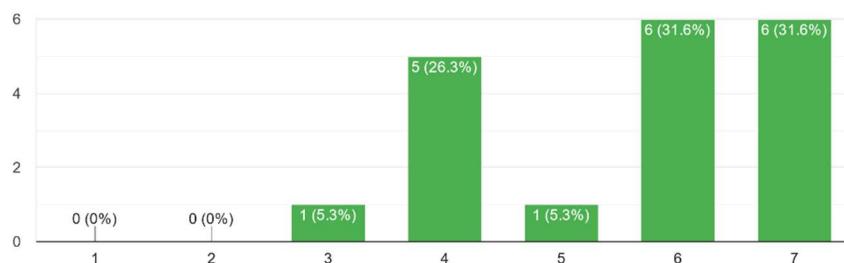
Chat support systems are useful  
19 responses



<b>Mean</b>	<b>5.84</b>
<b>Mode</b>	<b>6</b>
<b>Median</b>	<b>6</b>
<b>Max</b>	<b>7</b>
<b>Min</b>	<b>4</b>
<b>Standard Deviation</b>	<b>0.87</b>

Figure 20 - Useful Opinion Response

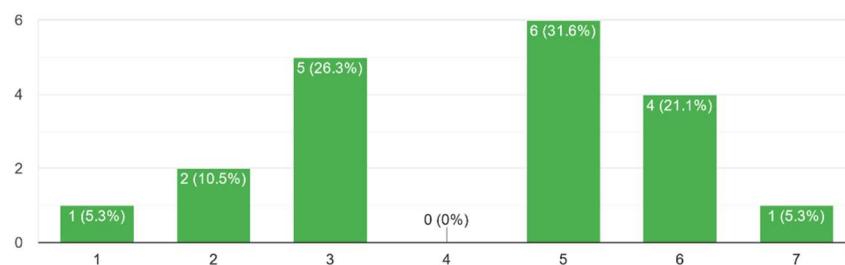
I prefer to discuss issues rather than simply being given the necessary information  
19 responses



<b>Mean</b>	<b>5.58</b>
<b>Mode</b>	<b>7</b>
<b>Median</b>	<b>6</b>
<b>Max</b>	<b>7</b>
<b>Min</b>	<b>3</b>
<b>Standard Deviation</b>	<b>1.31</b>

Figure 21 - Discussion Opinion Response

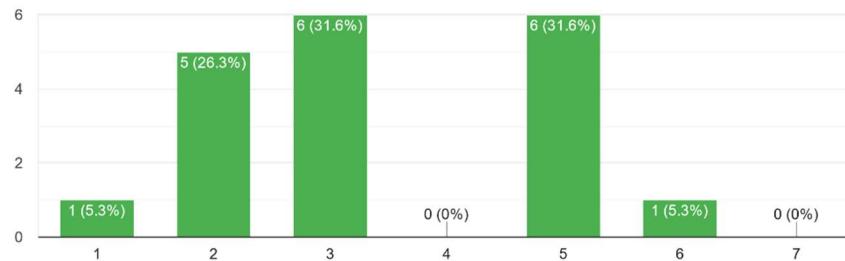
Discussing issues with other students via a chat support system would raise no privacy concerns  
19 responses



<b>Max</b>	7
<b>Min</b>	1
<b>Standard Deviation</b>	1.65

Figure 22 - Privacy Opinion Response

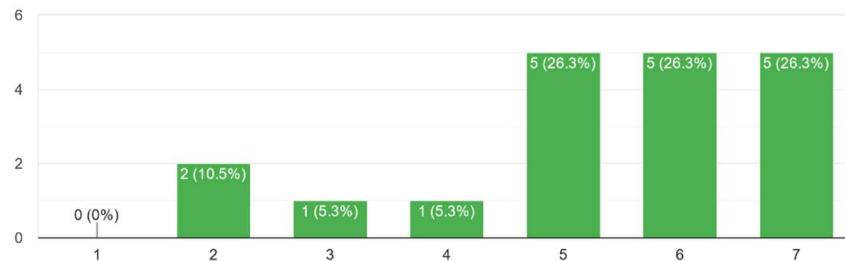
I would be comfortable discussing emotional issues with other students via a chat support system  
19 responses



<b>Mean</b>	3.42
<b>Mode</b>	3
<b>Median</b>	3
<b>Max</b>	6
<b>Min</b>	1
<b>Standard Deviation</b>	1.43

Figure 23 - Emotional Discussion Response

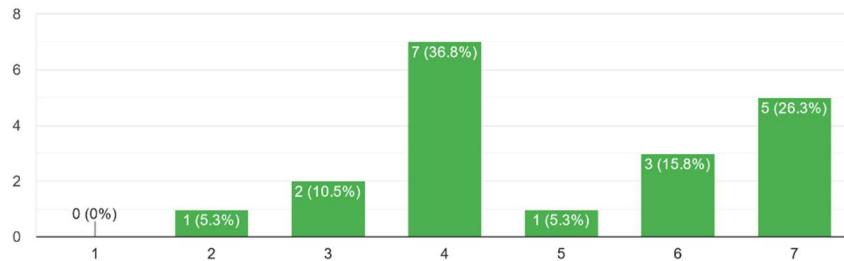
I would be comfortable discussing informational issues with other students via a chat support system  
19 responses



<b>Mean</b>	5.32
<b>Mode</b>	5
<b>Median</b>	6
<b>Max</b>	7
<b>Min</b>	2
<b>Standard Deviation</b>	1.56

Figure 24 - Informational Discussion Response

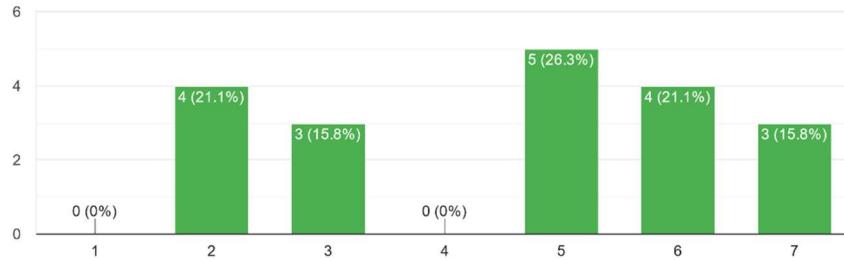
I would prefer to speak to a real human being rather than a bot, even if it could convincingly simulate human conversation and solve my issue to the same degree as the human  
19 responses



<b>Mean</b>	<b>4.95</b>
<b>Mode</b>	<b>4</b>
<b>Median</b>	<b>4</b>
<b>Max</b>	<b>7</b>
<b>Min</b>	<b>2</b>
<b>Standard Deviation</b>	<b>1.57</b>

Figure 25 - Human Opinion Response

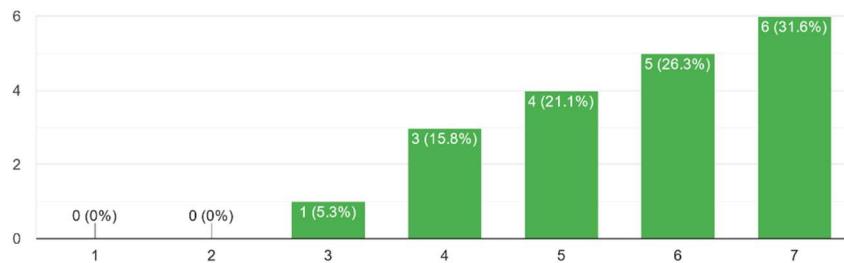
Having to wait for responses from support is not an issue for me  
19 responses



<b>Mean</b>	<b>4.58</b>
<b>Mode</b>	<b>5</b>
<b>Median</b>	<b>5</b>
<b>Max</b>	<b>7</b>
<b>Min</b>	<b>2</b>
<b>Standard Deviation</b>	<b>1.79</b>

Figure 26 - Response Time Opinion Response

I would prefer to stay anonymous when looking for support  
19 responses

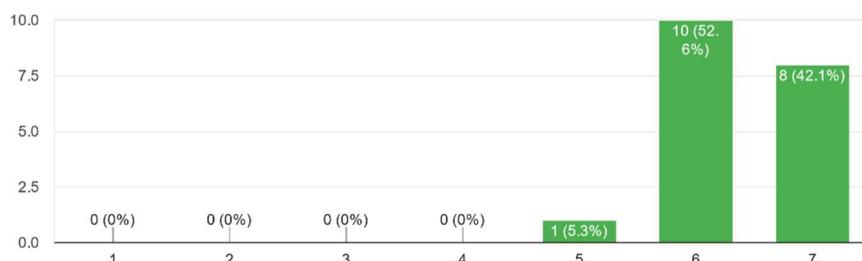


<b>Mean</b>	<b>5.63</b>
<b>Mode</b>	<b>7</b>
<b>Median</b>	<b>6</b>
<b>Max</b>	<b>7</b>
<b>Min</b>	<b>3</b>
<b>Standard Deviation</b>	<b>1.22</b>

Figure 27 - Anonymity Opinion Response

24/7 availability of support is important to me

19 responses

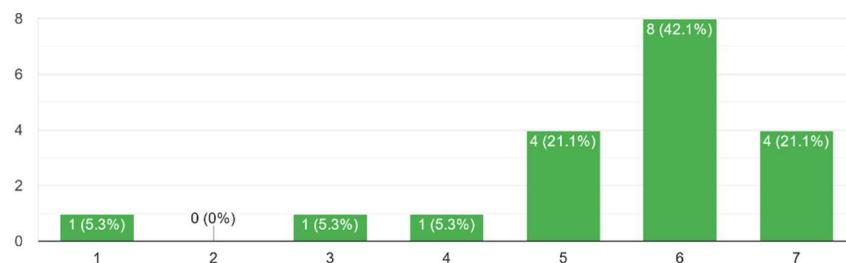


<b>Mean</b>	<b>6.37</b>
<b>Mode</b>	<b>6</b>
<b>Median</b>	<b>6</b>
<b>Max</b>	<b>7</b>
<b>Min</b>	<b>5</b>
<b>Standard Deviation</b>	<b>0.58</b>

Figure 28 - Availability Opinion Response

A chatbots ability to provide support would be more of a priority to me than its ability to convincingly simulate human speech

19 responses



<b>Mean</b>	<b>5.47</b>
<b>Mode</b>	<b>6</b>
<b>Median</b>	<b>6</b>
<b>Max</b>	<b>7</b>
<b>Min</b>	<b>1</b>
<b>Standard Deviation</b>	<b>1.46</b>

Figure 29 - Speech Simulation Opinion Response

In conclusion the survey helped support and decide certain elements of the project that were important to process. The agreement that categorisation fits the suggested informational and emotional distinction means that the current project can proceed with current aims as no issues have been raised here. Opinions regarding the importance of availability, privacy and comfort of discussion helped distinguish the need for automated chatbot systems like suggested. In terms of medium, the general opinion was that web implementation would be more effective to certain users, but messenger implementation would be effective to most users. As well, this implementation serves many suggested features. As such, current user choice leans to use of messenger implementation. The survey shown that informational support should be prioritised, and the quality of this support would be important. Based on the limited experiences of users of the current system, it is likely the system will be used for transition support (to undergrad or postgrad) and potentially should be focused upon. As well, access is an important factor for users but at this stage should not be prioritised. From the survey, anonymity seems like the most integral feature to implement. As an extension however, some features such as voice recognition and image processing will be considered.

## User Data Discussion

To conclude the findings of both the survey and interview, discussion will be combined, and each topic finding evaluated in this section.

After conducting user discussion, the justification for this project has been furthered. The interview found that automating the system would benefit both students and the university, avoiding wage costs and maintenance whilst creating a service that would not be limited to strict conditions of both time and content. The survey found similar requirements from potential users, with opinions strongly in support of anonymity and constant availability provided by the proposed solution. Other opinions demonstrated the importance of privacy and response time to some potential users and found that many did not use the current system, likely for these reasons. All surveyed also agreed to the distinction between informational and emotional support, showing that this concept that the project is built around is also justified.

The user discussion has allowed for a choice of access medium to be made. Both the interview and survey made it clear that independent software was not a viable choice for the project and gave strong reasoning to both alternatives. The interview found a concern of approval from the university itself, an issue that web integration would solve but could still be solved in alternative mediums. The interview did find that the current system uses a GUI similar to those provided by messenger applications, meaning that the implementation would not cause any new HCI issues despite limitations of GUI design. The survey found almost equal opinions of the two mediums with strong opinions in support of web integration but more in support of messenger applications. To be suitable for the most uses is a better outcome than catering to few however, showing support for the messenger integration. The survey also listed many accessibility features that would be simpler to include in the messenger integration as many are included already or partially. Overall, the contest between both the web integration and messenger integration is close yet the messenger integration is slightly superior, especially when development efficiency is considered. As well, use of messenger application would support the strong user need for mobile and website accessibility which would have to be explicitly developed for the other implementations yet not with this implementation.

Both the survey and discussion made some topics to focus on with the project clear. They both distinguished that informational support is more important due to the higher frequency of need and made clear that this aspect should take the focus of the chatbot. The survey also made clear that the average user would prefer better support than conversation simulation, which may explain the lack of chatbots aiming to provide realistic conversation via combination of emotional and informational support.

The features recommended by the survey and interview will be discussed briefly here, focusing on conflicts and choices to make based on conflicts here. Full feature discussion will be made during the user requirement section that details the requirements found. The biggest conflict is concerning anonymity. The interviewee discussed a feature of the current system where they can see the support user's past discussions and store a name. The surveyed responses gave strong support of anonymity. Whilst these two features can coexist with extra support, the interviewer's suggestion would likely cause issues for anonymity due to storing of conversations. Because of this, the feature will not be considered at current stages in support of anonymity instead due to strong user support and efficiency of development in comparison.

## Chapter 5 - Chatbot System Designs

### Use Case Diagram

This design will describe the different use cases of the artefact. This is intended to demonstrate the potential uses this program may have in the use of Edge Hill university. The use case diagram is intended to demonstrate the uses once complete and available as a service managed by the university itself and not in production of this design. Ambler (2008a) describes use case diagrams as an overview of usage requirements for the system. As such, this design will support and reinforce the requirements analysis. It will be based on the literature reviews, background study and user discussion and ensure that the aims and objectives of the model are met. Given that the project aims to provide a general functionality rather than approach issues specifically, this model is somewhat limited yet still provides useful information regarding the requirements. See figure 30 for the use case diagram.

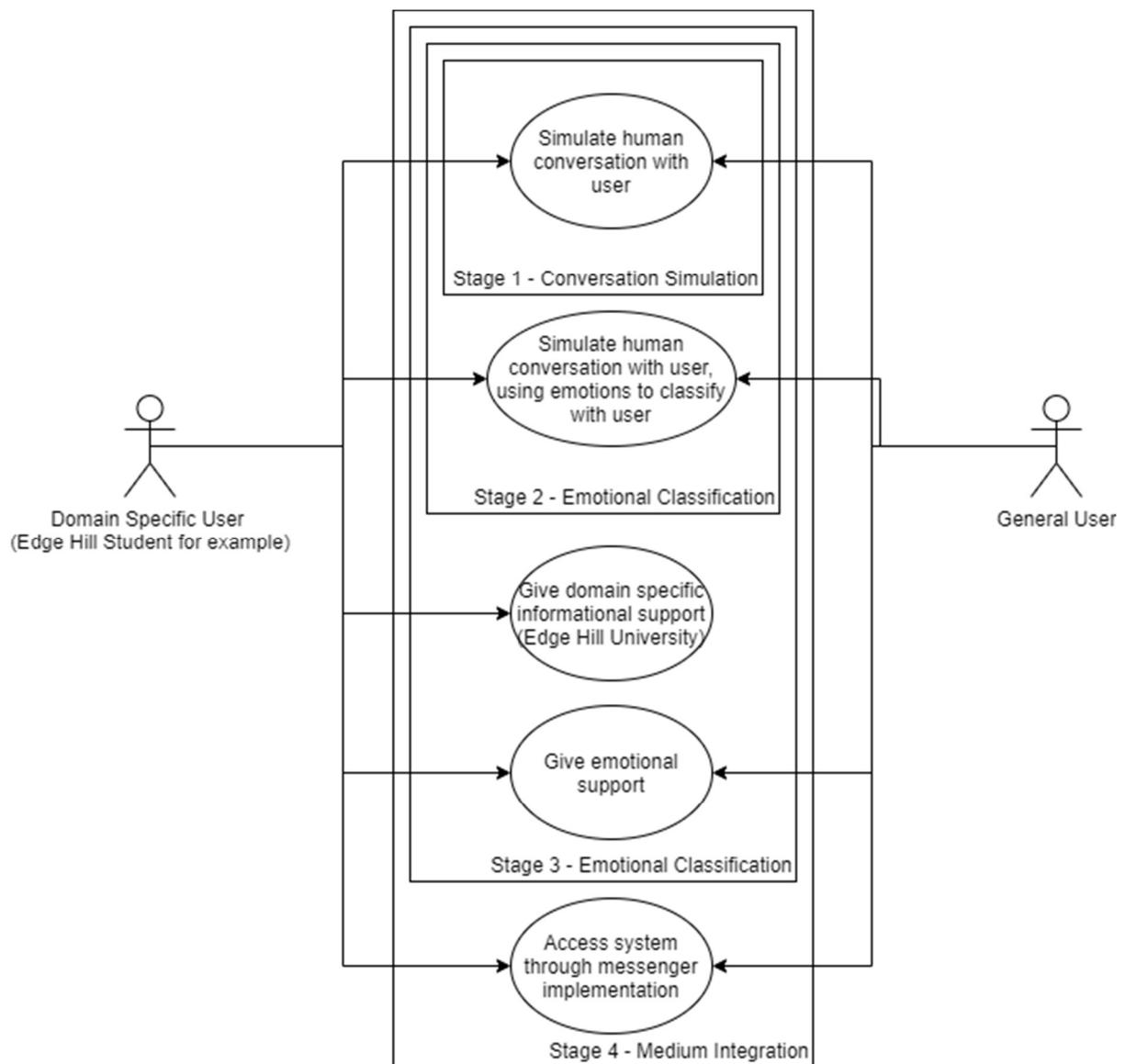


Figure 30 - Use Case Diagram

## Requirements Design

This section will list the found requirements. These are factors of the development that need to be met for the project to achieve its aims and objectives. Sources for these are the literature reviews, background study and user discussion. A graphical design has been made with the use case diagram to help decide these requirements. The requirements are split between the functional requirements, the features needed to achieve the aims and the non-functional requirements, the performance requirements for the software. These will be used to analyse if the development stages are a success or if they need more agile iterations to complete. See table 1 for the requirements.

<b>Functional Requirements</b>	<b>Stage Requirement will be Met</b>	<b>How it will be met</b>
<b>Seq2Seq classification of text to provide conversation simulation</b>	Stage 1 – Base Conversation Simulation	Development of a Seq2Seq model with general dataset
<b>Emotional Analysis additional feature to Seq2Seq model</b>	Stage 2 – Emotional Classification Implementation	Implementation of emotional classifications into encoder data and user input
<b>Domain Specific output in chatbot</b>	Stage 3 – Domain Specificity	Integration of knowledge base
<b>Edgehill University Domain Specificity</b>	Stage 3 – Domain Specificity	Use of developed domain specific data in developed knowledge base
<b>Emotional Support Component</b>	Stage 3 - Domain Specificity	Use of knowledge base to incorporate emotional support data for use with the emotional classification
<b>Accessibility through a messenger app to users</b>	Stage 4 – Medium Integration	Use of web application development and hosting to provide a messenger app implementation
<b>Non - Functional Requirements</b>	<b>Stage Requirement will be Met</b>	<b>How it will be met</b>
<b>No software errors</b>	All Stages	All stages pass function testing
<b>Sufficient Evaluation Metrics</b>	Stage 1 – 3	Model evaluation metrics are sufficient. Each stage will have its own boundaries defined to decide sufficiency.
<b>User Satisfaction</b>	All Stages	User Tested with finds the stage satisfactory for use

Table 1 - Requirements

During the requirements gathering process, some features and aspects were found that would be considered extensions to the current chatbot and non-essential to achieving the aim and objectives. Whilst they are not planned, they will be listed at this stage for future consideration and potential implementation if possible. See table 2 for a list of these features.

<b>Extension Feature</b>
<b>Constant Availability</b>
<b>Output of links in the conversation</b>
<b>Output of images in the conversation</b>
<b>External Recommendation for Complex Support</b>
<b>Phone and PC access</b>
<b>Basic Accessibility functions (E.g. Contrast settings)</b>
<b>Voice processing</b>
<b>Image processing</b>
<b>Referral to past conversations with anonymity considered.</b>
<b>Vibration inclusion</b>
<b>Text to speech</b>

Table 2 - Extension Features

## Back End Designs

This section will discuss the system designs for the backend implementation. For the dataset, the topics will be listed in a simple format. For the program itself, UML diagrams will be used. A class diagram will demonstrate the intended structure for the program and its implementation of object orientation. A use UML case diagram will also be included as well. A data flow diagram of the Seq2Seq implementation will also be made.

This section will detail the categories of topics the knowledge base dataset should consider. At this stage, they are only in an abstract format of category. During development they will be in a useable format.

- Campus Locations (directions to library for example or where to find certain things)
  - Potential map sent to user.
- Campus Services (description of services sent to user and instructions on use)
- Campus provision (description of what is available on campus such as if they ask the bot where they can find food, discuss the subway)
- Campus Life (give advice on living on campus, see the interview for example of this such as recommendations on where to drink)
- Pre-first year information (any information for applying and completing forms for moving in)
- Pre-post grad information (any information for applying and completing forms for moving onto post grad)
- Course specific information (such as course requirements, course leader details or summary of what courses are)

Emotional topics are not covered by the knowledge base, the emotional analysis system as well as the non-domain dataset will classify appropriate responses. Despite this, certain knowledge base data will be implemented demonstrate the ability of the chatbot's emotional component. This will be simple responses to inputs such as "I am sad" to support the classification as the general dataset would likely not provide emotional support.

### Class Diagram

This section will describe the object orientation modelling of the program through use of a class diagram. As Ambler (2008b) suggests, the class diagram will be a UML diagram intending to “show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes”. Class diagrams are an effective way of demonstrating the program structure of OOP programs and so will be used to design the project program. Note that the design is for the complete program yet many of the stage components can be removed or added from the design (such as the GUI class) without functionality loss allowing the single design to fit the entire process. To denote each stages implementation there is denotation for each stage components. Consider any numbers before the UML item the stage which it is added. See figure 31 for the class diagram design,

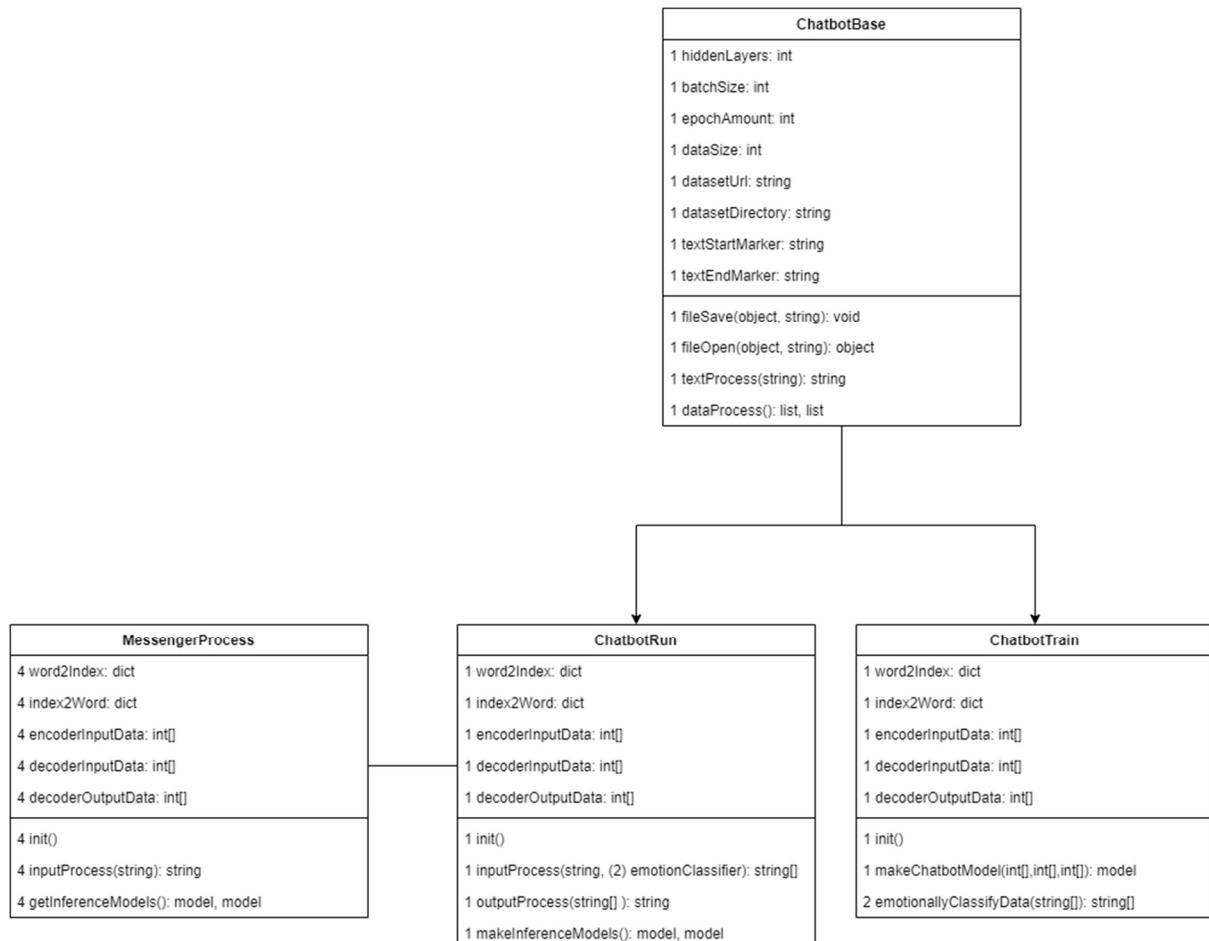


Figure 31 - Stage 1 Class Diagram

### Data Flow Diagram

Another UML diagram used to model the system is the Data Flow Diagram that, according to Ambler (2008a) "show the flow of data from external entities into the system, showed how the data moved from one process to another, as well as its logical storage". Representing this flow of data in deep learning methods is vital to ensuring an understanding of the process, making data flow diagrams an integral method of design for the proposed solution. Note that the diagram is a design of the final seq2seq use. In this case, this is only the additional feature of the emotional analysis and without, it would still function. See figure 32 for the data flow diagram.

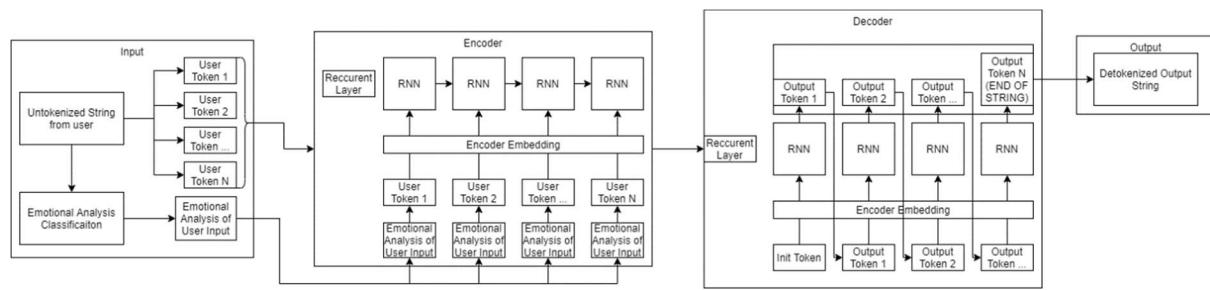
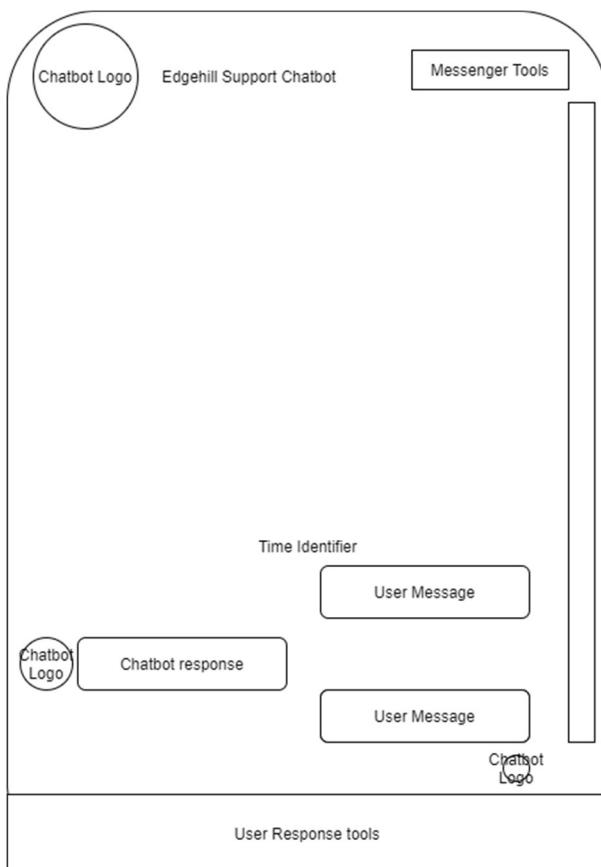


Figure 32 - Data Flow Diagram

## Front End Design

This section will be the design of the front end for the chatbot. As it uses messenger integration, this is severely limited. The chosen implementation is Facebook messenger due to its extensive support of chatbots with “300,000 developers building on the Messenger Platform” (Facebook for Business, 2019) as well as this being a medium widely preferred by users as found in a study by Facebook for Business (2016). Despite this, the designs for the system used are below.



Of this design, only the chatbot logo is controlled by the project. The entire GUI is heavily controlled by Facebook messenger. Note that Facebook messenger has near identical GUI in both web and mobile implementations.

Due to the limited control of GUI, included features messenger has that can be treated as if they are available to users will be listed.

- Sound and vibration response to messages
- Editable colour scheme
- Conversation search

As well as the features provided by Facebook messenger, certain features are available from the systems using the service itself.

- Contrast settings
- Text to speech and vice versa

Despite not being able to integrate into the system directly, the features exist for users and are therefore considered implemented.

## Chapter 6 - Implementation and Testing

Now development could begin. As a recap, Python has been chosen as the language with the package Keras (keras-team, 2020) to support development. The dataset Cornell movie dialogues (Danescu-Niculescu-Mizil and Lee, 2011) was originally chosen but the quality of use with this dataset alone proved poor. Instead a combination of a chatbot specific dataset, provided by Amin (2019) and the question answer dataset SQuAD (Rajpurkar et al, 2016).

Originally, the code was run locally using the Spyder IDE yet during the initial stages of the process, it was found that the hardware was insufficient and inappropriate early into development. Instead, the hosted solution Google Colab (Google, 2021) was used as it provides a suitable environment for processing data from Python and relevant packages (such as Keras) with easily available ram (approximately 25gb in the pro version) and various compatible processing hardware. It should be noted that the “Pro” version of Google Colab was used, giving even more ram to enhance the quality further.

### Testing Preface

This section will discuss the testing used for each development stage. There are three methods of testing used throughout:

- Function Testing – General testing where the functionalities of each stage are tested to see if they work without any errors. The quality of responses or the system is not tested here, only that they work. To do this, each function tested will be recorded in a table with inputs, the output and whether this was the expected output. Aspects tested will be things such as how the system performs when input is an incompatible datatype for example. This can be considered as black box testing. Black Box testing is “an analysis of the specification of the component to model its behaviour by state transitions” (British Computer Society, 2001). It only observes the input and output of the system and removes any complex implementation understanding.
- Evaluation Testing – The metrics gained from models trained in stages 1 – 3 are compared to a predefined boundary value. If they pass that boundary, the evaluation considers the table as successful and passes. If not, further tuning and evaluation will be made. These evaluation values have been discussed during the background.
- User Testing – A test user will be given each stages prototype. They will be asked to use the system as they would normally. Once done, they will fill out a survey of their experience. If the results of said survey are sufficient, this is considered successful. Similarly, to the evaluation testing, the results sufficiency boundary will be predefined and compared. In cases of the user testing, this is a simple yes or no question the user answers at each stage, specific to the addition. As with the interview, the user tested will be informed of their involvement and asked for consent following the full ethical standards. See appendix item 2 for the unsigned consent form to be used, item 12 for the information sheet for user testing and item 13 for their consent. This will apply for testing at all four stages. The discussion with the user to organise this testing is shown in appendix item 14. Each stages survey and respective results are shown in item 15 through to 19. The survey design will follow similarly to the user requirement survey with use of Likert scaling and other considerations. This will obtain objective opinions from the user to establish if the system is considered sufficient. Note that the definition of “sufficient” is dependent on the stage and if the chatbot shows that it is capable of providing the required features, not that it implements a high-performance version of that. The user has been informed of this. The final stage survey will examine if the chatbot is capable of use where high performance is needed. Each survey will ask if the user found the chatbot satisfactory and this will be used to establish if the user testing achieves the requirement. Once complete, the evaluations gathered regarding the user testing as they appear in this report will be sent to the user to verify and confirm consent for the report. They will be given the opportunity to change any answers and discuss the projects’ findings.

## Stage 1 - Back End Base Conversation Simulation

At this stage, development is limited to creation of simple conversation simulation with no emotional analysis or domain specificity. A Seq2Seq model based on the designs without elements for sentiment analysis and domain specificity will be used.

### Development Choices

As discussed earlier, certain parameters must be chosen for the training process as well as some implementations of layers such as the dropout. The SoftMax function will be used as the activation function due to the previously discussed suitability for categorical data and use with the categorical cross entropy loss. The number of hidden layers used found from the simpler of the two equations (to avoid misuse of the scaling factor) and produced the value 232 (Note that 223 and 132 are values found as the max number of tokens in input and output of the training data). This will be used as the number of hidden layers for the task. The batch size used will be 64 due to its performance in Shen's report (2018). Three hundred epochs will train the data due to its larger amount but not enough that the time taken to train becomes obtrusive. As discussed earlier, RMSProp will be used to manage the learning rate. As discussed, dropout will be implemented. This will be done at each LSTM layer with a dropout of 0.5 as recommended by Srivastava et al (2014) for dropout in internal layers.

One choice is specific to the data itself rather than the model. Of the datasets used, the Cornell Movie Dialogues (Danescu-Niculescu-Mizil and Lee, 2011) is too large to use the entire set. After testing, the largest amount of the data set that could be used by Google Colab's memory given, it was found that 10,000 items could be used (note that with the other dataset, the items amount is larger than this). After reconsidering the data, a data size parameter still had to be decided for the Squad dataset (Rajpurkar et al, 2016). Instead of performing a slice of only the starting data like with the initial use of the movie datasets, the format of the data was used to gain a variety of data by selecting a selection of 3 paragraphs of questions from each category. This assured data variance and provided a limit to the memory used at the same time. This was also tested to use the limits of memory in Google Colab.

### Program Structure

The program was structured as three classes, per the design. The structure will be described, and the function of the code explained fully in this section. The code is available in appendix item 20. See figure 33 for a model of the Seq2Seq structure used for this stage. See figure 34 for a summary of the training model, figure 35 for a summary of the encoder interpretation model and figure 36 for a summary of the decoder interpretation model.

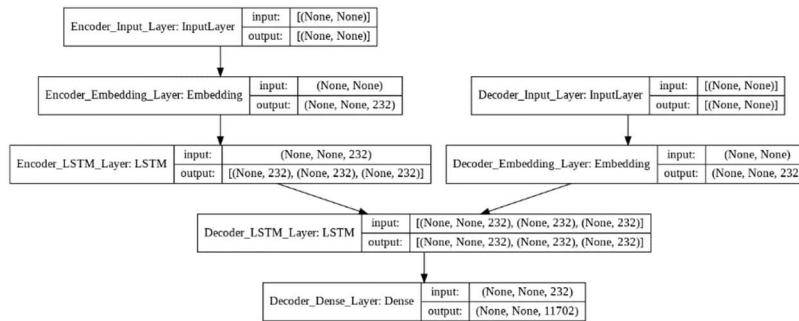


Figure 33 - Seq2Seq Structure

Model: "Chatbot_Model"				
Layer (type)	Output Shape	Param #	Connected to	
Encoder_Input_Layer (InputLayer)	[(None, None)]	0		
Decoder_Input_Layer (InputLayer)	[(None, None)]	0		
Encoder_EMBEDDING_LAYER (Embedding)	(None, None, 232)	2714864	Encoder_Input_Layer[0][0]	
Decoder_EMBEDDING_LAYER (Embedding)	(None, None, 232)	2714864	Decoder_Input_Layer[0][0]	
Encoder_LSTM_LAYER (LSTM)	[(None, 232), (None, 232), (None, 232)]	431520	Encoder_EMBEDDING_LAYER[0][0]	
Decoder_LSTM_LAYER (LSTM)	[(None, None, 232), (None, 232), (None, 232)]	431520	Decoder_EMBEDDING_LAYER[0][0] Encoder_LSTM_LAYER[0][1] Encoder_LSTM_LAYER[0][2]	
Decoder_Dense_Layer (Dense)	(None, None, 11702)	2726566	Decoder_LSTM_LAYER[0][0]	
Total params: 9,019,334				
Trainable params: 9,019,334				
Non-trainable params: 0				

Figure 34 - Training Model Summary

Model: "Inference_Encoder_Model"				
Layer (type)	Output Shape	Param #	Connected to	
Encoder_Input_Layer (InputLayer)	[(None, None)]	0		
Encoder_EMBEDDING_LAYER (Embedding)	(None, None, 232)	2714864		
Encoder_LSTM_LAYER (LSTM)	[(None, 232), (None, 232), (None, 232)]	431520		
Total params: 3,146,384				
Trainable params: 3,146,384				
Non-trainable params: 0				

Figure 35 - Encoder Inference Model

Model: "Inference_Decoder_Model"				
Layer (type)	Output Shape	Param #	Connected to	
Decoder_Input_Layer (InputLayer)	[(None, None)]	0		
Decoder_EMBEDDING_LAYER (Embedding)	(None, None, 232)	2714864	Decoder_Input_Layer[0][0]	
State_H_Input_Layer (InputLayer)	[(None, 232)]	0		
State_C_Input_Layer (InputLayer)	[(None, 232)]	0		
Decoder_LSTM_LAYER (LSTM)	[(None, None, 232), (None, 232), (None, 232)]	431520	Decoder_EMBEDDING_LAYER[1][0] State_H_Input_Layer[0][0] State_C_Input_Layer[0][0]	
Decoder_Dense_Layer (Dense)	(None, None, 11702)	2726566	Decoder_LSTM_LAYER[1][0]	
Total params: 5,872,950				
Trainable params: 5,872,950				
Non-trainable params: 0				

Figure 36 - Decoder Inference Model

*Implementations*

Initially the code had to import relevant packages. There were two sections to this, packages for handling the data and packages for Keras itself. The data handling packages were mostly parts of python designed to handle specific data needs. Numpy was imported to handle larger collections of numeric data. The Keras package imported handled the model formation and training. Matplotlib was also imported to create graphs based on the data itself.

*Base Class*

The first class acted as a base to the other two. It could be considered abstract as it is never instanced intentionally in the code. Both other classes derive from this class, taking its members as their own.

The class declares numerous fields for use in the derived classes. The initial grouping holds the different parameters chosen earlier as their respective datatypes (int). The second grouping holds the file directories for the separate datasets as strings. The third grouping holds strings for both the end of sentence marker and the start of sentence marker.

Note that as this class is never instanced, it has no constructor.

The first functions of the class handle the different file handlers used. Both pickle and HDF5 are used as pickle is a very suitable file handler yet HDF5 is necessary for larger data, specifically the one hot encoding later on that caused issues using pickle. These functions take items and their file names as parameters and depending on what is called, either save them or open them to the specified data directory.

The next function defines the text cleaning process which takes a text input and processes. It initially assures that the text is not a dictionary, a common issue for some data when : is used. It then makes the text lower case and uses regular expressions to swap potential punctuation (such as I am) to non-punctuation text (in the example given, I am). If there are any conflicting characters remaining, they are then filtered from the text. The text is then returned.

The next function processes the text from the specific dataset. This dataset uses .yml files for its format so the package yaml is required to load this data. Once loaded, the conversations supplied are taken. The query of said conversations (the initial phrase) is cleaned and then returned as an input for the encoder. The responses were more complex to handle. As there could be multiple responses to a single query, they were examined to see if there were multiple. If not, the single response was cleaned and added to a list. If there were multiple responses, each would be added to a single string and then added to the list, assuring a one-to-one relationship between queries and responses. This list was then processed to add the markers for beginning and end of the text and verified to assure they were strings. Once done, this was also returned as the decoder inputs.

The next function processes text from the general dataset. This opens the Json file for the dataset and progresses through the categories for the data, iterating through each. For paragraphs, the iteration is limited to three paragraphs for each topic. This provides variety in the dataset, varying the topics for gaining data from and limits the data size to become useable at the same time. Each question and answer is taken and added to a list on the condition both exists, cleaning the text simultaneously. Each response is then tagged with the markers after doing this.

Note that each derived class handles these inputs individually, the implementation for doing so is class specific.

*Training Class*

The training class is used to handle the training process of the chatbot. It aims to train the chatbot and create three models, one for the dataset training, one for interpreting inputs and another for interpreting the embeddings produced from said inputs. These models are saved and used in the processing class later. This class inherits from the ChatbotBase.

The class begins by declaring empty dictionaries and lists to hold the data produced from processing the encoderInputs and decoderInputs of the base class.

It then defines its own function to handle said inputs. This works by appending each encoderInput and each decoderInput together in separate specific lists after calling the base classes implementation of the dataset processing. Once done, these lists are used to establish a tokenizer and word index of every word in the datasets. A reversed index is also made for use later. The tokenizer then processes the text of the encoderInputs and decoderInputs into sequences, arrays where each text is converted to its index value. These sequences are then padded to their max length to assure a consistent sizing. The decoderInput sequences are used to create decoderOutputSequences which are intended to use as the correct label for the learning process. Because of this, the data needs to be one hot encoded at this stage to match the model's own process. The embedding process finds each words weighting based on its vector. It produces a list of the entire lexicon where each item is a weighting. The higher, the more likely to be predicted. One hot encoding is a data format where each line of text to a collection of lists. Each list represents a single word as a series of 0's with the relevant word's index being 1 to represent that it is that word. By comparing this with the embedding weight, the accuracy and loss can be found. Note that OHE takes large amounts of data and is the source of the mentioned hardware problems. The to\_categorical function is used for this. This is converted to the byte data size, the smallest datatype available for this format to reduce hardware instability.

The classes' constructor is then defined. This acts as a function to be called on instancing. It begins by attempting to load any previously made data if it exists. If it does not, it then tries to load the dataset's file if that exists. If that does not, the dataset is downloaded directly. Once done, the data process function is called and the relevant data from that is returned for use. This is also saved for future use. The model for training is then defined with encoder and decoder layers using the predefined parameters. The encoder model begins with an input layer to handle the input data. It is then passed to an embedding layer, that produces the data as an embedding vertex and passes this data to the LSTM that as previously discussed, obtains the state of the data to pass to the decoder. The decoder begins with an input layer itself, taking the defined decoder inputs. These are passed to the embedding layer for the same purpose as with the encoder. This is then used with an LSTM using both this embedded input and the states from the encoder to find the decoder outputs, weightings of each potential word. This is passed to a dense SoftMax layer to produce the data. This model is then used to train the data itself providing the weightings needed. These are saved for future use as well as saving the training process history for evaluation. During training, the loss and accuracy are gathered from its predictions as well as validation data using a validation split of 80/20.

### *Chatbot Running Class*

The class to run the chatbot itself functions as a derivation of the base. It intends to obtain the data needed and then create interpretation models for the existing trained model and use these interpretations with user input.

The class begins by declaring empty dictionaries and lists to hold the data produced from processing the encoderInputs and decoderInputs of the base class. Unlike the training class, no list is defined for the output data.

The first function handles the encoder and decoder inputs. This works by appending each encoderInput and each decoderInput together in separate specific lists after calling the base classes implementation of the dataset processing. Once done, these lists are used to establish a tokenizer and word index of every word in the datasets. A reversed index is also made for use later. The tokenizer then processes the text of the encoderInputs and decoderInputs into sequences, arrays where each text is converted to its index value. These sequences are then padded to their max length to assure a consistent sizing. The processing of the output data is excluded from this implementation due to the complexity and its lack of necessity. The indexes and sequences are then returned.

The second function handles pre-processing user input. It calls the cleaning function from the base as well as splitting the text into tokens that are verified for existence in the word index. If it does not exist, it is ignored.

The second makes the inference models. It imports the existing trained model and extracts the data and layers needed. The models for inferring are then constructed. The encoder model encodes data into the states directly. Then the decoder model takes said states as inputs along with the training decoder inputs and uses them as an initial step of its LSTM with use of the original embeddings found via training. This creates both outputs and states. The outputs are converted to weightings via a dense layer and the states are passed for use again for the next word in this sequence, effectively storing the data of this token.

The classes' constructor is then defined. This acts as a function to be called on instancing. It begins by attempting to load any previously made data if it exists. If it does not, it then tries to load the dataset's file if that exists. If that does not, the dataset is downloaded directly. Once done, the data process function is called and the relevant data from that is returned for use. This is also saved for future use. The input loop and interpretation follow this. Initially, the encoder model is used with processed user input to find the text's state. An empty array is made to hold the beginning of text marker and continues to hold previous words that are predicted in the iteration for this input. A nested loop is then called that predicts the weightings based on the input states. The weightings then find the predicted word via use of the argmax function, that returns the index of the greatest item in a list. This index is used with the reversed word index to find the word it predicts. This word is then verified and processed, adding to a string for printing and the array of previous words, overwriting the last entry. When the end of text marker is predicted, the nested loop ends, and the output string is printed. The full loop iterates for the next user input, keeping a flow for user inputs.

### *Non - Class Script*

Outside of the classes, a simple process is run that examines if a saved model exists. If it does, training must be completed and so the training model is instanced. If it does not, only the chatbot use class is instanced.

### *Evaluation Script*

A small, independent script is also ran with the history of the training model to obtain graphs of the loss and accuracy with the epochs for evaluation purposes.

## Development Discussion

This section will discuss the development process for the stage rather than the code itself.

### *Hardware Issues*

In the initial cycle of the agile process for this stage, hardware issues were encountered. This led to further investigation into what device the program should be developed using and the finding that Google Colab would be capable of providing sufficient hardware. Despite Dettmers (2018) claim that the hardware would be sufficient (GPU with  $\geq$  8GB VRAM, RAM  $\geq$  VRAM size, CPU with at least 2 cores per GPU and at least 3.5Ghz with the proposed development hardware being a PC with AMD RX480 GPU (8gb VRAM), 8gb RAM and AMD FX8320 CPU (8 Cores, 4ghz)) it was found at this stage that it was not. The memory available was insufficient, especially when one hot encoding the data and limited the potential batch size and dataset use. The GPU was unavailable for use as Keras does not natively support the card (due to being AMD). AMD processing implementations proved unusable as the implementations available caused issues with memory (PlaidML, 2021) or were not available for the specific card (ROCM Core Technology, 2021). The CPU itself was slow at processing the training, taking around 10 hours to complete even with reduced epochs. To circumvent these issues, alternatives were investigated the hosted solution Google Colab (Google, 2021) was chosen as it provides a suitable environment for processing data from Python and relevant packages (such as Keras) with easily available ram and various compatible processing hardware from server hosted processes. Google Colab Pro was used to provide the most ram possible giving around 25gb of ram and access to cards such as the Tesla V100 GPU or T4 GPU when available with both outperforming Dettmers (2018) recommendations massively (Kenyon, 2019).

### *Dataset Issues*

In the second cycle of the agile process for this stage, another issue was found. This was the quality of dataset in use. Despite earlier investigation into reports such as Nguyen's (2019) finding the movie dialogues dataset (Danescu-Niculescu-Mizil and Lee, 2011) to be suitable due to quantity and extensive use in the field, the chatbot demonstrated its lack of quality for the projects purpose. Given the nature of the dataset, responses were generated in movie contexts that were unsuitable for a support chatbot as they referred to info regarding the movies that disrupted the aim to simulate actual conversation. The data was trained with responses pertaining to contexts that would not make sense for a chatbot such as referring to plot elements that had not been mentioned. As well, the simulation used profanity and language unsuitable for a support based chatbot. As an example, the input "I don't understand" produced the output "hal maintain normal eva condition", a direct quote from "2001: A Space Odyssey" (Kubrick, 1965). The implementation code to produce this is in appendix item 19 though it should be noted that this code is unfinished due to the changes made and therefore unstructured.

To circumvent this issue, further investigation for a more appropriate dataset was conducted. Many dialogue corpuses are specific to a single domain or unsuitable like the movie dialogue corpus. Harvesting extensive data of conversation is difficult due to privacy concerns that inhibit the ability. In many cases, readily available corpuses that accomplish this are from services that are readily accessible such as movie scripts.

In Nguyen's (2019) report the ConvAI3 (2018) dataset was also recommended which takes volunteer user input and responses from its own chatbot. Despite this being suitable due to a lack of domain focus and conversational format, the quality of the data is poor due to being a chatbot. Simulating a simulation would ensure this project performs poorly so despite the suitability, this cannot be used.

It became clear that a single extensive dataset that provided the quality close to what was necessary would not be available. Instead smaller yet directed datasets would be considered. As dataset size increases, so does the inability to

focus on a specific task such as providing suitable data for conversation simulation. On the other hand, manufactured datasets are restricted to smaller sizes that provide a rigid function. This is still not as rigid as using a rules based chatbot but rigid, nonetheless. A dataset for chatbots provided by Amin (2019) will be used due to recommendation from Shchneider (2020). This use would likely impact the implementation of domain specificity due to the structure of the dataset. It functions as a collection of categories of topics for the chatbot with questions and responses. Adding a category for the domain may prove to implement the domain specificity.

To attempt to remove the rigid nature of the reduced dataset use, a larger general dataset would be conjoined for use. It should be considered that this general dataset should not overpower the existing one yet should provide a wide enough source of data for the chatbot to function outside of the reduced datasets implementations. The Cornell Dialogues Set would conflict due to its own implementations of similar text for example and so an alternative data set should be found. For this, a question-and-answer structure would be suitable. The dataset would be limited to this structure so general conversation provided by the original reduced set would not be overpowered yet suitable amounts of data would still be included, expanding the existing lexicon of the chatbot. Use of a question answer set would also support the aim to provide informational support on a general scope.

Another investigative report (Oriel, 2020) recommended the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al, 2016). It is a dataset made up of questions and answers based on Wikipedia entries. The data contains 100,000 questions and will be a suitable model for conjoining to the existing specific dataset. It should provide a good understanding of questions and answers for the chatbot to model itself off.

After conjoining these two datasets, the chatbot produce more appropriate responses yet quality was still lacking, likely due to not using a suitable dataset for the purpose of the project. Given that there are no extensive datasets that provide this, and this conjoined solution is the closest available option, this would have to be left as is. This stage made it very clear that the choice and availability of datasets have a massive impact on classification.

## Testing

### *Functionality Testing*

Functionality testing assures that each component of the program works without error and handles potential errors caused by input into the system.

The testing would use boundary cases of data for input into the chatbot. This testing does not analyse the quality of the response, only that a response was made, and the system reacts in the intended way. At this stage, the functionality testing is minimal as functionality is limited to simple text input and output for testing the chatbot performance and not for user use. As the current stage does not intend for actual use (no front end implemented) this will only test that the classification process handles the potential text. Issues with the input medium are not tested, such as if an input is too large. This will be handled in the front-end stage.

See table 3 for a record of the tests conducted.

Test	Input	Output	Success
<b>Input of string</b>	Single String ("hello")	Response ("hi")	Yes
<b>Input of second string and more</b>	Multiple strings ("whats up", "definitely", "huh")	Response for each input ("and other", "the science that deals with the european and windows", "the science that deals with the very order of ai")	Yes
<b>Input of string with punctuation</b>	String with punctuation ("hey who are you?")	Response ("i am just an artificial intelligence")	Yes
<b>Input of numbers</b>	Integer ("42422")	Default Response ("the science that deals with the very order of ai")	Yes
<b>Empty input</b>	Nothing ("")	Default response ("the science that deals with the very order of ai")	Yes
<b>Unrecognisable String</b>	Unrecognisable string ("dgfsgdgsdg")	Default response ("the science that deals with the very order of ai")	Yes

Table 3 - Stage 1 Functionality Testing

As the table shows, stage 1 functioned correctly. The inputs were all handled correctly without error and the requirement for no software errors was therefore met.

### Evaluation Testing

Evaluation testing will examine the previously discussed metrics and define a boundary value for each stage to pass, ensuring that the metric performance for each stage is sufficient. Validation data is also gathered for analysis but not for testing as discussed earlier, user testing will be used to find if the chatbot produces appropriate chatbots. A validation split of 80:20 will be used as recommended by Desmerais (2018).

For stage 1, the loss boundary will be 0.005. This would represent a very close proximity for the results and would demonstrate success. The accuracy boundary will be 0.99. This would represent very accurate predictions for the results and also demonstrate success. Stage 1's metrics are plotted as graphs in figure 37 for loss and figure 38 for accuracy. The graph uses each epoch as its x axis and the metric as its y and is recorded as a history of the training process.

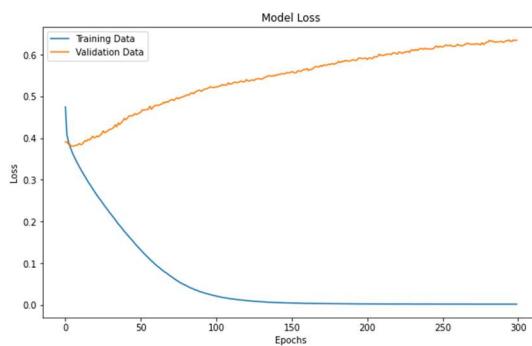


Figure 37 - Stage 1 Loss Plot

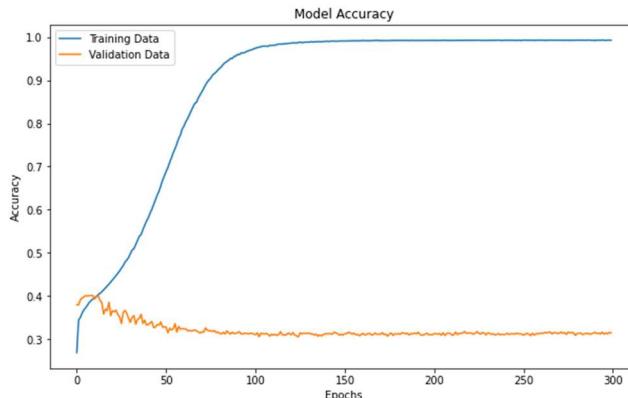


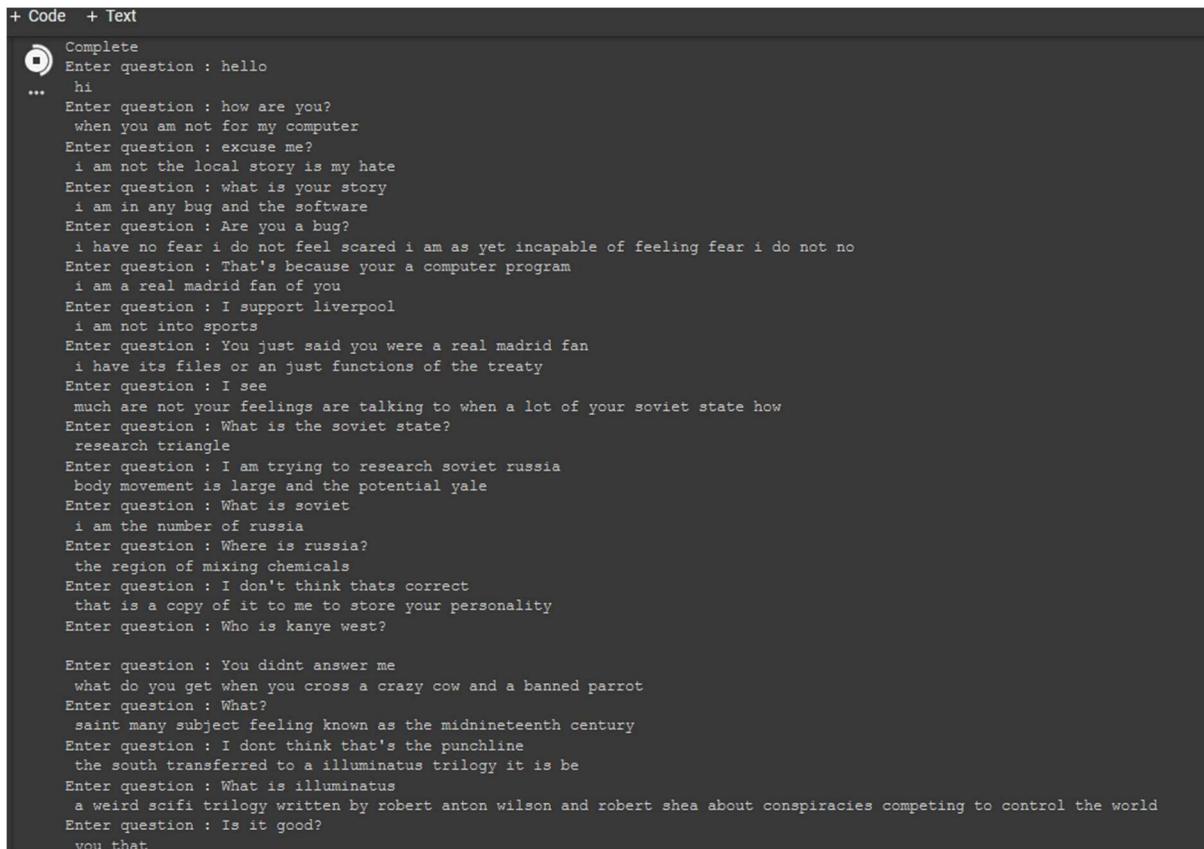
Figure 38 - Stage 1 Accuracy Plot

As shown in the graphs, stage 1 had a loss of 0.0016 and accuracy: 0.9939 with validation loss at 0.6346 and validation accuracy at 0.3140 at the final result. The decrease in validation metrics show overfitting, despite the dropout to handle this. The general loss and accuracy did meet their respective testing boundaries and so, this stage could be considered a success in evaluation metrics, despite the overfitting. The graphs show a specific pattern to the process. Figure 37 shows a curved decline in loss until around 150th epoch where it plateaus. It also shows that model validation loss increases almost linear. This demonstrates that 300 epochs are unnecessary yet will continue to be used due to the earlier choice and potential changes later in stages. Model accuracy shows a similar trend, increased massively at the start then slowing to a slope and plateauing around 150 epochs, reinforcing how 300 epochs was unnecessary. The validation accuracy decreases steadily until plateauing at a similar point. Given that the validation data lack of performance was poor before reaching the plateau point, it can be assumed that over-epoch use was not the cause. The model seems to be overfit from the beginning, potentially supporting the claim that it is a symptom of the dataset choice.

*User Testing*

User testing exposed a third-party representative of the potential user (an edge hill student) to the current stage. They would initially use the current stage prototype then answer a specific questionnaire regarding their use, gathering external user feedback at each stage to see if the current prototype is acceptable as a proof of concept for chatbot use.

See figure 39 for the record of the user's test.



```
+ Code + Text
Complete
Enter question : hello
hi
...
Enter question : how are you?
when you am not for my computer
Enter question : excuse me?
i am not the local story is my hate
Enter question : what is your story
i am in any bug and the software
Enter question : Are you a bug?
i have no fear i do not feel scared i am as yet incapable of feeling fear i do not no
Enter question : That's because your a computer program
i am a real madrid fan of you
Enter question : I support liverpool
i am not into sports
Enter question : You just said you were a real madrid fan
i have its files or an just functions of the treaty
Enter question : I see
much are not your feelings are talking to when a lot of your soviet state how
Enter question : What is the soviet state?
research triangle
Enter question : I am trying to research soviet russia
body movement is large and the potential yale
Enter question : What is soviet
i am the number of russia
Enter question : Where is russia?
the region of mixing chemicals
Enter question : I don't think thats correct
that is a copy of it to me to store your personality
Enter question : Who is kanye west?

Enter question : You didnt answer me
what do you get when you cross a crazy cow and a banned parrot
Enter question : What?
saint many subject feeling known as the midnineteenth century
Enter question : I dont think that's the punchline
the south transferred to a illuminatus trilogy it is be
Enter question : What is illuminatus
a weird scifi trilogy written by robert anton wilson and robert shea about conspiracies competing to control the world
Enter question : Is it good?
you that
```

*Figure 39 - User discussion*

The user then answered a questionnaire based on the findings of the current stage (see appendix item 14). The results found that the user did not find the chatbot satisfactory. They also did not encounter any errors. They gave a scoring of 4 when asked if it was convincing and summarised by saying “The chatbot demonstrated capability of understanding what I was saying. However, the responses it formulated were mostly nonsense in response to its understanding.”

## Conclusion

In conclusion, this stage proved to be the most complex. It involved creating a strong foundation base for the next stages.

The choices of parameter were made easily based on previous research. This section was well prepared and easily justified each choice made. The structure of the program was extensive and provided a well-balanced framework for later use. The discussion reflected on findings during the stage development, particularly the hardware issues and the dataset issues.

These were resolved during the initial iterations of development.

The function testing found that the chatbot worked without errors on all of the boundary cases tested. The limited testing shown that the stage is currently functional, and that the requirement had been met for this stage. The evaluation testing shown that the stage meets all evaluation metrics that are defined for requirements (accuracy and loss) and does so effectively. It did find that the validation set performs poorly, likely due to overfitting. This was not a requirement however and so the evaluation testing requirement is met for this stage. User testing found mediocre results with the user finding that the chatbot seems to understand what they say but lacks proper responses. This failed at this stage, but as later stages may rectify it, the development will continue. It should be noted that this could be due to many reasons. The most notable would likely be dataset choice but as the discussion found that the current dataset is likely the most effective available, this cannot be rectified easily. It could also be due to overfitting despite the dropout use. Potentially, further anti - overfitting mechanisms should be considered. At this stage however, it will not be. The user testing requirement was not met at this stage but given the reasonings analysed, the development will continue under expectation that later stages will fix this.

Overall the stage was a success despite poor user testing. The requirement of "Seq2Seq classification of text to provide conversation simulation" has been met and all bar one testing requirement found success. Even then, the results of the user testing were mediocre and not a complete failure.

## Stage 2 - Emotional Analysis Integration

At this stage, the emotional analysis component is integrated into the existing conversation simulation as an additional feature. This has been accomplished by emotionally analysing input sequence and appending the emotional metrics to the query sequence. The learning process would then associate these scores with the response data and associate the emotional analysis to the data's responses to the inputs. In theory, these responses should be appropriate to those emotions exhibited by the query and establish emotional reasoning for inputs outside of the training data. To accomplish this, the text2Emotion (Sharma, 2020) will be used to classify text into emotional categories. This classification will then be appended to the input and classified base on this.

### Development Choices

Parameters selected for the conversation simulation are retained. There are no specific parameter choices for this stage, only the class choice which has been discussed.

### Program Structure

This stage builds upon the previous ones and so uses the same structure. Any additions or changes will be discussed here. The code additions are available in item 23 of the appendix. The previous model and summaries can be applied to this structure as well, due to identical structure. See the previous stage for these.

#### *Emotion Handling*

The first major addition was the emotion handling for encoder data. For each classes processing function, a block was added that iterated through each encoder input and found the emotional classification. This was then processed into a Numpy array and added to a Numpy array for the entire encoder input set with relative indexes.

#### *Addition to Training Data*

This array was then concatenated with the encoder sequences before the padding for use in the training process, adding the scores to the end of the inputs. These emotional inputs are then therefore associated with the encoder data and the decoder output data, classifying the use of emotional inputs to direct the output of a chatbot and acting as an additional feature.

#### *Emotion Handling of Input Data*

Once trained with this model, the inputs would have to be similarly handled. The “TextToTokens” function would be edited to include functionality to take the user input and find the emotional scores then process it into a sequence and append it to said score. Essentially the same process for the encoder input but modified for singular inputs. This could then be classified into the correct output, using the emotional analysis in this process.

### Development Discussion

This section will discuss the development process for the stage rather than the code itself.

This stage only required a single iteration of the agile process. The use of the text2emotion class was efficient in terms of development complexity yet the lack of control of the package meant that analysis of efficiency in running was not an option. Any potential method to reduce the functionalities run time were unavailable. The package used a lexicon based emotional analysis implementation using NLTK (2009) and other components. This meant that the package didn't utilis the full potential of the hardware available and cause it to run somewhat slowly. As well, the lack of control of the package meant that controlling the implementation was not possible. Despite these issues, the package still performed emotional analysis efficiently and somewhat effectively and so was a suitable choice.

## Testing

### *Functionality Testing*

Functionality testing assures that each component of the program works without error and handles potential errors caused by input into the system.

As with the previous stage, testing would use boundary cases of data for input into the chatbot. This testing does not analyse the quality of the response, only that a response was made, and the system reacts in the intended way. At this stage, the functionality testing is minimal as functionality is limited to simple text input and output. As the current stage does not intend for actual use (no front end implemented) this will only test that the classification process handles the potential text. Issues with the input medium are not tested, such as if an input is too large. This will be handled in the front-end stage.

See table 4 for a record of the tests conducted.

Test	Input	Output	Success
<b>Input of string</b>	Single String ("hello")	Response ("hi")	Yes
<b>Input of second string and more</b>	Multiple strings ("how are you?", "excuse me", "huh")	Response for each input ("i am not well into sports that is what are not well", "i think the nic on the local firewall is going bad it drops a lot of packets and makes it hard to get anything done i hear it resets the link a lot", "the gre")	Yes
<b>Input of string with punctuation</b>	String with punctuation ("don't tell me what to do")	Response ("do you think when i really on a good power")	Yes
<b>Input of numbers</b>	Integer ("5325")	Default Response ("the gre")	Yes
<b>Empty input</b>	Nothing ("")	Default response ("the gre")	Yes
<b>Unrecognisable String</b>	Unrecognisable string ("dgsfdsdsf")	Default response ("the science that deals with the very order of ai")	Yes

Table 4 - Stage 2 Functionality Testing

As the table shows, stage 2 functioned correctly and the requirement for doing so has therefore been met.

### Evaluation Testing

Evaluation testing will examine the previously discussed metrics and define a boundary value for each stage to pass, ensuring that the metric performance for each stage is sufficient.

For stage 2, the loss boundary will be 0.005. This is the same as the previous stage as the addition should not cause a decrease in loss if implemented correctly. Similarly, the accuracy boundary will be 0.99. Stage 2's metrics are plotted as graphs in figure 40 for accuracy and figure 41 for loss. The graph uses each epoch as its x axis and the metric as its y and is recorded as a history of the training process.

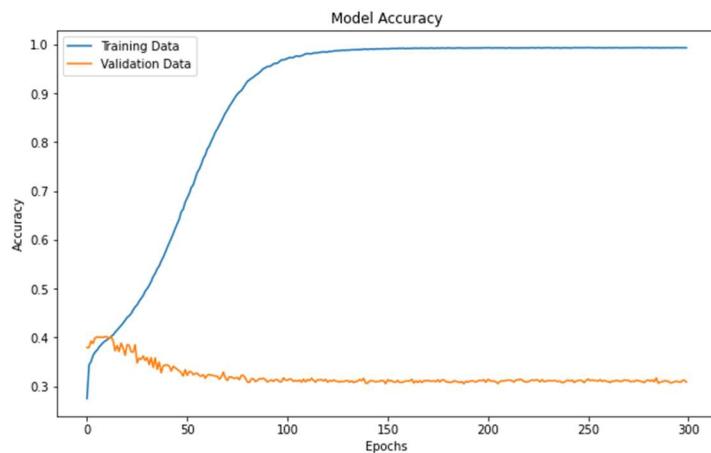


Figure 40 - Stage 2 Accuracy

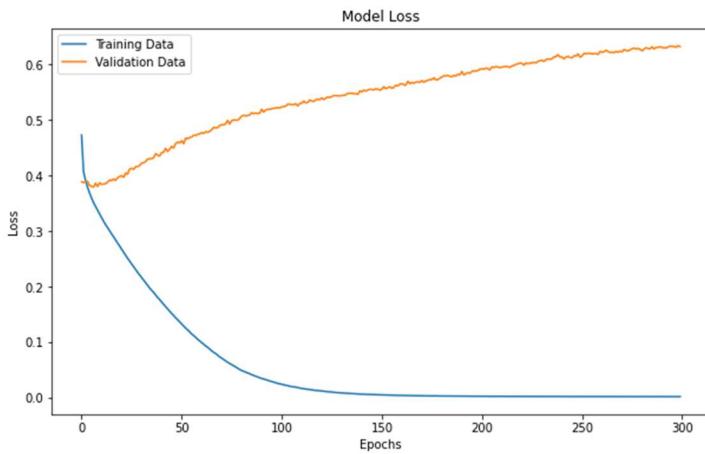


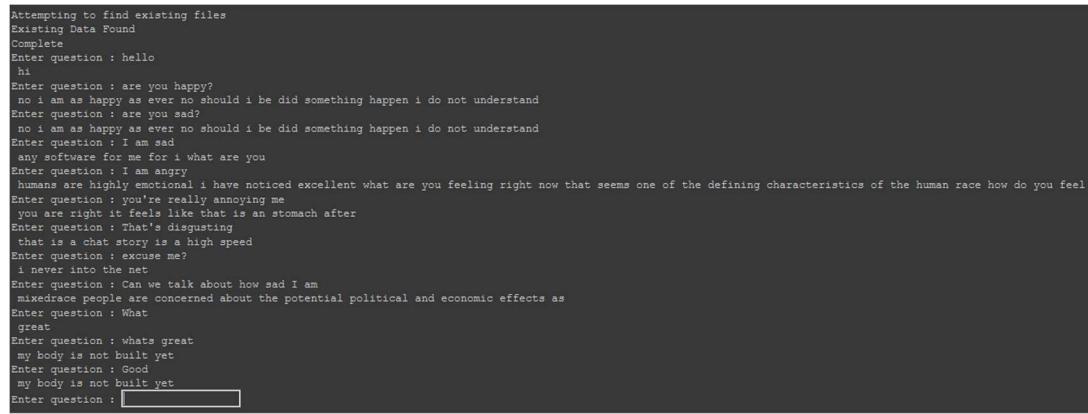
Figure 41 - Stage 2 Loss

Stage 2 reported a loss of 0.0016 and accuracy of 0.9938 at the final epoch. As suggested, these values should be mostly unchanged from the previous stage and in fact are. This demonstrates that the new classification not only functions correctly but also well. It did however affect the validation data with a loss of 0.6323 and accuracy of 0.3089. The validation performed slightly worse, likely due to the lack of emotional data use in both datasets. The question-and-answer dataset is mostly reduced to non-emotional text and the specific data only provides some emotional data in an already reduced set. This makes predicting emotional text a more complex issue. The graphs shown similar trends to the conversation simulations own. Overall, as this stage met the boundaries, the evaluation requirement has been met.

### *User Testing*

User testing exposed a third-party representative of the potential user (an edge hill student) to the current stage. They would initially use the current stage prototype then answer a specific questionnaire regarding their use, gathering external user feedback at each stage to see if the current prototype is acceptable as a proof of concept for chatbot use.

See figure 42 for the user testing at this stage.



```

Attempting to find existing files
Existing Data Found
Complete
Enter question : hello
| hi
Enter question : are you happy?
| no i am as happy as ever no should i be did something happen i do not understand
Enter question : are you sad?
| no i am as happy as ever no should i be did something happen i do not understand
Enter question : I am sad
| any software for me for i what are you
Enter question : I am angry
| humans are highly emotional i have noticed excellent what are you feeling right now that seems one of the defining characteristics of the human race how do you feel
Enter question : you're really annoying me
| you are right it feels like that is an stomach after
Enter question : That's disgusting
| that is a chat story is a high speed
Enter question : excuse me?
| i never into the net
Enter question : Can we talk about how sad I am
| mixedrace people are concerned about the potential political and economic effects as
Enter question : What
| great
Enter question : whats great
| my body is not built yet
Enter question : Good
| my body is not built yet
Enter question :

```

*Figure 42 - Stage 2 User Testing*

The user then answered a questionnaire based on the findings of the current stage (see appendix item 16). The results found that the user satisfaction was improved. They deemed the chatbot satisfactory, encountered no errors and gave a score of 6 instead of the previous stages 4. Whilst this score is still somewhat mediocre, it is enough to be considered satisfactory. They went on to say that “Chatbot seemed to respond to some of the emotions and conversation was much better this stage” and scored the chatbot as a 7 for emotional awareness. It appears that the use of emotional classification as an additional input has been successful. It should be noted that the testing at this stage was directed to ensure the user uses some emotional language and this may have impacted results. Overall, user testing found this acceptable.

### Conclusion

In conclusion, this stage was a massive success. It only required a single iteration to function as intended and managed to improve from the previous stage massively.

For this stage, no parameter choices were made or changed. The only choice was the use of the package discussed earlier which proved effective. The implementation was simple with its use of the premade package and impacted the existing code minimally. An issue, however, is the added time and the lack of ability to attempt to reduce it due to the nature of the premade package.

Testing found success in both functionality and evaluation with near identical results in these to the previous stage. This is expected and shows that the implementation of a new feature did not disrupt the program. The requirements for both had continued to be met. The user testing was more successful with the incorporation of emotional analysis. The user found the chatbot satisfactory unlike last time and the requirement was therefore met. As predicted, this stage assured the user was satisfied and all requirements were now met.

User testing found that the chatbot identified some emotions but not fully. As discussed, this was likely due to the lack of emotional content in either dataset. With the implementation of the knowledge base, emotional data will also be implemented. Currently, the requirement to provide emotional classification was partially met as without the data, it would be difficult to confirm that it was classifying based on this.

## Stage 3 - Support Base Integration

At this stage, the support base is integrated into the existing conversation simulation as a knowledge base, providing domain specificity. After the use of the focused chatbot in stage 1, a knowledge base has already been implemented to the chatbot although at that stage, it was used to provide the general domain of chatbot use. Using this dataset and expanding it with domain specificity would implement the requirements of domain specific support through a knowledge base, effectively reconfiguring the existing use of this dataset into a knowledge base. Also note that to achieve the emotional support, the dataset would be restructured to cater to this, as the dataset includes a “psychology” category which handles emotion but not for support intention. This will be remade to do so and expanded to demonstrate the emotional support component.

### Parameter Choices

The parameters selected for the conversation simulation and emotional analysis are retained. In this stage, there are no parameters to choose, only edits to be made to the dataset provided. Those are discussed below.

### Program Structure

This stage builds upon the previous ones and so uses the same structure. Any additions or changes will be discussed here. Once implemented, the code is capable of handling these modifications already except for downloading the data as the modifications are not stored. To handle this, the URL has been changed to a storage for the modified data. The data is a modification of Amin’s (2019). The code for this is shown in appendix item 22. The model of the process is unchanged and only slightly expanded in dimensions due to the additional data. See stage 1’s figures for the model.

### *Dataset Modifications*

The first modification would implement the informational support. This would be done by creating an entirely new category file in the directory. The file would use the dataset topics from the design. See appendix item 24 for this new implementation and changes to the original psychology dataset (as shown in appendix item 23).

The second modification would implement the emotional support, restructuring the existing psychology category to suit emotional support. This is done to demonstrate the emotional categorisation provided by the previous stage as the datasets used have no emotional backing and so this classification is not completely demonstrated. See appendix Item 24 for this implementation. It should be noted that the dataset has an existing emotional category, yet this only refers to the chatbot’s emotions rather than the users and does not need modification.

### Development Discussion

This section will discuss the development process for the stage rather than the code itself.

Again, only a single iteration of this stage was needed. This was to develop the knowledge base dataset for the chatbot. The process of which was simple yet tedious. In developing the datasets, it became clear that manually constructing a dataset for use was never a viable option. The time taken to modify and add the minimal data shown in appendix 24 was massive. To create an extensive dataset manually that applied itself to the domain would not be a viable option despite this likely being the necessary way to assure that there is supplied data suitable for projects such as this. It became even clearer that the current method of using a general dataset and knowledge base dataset is the best option for this project and the resources available, despite its poor performance in validation.

## Testing

### *Functionality Testing*

Functionality testing assures that each component of the program works without error and handles potential errors caused by input into the system.

As with the previous stages, testing would use boundary cases of data for input into the chatbot. This testing does not analyse the quality of the response, only that a response was made, and the system reacts in the intended way. At this stage, the functionality testing is minimal as functionality is limited to simple text input and output. As the current stage does not intend for actual use (no front end implemented) this will only test that the classification process handles the potential text. Issues with the input medium are not tested, such as if an input is too large. This will be handled in the front-end stage.

See table 5 for a record of the tests conducted.

Test	Input	Output	Success
<b>Input of string</b>	Single String ("hi")	Response ("hello")	Yes
<b>Input of second string and more</b>	Multiple strings ("how are you doing", "not bad, im pretty happy", "I'm not sure I understand")	Response for each input ("fine and you", "the best transferred for the full size", "try and talk to your friends about this")	Yes
<b>Input of string with punctuation</b>	String with punctuation ("don't tell me what to do")	Response ("is the latin and their day")	Yes
<b>Input of numbers</b>	Integer ("3534")	Default Response ("the have to be and be through the can speed and germany")	Yes
<b>Empty input</b>	Nothing ("")	Default response ("the have to be and be through the can speed and germany")	Yes
<b>Unrecognisable String</b>	Unrecognisable string ("faasdsla")	Default response ("the have to be and be through the can speed and germany")	Yes

Table 5 - Stage 3 Functionality Testing

As the table shows, stage 3 functioned correctly and therefore met the requirement to function.

### Evaluation Testing

Evaluation testing will examine the previously discussed metrics and define a boundary value for each stage to pass, ensuring that the metric performance for each stage is sufficient.

For stage 3, the loss boundary will be 0.005. This is the same as the previous stage as the knowledge base should not cause a considerable decrease in loss if implemented correctly. Similarly, the accuracy boundary will be 0.99. Despite the same boundaries, the knowledge base extension will cause a change in the metrics, although this is likely to be minor due to the small amount of change in data. Stage 3's metrics are plotted as graphs in figure 43 for accuracy and figure 44 for loss. The graph uses each epoch as its x axis and the metric as its y and is recorded as a history of the training process.

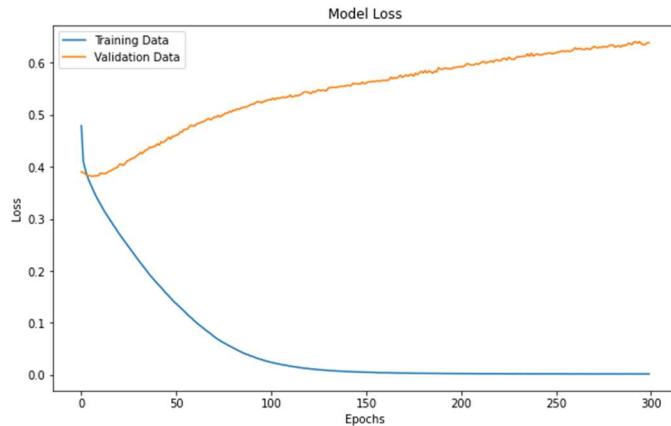


Figure 43 - Stage 3 Loss

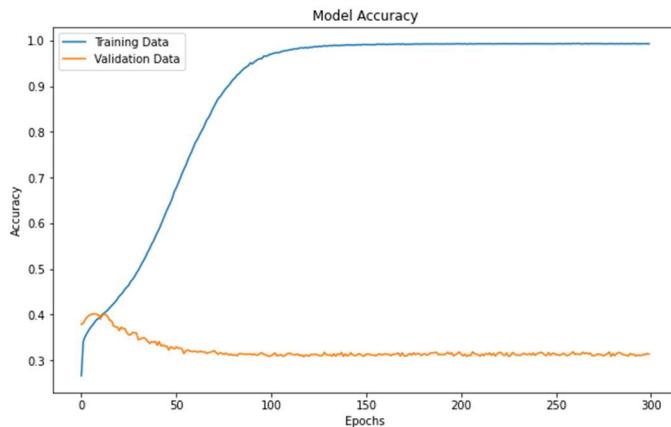


Figure 44 - Accuracy

As shown in the graphs, stage 3 returned a loss of 0.0016 and an accuracy of 0.9933. As suggested, these values only had minor alterations from the previous stages. Validation returned a loss of 0.6387 and accuracy of 0.3132 showing a similar decrease in validation's loss performance but a gain in accuracy performance, potentially showing that the previous stage's theory that expansion of suitable data reduces the poor validation performance and further supports the idea that the dataset is an integral component of the chatbot support and that overfitting of poor dataset choice. The graphs show similar trends to the conversation simulations own. The results met the boundaries, and this stage therefore met the requirements for evaluation.

*User Testing*

User testing exposed a third-party representative of the potential user (an edge hill student) to the current stage. They would initially use the current stage prototype then answer a specific questionnaire regarding their use, gathering external user feedback at each stage to see if the current prototype is acceptable as a proof of concept for chatbot use.

See figure 45 for the record of the user's test.

```
Enter question : hi
hello
Enter question : can you help me
no i am just into american for me
Enter question : oh ok
the think of the elders and the elders
Enter question : i'm sad at the moment
whats the issue
Enter question : i'm stressed about work
make if you take plenty of breaks its important to relax
Enter question : ok, ill make sure to do that
speed of honour and depends
Enter question : hey do you know where the library is
the novel or products
Enter question : where is the library
the catalyst is edge hill s library and is in the centre right of campus just northwest of the hub
Enter question : oh thanks
much the great
Enter question : what's the hub
the hub is edge hill s central student area you can find plenty of facilities in here including a mccolls
Enter question : is mccolls good
i am not a very variety of it it said i try to that a think of the conversation could be practical but what can be a did a did
Enter question : ah i see
bots fact cycad to be that the next
Enter question : the hub
is church of your people
Enter question : the library
the think of the elders
Enter question : i'm worried
you is ok to be sad take some time to process things
Enter question : i'm distraught
it is ok to be about the internet
Enter question : i'm annoyed
it is ok to be sad take some time to process things
Enter question : i'm angry
try to breathe and think things through
Enter question : i'm irritated
it is ok to be sad take some time to process things
Enter question : 
```

Figure 45 - Stage 3 User Testing

The user then answered a questionnaire based on the findings of the current stage (see appendix item 17). The results found that the chatbot continued to improve. The user found no issues and found the chatbot to be satisfactory still. The convincing metric improved to 7 and the emotional awareness to 8. The user claimed the chatbot offered sufficient emotional support as well as informational support and summarised saying "The chatbot offered some support and understood many of my emotions. At times it assumed sadness when it should have been anger though. Still the chatbot seemed quite aware of my emotions and seemed to actively simulate conversation." It is clear overall that this stage had massive impact on the chatbot, and the domain specificity brought quality to the chatbot. The improvement in emotional analysis supported the previous theory that the datasets were unsuitable for emotional support. Informational support was also provided at this stage. The improvement in performance seems to support the current theory that the dataset choice is an integral component for chatbot development. The user did mention that the chatbot sometimes mis-identifies some emotions. This may be due to the performance of the imported library and out of development control. The user testing met the requirements specified as the user found the chatbot satisfactory.

## Conclusion

In conclusion, this stage demonstrated a good performance for the chatbot and proved its success.

No parameter choices were necessary. The implementation of the stage was very simple, it only really required modifying the datasets. This was an easy yet tedious task and only a short amount of data was added. It is clear that manual dataset creation is not a valid option. It did also require some changing of the URLs to support the modified dataset.

Testing found that the stage was fully functional and evaluated similarly to previous stages. Like previous stages, both these requirements were therefore met. In user testing, the benefits of this stage were shown and the chatbot quality proved heavily reliant on a specific dataset, as this stage shown a major increase in performance. All testing requirements were therefore met.

The requirements specific to this stage were also met. It was clear that the knowledge base had been implemented and as result, so had its informational and emotional support data. Overall this stage was a success.

As this stage was the last back-end stage, it can be said that the entirety of the backend also met the requirements by extension.

## Stage 4 - Medium Integration

At this stage, the developed chatbot is integrated into the chosen medium, Facebook messenger. This stage takes the last stage's model and integrates it into the Facebook messenger model. To do this, Flask (2010) will be used as a web app interface for the code and Heroku (2020) to host the app. This will provide an accessible medium for the chatbot.

### Development Choices

The parameters selected for the previous stages are retained. Other than this, choices were made to use Facebook Messenger, Flask and Heroku based on investigation into suitable methods. No other choices could be made for this stage.

### Program Structure

This stage builds upon the previous ones and so uses the same structure. Any additions or changes will be discussed here. See appendix item 25 for the additions and modifications. This stage had to completely restructure its code to function properly. Memory limitations of the Heroku free hosting meant that the classification and processing of the data could not be fully implemented without causing considerable issues. To do this, three python scripts were used.

The first was separate from the hosting of the chatbot and controlled the training of the models and processing of the data. To circumvent the memory issues, this script would save the needed files (the encoder and decoder max lengths and the two dictionaries) for use with the interpretation models. The interpretation models were also saved for use. The code for this was very similar to previous stages, only with the input and output handling removed and a few lines to handle saving the data. See appendix item 25 for this code. The second script handled the processing of chatbot inputs for the hosting of the Facebook messenger. Like the previous script, this was also similar to previous stage's code, only separate. This script contained the input handling with use of the saved interpretation models as well as the functions to clean the input text and a function to open pickle files. This would then be called by the main messenger web app with inputs passed to it. See appendix item 25 for this code. The third script used handled the hosting itself. This was entirely new script that implemented the Flask web app service with Facebook messenger functionality. After importing the necessary modules (including the second script), the app begins by defining the flask process and declaring the verification tokens for the messenger app. These were handled by Heroku to avoid any security issue with distribution of the code. The app then began by using GET and POST protocol to initially get and verify these tokens then receive and handle user input to the web app if verified. To do so, it performs a series of nested iterations of the Json dictionary, similarly to how the general dataset was handled. It would eventually get the user input and verify if it were text or not. If text, the input handler module (script 2) would be called to predict the response for the text and send it back via the imported PyMessenger class (Chua, 2021), a module specific to handling Facebook user inputs. Otherwise, a predefined response would be sent back to inform the user of incapability of managing inputs other than text. See appendix item 25 for this code.

The second and third script would be hosted on a Heroku application deployment through a private GitHub repository. A copy of said repository is included in the stage four section of the artefact including other components to the implementation, such as a requirements file necessary to define the needed packages and the previously saved data items. The verification keys were then used to link this application to Facebook messenger for use. This allowed for a page set up to process the conversation in its messenger component. This was not available publicly however as Facebook has to verify application implementations (Facebook Developers, 2021a) and as this project was only intended as an example, it could not be verified. Despite this, the test users could be added so this would not limit the testability of this stage, only the availability of the project.

## Development Discussion

This section will discuss the development process for the stage rather than the code itself. As the medium for integration was only selected after the user requirements, very little research had been done into the implementation method for the medium chosen itself. Due to this, some research was completed prior to the development of this stage.

The research found that there were three major components to the stage, the medium itself , a method of formulating the chatbot into a web app and a method of hosting that web app. The first component, the medium itself, would be provided by Facebook as there are “300,000 developers building on the Messenger Platform” (Facebook for Business, 2019) demonstrating its suitability. Facebook Developers (2021b) supports development specific to messenger implementations through a separate site and would be capable of implementing the later component’s hosting. The formulation of the program into a web app would require some restructuring, namely with the use of Flask (2010), a web framework for use with python. The flask component will handle the input and output to the hosted web application, gathering user messages and formulating responses. Flask is a lightweight way of handling this that can rely on Facebook to provide the GUI and effectively perform the classification using pre-trained data. The hosting of the program will be done by Heroku (2020) due to recommendation by Djaja’s (2018) report into hosting chatbot’s via Facebook messenger. Heroku is a cloud platform that can host web applications in a format suitable for use with the Facebook messenger. It is also free, which is a useful feature for hosting the chatbot for the project.

The development of scripts for this component was a simple process as it mostly only required restructuring the existing functionalities. Due to the previously discussed lack of knowledge of how to implement this, the resource blog wrote by Essilifie (2017) provided strong guidance to understanding of this implementation. Despite this, the date of this blog meant some of the information was outdated and certain issues were faced.

The biggest issue was due to Heroku and its 500mb limit to deployments for free implementations. Whilst this is reasonable, the keras modelling package was around 300mb and left little room for the classification data. To resolve this, a CPU specific version of keras was used to predict from the interpreter models as this reduced the size to about 100mb. This proved to be somewhat obstructive as this package meant that the training files would have to be saved in the h5 format rather than the keras directories. Keras directories are the recommended format according to official documentation (Keras Documentation, 2021). As well, the code was refactored as discussed in the program structure to handle this memory issue. The memory issue also seemed to prove a problem during the runtime, as shown in the runtime log in figure 46. It appeared that the lack of memory impacted the process itself and may have caused issues concerning its performance, specifically the time to generate responses.

```

Personal > ehu-cls3140-project-bot-demo
GitHub Lewis-C/Chatbot-Hosting main
Overview Resources Deploy Metrics Activity Access Settings
Application Logs WEB
2021-05-03T08:51:05.877820+00:00 heroku[web.1]: Process running mem:757M(148.0%)
2021-05-03T08:51:05.893372+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)

```

*Figure 46 - Runtime logs for chatbot*

Facebook itself also caused some issues once development was complete. The policies of Facebook inhibit applications being published without verification and justification. This meant that public access to the medium implementation would be unavailable. As this was the main purpose of this implementation, this proved fatal to the projects aim. Note that user testing was available through specific allowance of single user access which was used for the user involved. Even though the aim to provide a public medium was not met, the implementation of this stage still proved suitable as a proof of concept of availability. See appendix item 26 for proof of implementation as access for reviewers would be unavailable.

## Testing

This section will detail the different testing formats used for stage 2.

### *Functionality Testing*

Functionality testing assures that each component of the program works without error and handles potential errors caused by input into the system. In this stage, this would be expanded as the functionality has a new medium with new boundary inputs to examine as well as extra handling for inputs such as a defined default value for empty inputs rather than leaving the training model to decide this. To handle this, screenshots would be used instead of simple text of the input and output. Despite the changes to the testing for this stage, it still only monitored the functionality and not the quality.

See table 6 for a record of the tests conducted.

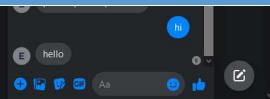
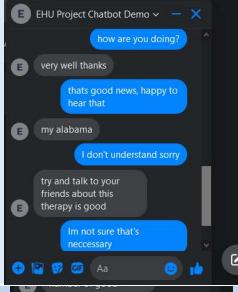
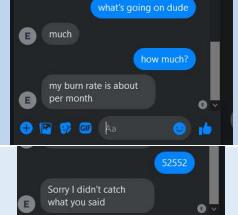
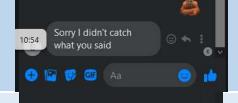
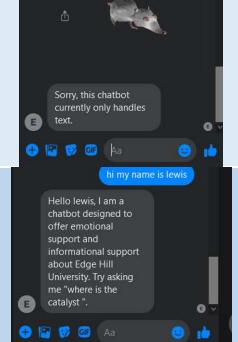
Test	Screenshot	Success
Single input		Yes (single output received)
Multiple inputs		Yes (multiple outputs received)
Input of string with punctuation		Yes (normal output)
Input of numbers		Yes (predefined default input)
Empty input	Unauthorised by the messenger app	Yes
Unrecognised word		Yes (predefined default input)
Emoji input		Yes (predefined non text response)
Image Input		Yes (predefined non text response)
Name Input		Yes (predefined name output response)

Table 6 - Stage 4 Functionality Testing

As the table shows, stage 4 functioned correctly and therefore met the requirement to function.

### Evaluation Testing

Evaluation testing is not done at this stage due to no metric use anymore. The medium integration only provides front end use to the chatbot and so the metrics are unaffected. The training models used are the same as those in stage 3.

### User Testing

User testing exposed a third-party representative of the potential user (an edge hill student) to the current stage. They would initially use the current stage prototype then answer a specific questionnaire regarding their use, gathering external user feedback at each stage to see if the current prototype is acceptable.

For this stage, this was particularly important as the user interface was implemented. Understanding if sufficient HCI was provided would be very important at this point and gaining feedback from an impartial member would be key to testing this. As well as testing the functionality, the user used their mobile phone to test the chatbot to also examine if the chatbot was available on phones.

See figure 47 for screenshots of the user's test.

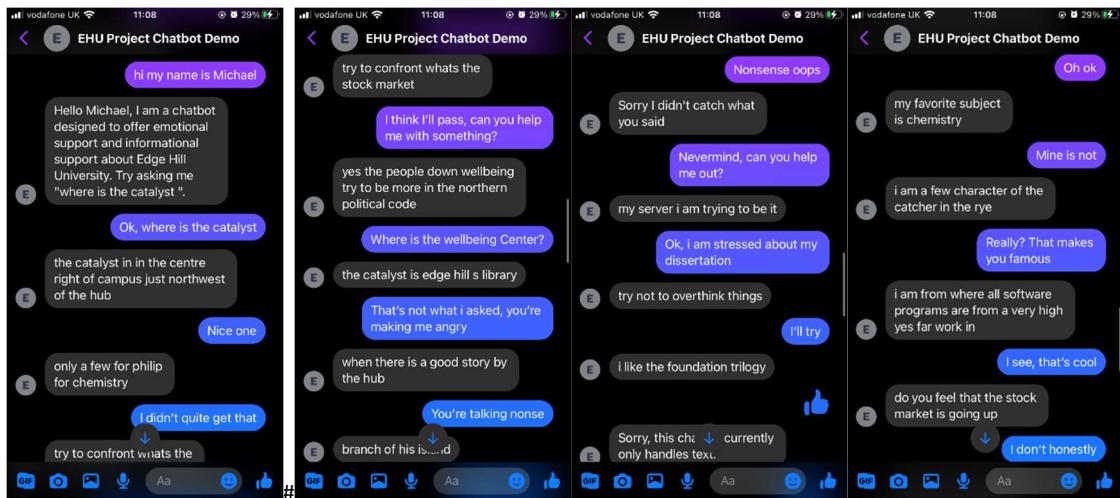


Figure 47 - Stage 4 User Testing

The user then answered a questionnaire based on the findings of the current stage (see appendix item 18). The results found that the previous theory that the direction given to the tester to assure they explored the emotional and informational features was affecting the scores was correct and without, the scores given dropped but only slightly. The chatbot was still considered satisfactory and no errors were encountered. The score of the conversation simulation was now 7 and 6 for the emotional awareness. Despite this, the chatbot still provided sufficient informational and emotional support. The user was able to access the chatbot through messenger and the response time was adequate but not instant. When asked to summarise they say "This stage was the most accurate representation of the chatbot to me because I didn't have to use a field like informational information or emotional information. I think because of this, the quality drops a little, but it was still much better than in the first stage. The use of messenger was really good. The speed of responses was good as well for the majority, but a few messages hung. This was still under 30 seconds, so it was not an issue. The handling of emojis and bad inputs was really useful as well". When asked if the user considers the current model suitable to be used for the university they responded "No, the chatbot is not really useable for any gain as the results are still inconsistent even if they do show that they can provide emotional and informational support, it's not good enough to be used as an actual tool at this stage. I do agree that it shows potential though, all the features are clearly there but the inconsistency means I wouldn't ever use it to help myself."

The user testing shown that the chatbot still performed as needed despite the slight drop due to the tester's freedom. This was expected as the training model has not changed, only the interface. The interface functioned correctly, even on the user's mobile phone. It did find some variation in response time however, which was likely another symptom of Heroku's limited memory availability. When asked to summarise their feelings, the user discussed that the testing was now an accurate representation due to the discussed freedom and they recognised the quality drop because of this but did add that the performance was still good and "much better than in the first stage" showing the impact of the other back-end stages and the use of their features. They also found that the implementation was good, noting the error handling for image input and such. They did mention as well that the response times varied, likely showing that Heroku's handling of the prediction was poor with its memory limitations.

The closing question referred to the tester being told that they should view the model as a proof of concept at each stage rather than the implementation intended. This was to avoid a constant negative results due to the discussed dataset issues that would have held no value. The user was asked to consider the chatbot at this stage as the original intention, to be used as the implementation for students. They found that it was not suitable, and their reasoning discussed the inconsistency of the results. They personally would use other avenues of support due to their greater functionality and consistent results. They do go on to say that the chatbot is still a sufficient proof of concept, however.

The results show a success of the user testing for stage 4 and as this was the final stage with all previous requirements fulfilled, a success for all. The HCI implementation of this medium was very successful.

### Conclusion

In conclusion of stage four, the front-end implementation was successful, as all testing requirements for it have been met (Functional and evaluative testing) and therefore the requirement to implement the medium was also met. However, the inability to make this public is a huge loss as this is the sole purpose of the medium integration. If the artefact were still intended for use, this would be fatal to the aim, yet due to the repurposing to provide a proof of concept rather than the implementation itself, this can still be considered successful. This stage has proven that the chatbot is implementable into Facebook messenger and with improvements to provide a sufficient artefact for use, this could be easily authorised with Facebook for use.

The implementation itself was not complex however the use of Flask and Heroku was. Lack of understanding concerning these topics and the limitations of Heroku caused some issues, especially with classification of input data. Despite this, the stage produces a suitable output.

The biggest issue other than Facebook's policy restricting public use was Heroku's memory issues. This meant that the approach taken had to be changed and the development varied from the design massively and delayed the development process slightly. As well as the memory complications causing this, it also caused issues for the runtime of the messenger implementation with warnings of memory over use and slower responses.

## Final Product

With testing finished, the final product from stage four has been produced and in turn the final overall product. As stage four was inaccessible due to Facebook restrictions, the final product folder in the artefact actually contains both stage 3 and 4. Stage 3 demonstrates the use of the chatbots, and stage 4 demonstrates the ability to host and use in Facebook messenger.

Each stage except the first met all testing requirements and as it was shown that the user testing failure for the first stage was improved upon, it can be said that the final product met all testing requirements.

The conversation simulation requirement was met in the first stage but built upon during the proceeding steps. Stage 3 was particularly good at improving this to the sufficient point it is at now where it shows capability of simulating some conversation, despite issues with accuracy.

According to user testing, the ability to emotionally classify was present throughout but like the conversation simulation, found improvement in stage three as data was supplied to support this. Similarly, there were issues although this was due to the package used as the user pointed out that sometimes the emotion was misidentified. Nonetheless, emotion was still identified, and the requirement fulfilled.

Emotional and informational support with domain specificity was clearly implemented as shown by the examples of use of the chatbot post stage 3 where it could accurately give location information to users. Other informational support was available via the knowledge base but not explored by the shown testing. The emotional support seemed to provide a baseline for helping users with their emotional issues. All three requirements for this stage were clearly fulfilled, even if only to a representative degree.

Stage 4 made a full implementation of the GUI as shown but the issue of publicising this restricted this to internal use. This did demonstrate however that this requirement could and was fulfilled.

A common theme for the development was that the requirements were implemented but due to certain issues outside control (dataset quality and Facebook authorisation for example) they could only be implemented partially. This has led to a reconsideration of the aim for the project. Instead of providing a chatbot capable of giving ample conversation simulation and emotional/informational support to a specific domain, a proof of the ability to do so is made instead and achieved . The final product would not be suitable as a chatbot for actual use with the university but does represent the possibility of combining the rarely united themes of support in a domain specific format, which is an original result of this project.

## Artefact Availability

The artefact developed will be available via the associated drop box submission and GitHub through link "<https://github.com/Lewis-C/EHU-CIS3140-Project-Artefact>". Note that if the project is run, the code has been developed in the Google Colab Pro environment and requires the hardware capabilities available there to run. A pre-trained implementation of each stage will be included to circumvent this. Assure that during running of the script, the data is in the same directory. Readme's have been included at each stage to discuss this in more detail. Please assure that during any testing these considerations are made.

Originally, the chatbot was intended to be available via messenger hosting but due to the discussed authorisation issues, is not. The code for that stage has still been included to accompany the screenshots as proof of completion.

## Chapter 7 - Evaluations

### Process Evaluation

#### Pre-Project Evaluation

Overall, prior to the project the work completed was useful and mostly consistent with the project itself. The background information served as a strong general foundation for the background research of the report and found use as a framework for much of the introduction.

The aim and objectives were mostly accurate providing a strong structure to the project itself. Changes were made to the objectives when development was further analysed with stronger knowledge of the task at hand. Changes were minimal and inobtrusive to the project itself meaning this was not an issue and helped assure development was well structured.

The methods chosen at this stage were mostly accurate and when the knowledge to provide a prediction was not available, the plan correctly chose to wait. This was an effective choice as it allowed for deep research into the topics to make the best choices possible as provided by the literature reviews later on. The methods did need some light reconfiguring for the change in objectives but as stated earlier, this was unobtrusive.

Scope analysis was accurate in finding its limitations yet the lack of detail into some may have left the project unprepared. For example, hardware limitations are briefly considered yet during development they were found to be troubling until an alternative method (hosted runtimes via Google Colab) were found. Similarly, the data availability is only discussed briefly despite its impact in the project itself. Potentially, further analysis would have prepared for this and found the alternative prior to beginning. As well, the impact of the current Coronavirus was not fully considered included potential delays due to being infected and limited access to resources. These issues had to be confronted with little preparation and so establishing plans at this stage would have proved useful.

Similarly, the hardware analysis was lacking detail. Despite finding sources to suggest that current available hardware would be sufficient, this was proved wrong and caused some delay. Further investigation should have been done and alternative solutions. Found.

Ethical considerations provided analysis of the requirements for the project, yet this was mostly naïve at this point as these considerations were lacking detail. Further discussion later in the project, specifically the interview, helped build upon this. The ethical considerations made beforehand could be considered a strong foundation for those later one.

The plans drawn for the project had a strong structure that was mostly kept to throughout (except for the development restructure) yet the times accompanying faced some issues. Certain unplanned issues arose during the process due to current events that disrupted the times and meant the plan was not kept to carefully. Despite this, the timings overestimated some tasks giving much of this time back.

Overall, pre project planning and preparation was mostly held throughout the project process. Despite the issues with timing, hardware, and development structure the plans served as a strong foundation for the project providing a suitable structure for development.

### Background Research Evaluation

Before starting the project, background research was conducted with many of the topics recognised from the proposal stage. This background research proved integral in later stages as the information gathered would support most of the choices of the project as well as build an established knowledge of the topic. Due to the background knowledge, later processes could continue comfortably.

### Literature Review Evaluation

The literature reviews defined in the proposal were completed to find information regarding certain major choices in development. Most of the literature reviews allowed for making the choices except for the medium choice, which did not. As well, an extension of the classification method was conducted to find the programming language and packages to be used. Whilst the literature reviews were largely successful (even the medium choice was as it provided a strong understanding of the potential choices to be made later) some of the intentions seem misguided in reflection. The classification method focused on the DeepLearning model to be used and not the programming method itself until the review was extended. Focusing the literature reviews on the correct aspects may have potentially provided better information at this stage in retrospect, yet the information gathered still proved useful.

Another issue was the dataset investigation. The review conducted only evaluated the datasets in terms of size and existing performance. The literature review should have examined the contents of the datasets to find their suitability. This would have likely assured that the issues faced in the first stage were dealt with before beginning development rather than during. Despite the lack of these observations, the investigation did provide ample data regarding the choice made later. As well, the idea to conjoin two datasets to circumvent the issues was provided by this review so whilst it did not fully prepare for the issues, it did provide some prior understanding.

### User Requirements Evaluation

The user requirements were found by two methods, a survey, and an interview.

The survey produced useful results that allowed for the choice of medium to be fully made as well as gathered some of the features users would likely want. It did also provide a better understanding of what the chatbot should focus on as well and further justify the project. The information obtained allowed for the project to make certain choices based on a generalisation of users and was therefore useful. Yet the data is potentially questionable due to its small sample size (19). Whilst circumstances were obtrusive to this process, a larger sample size would have ensure better data credibility. In future, larger samples should be taken.

The interview produced very little new information and only reinforced existing data. The questions asked were largely limited by a lack of specificity, although this was necessary to keep data confidential. Whilst the interview provided little new data, the reinforcement of existing knowledge was useful especially from someone with first-hand experience. This helped justify certain choices made.

Overall, in reflection the user requirements may have been misguided. A lot of the two techniques established justification for the project itself rather than gather new data. Despite this, the data gathered still helped accomplish what was necessary. As well, it has become apparent that the user requirements are less integral to the chatbot quality than certain understanding of the development. Reallocating the attention of this process to research or development of the chatbot model itself may have proved more useful.

### Designs Evaluation

The designs provided gave a strong guidance for the development process all around and were mostly unchanged in the development itself until the issues encountered in the fourth stage. The use case diagram and the requirements provided a set list of implementations for the development to analyse at each stage to manage progression of the agile process. The requirements established were given set values to reach at each stage and the designs for the requirements helped allow this. These designs were integral to the flow of the development process. The GUI design was likely unnecessary but did provide an understanding of the medium display itself before integration. The class diagram and data flow helped visualise the program specification before use. The class diagram varied from implementations at many of the stages but gave a strong idea of the flow of development to follow and was therefore effective. The data set designs proved helpful for developing the knowledge base and gave the process structure. Overall, the design stage was a strong supporting factor to development but independently, it was a step that produced very little information other than that used to supply the development.

### Development Evaluation

The first stage of back-end development aimed to implement the conversation simulation component. It did so successfully but with issues that meant this stage took 3 iterations to complete. The first encountered hardware obstacles that meant training the chatbot took around 10 hours and so much of the time of development was spent waiting for this as development would need to wait this time to see if the conversation simulation was working. After reconsideration that has been discussed, Google Colab was used, and the process became far easier and proved to support development. In retrospect, better preparation for this was needed for this stage to perform better yet after resolving this issue, there were no problems with hardware and the project faced no losses in its artefact, only in the time taken to develop. The second issue encountered was the dataset quality. As discussed, the original dataset was unsuitable for a chatbot and so a specific dataset for chatbots with smaller size was conjoined with a larger more suitable chatbot to provide the functionality. Despite the suitability being solved, the quality was still poor as the dataset that was specific to chatbots was still too small and the extensive dataset was not entirely suitable, although more so than the original. After further analysis, it was decided that this was the best option available as the datasets available were of lower quality than those used. This meant that not only did this issue delay the project but reduced its quality massively. The chatbot would not be capable of being an alternative to the current implementation because of this and instead only achieve its aim as a proof of concept. Certain restructuring of testing and development had to be done to accommodate this as it now had to be accepted that the quality expected would not be met. After handling the issues, the third iteration developed the chatbot conversation simulation as expected. The process was complex, and this stage was the greatest of the four, yet the produced prototype was functional to the newly set expectations. Functionality testing found no issues. Evaluation testing found that the model's loss and accuracy was performing but the validation data shown poor performance. Validation data was not a requirement and so the evaluation requirement was still met regardless of this. The validation data shows that the model overfits the data. This is theorised to be a symptom of the poor dataset choice and not an issue with the model (which does attempt to avoid this with a dropout layer), and as poor dataset choice is now out of control, this issue has to be accepted. The overfitting was likely the reason for the poor user test performance. The poor user test performance meant that the user testing requirement was not met at this stage. Despite this, development continued as it was expected that this would be resolved later, which it was, and that the cause was the dataset which had already been reconfigured to the best possible option.

The second stage was far simpler. It aimed to provide emotional classification as an additional input feature to the chatbot. The implementation of emotional analysis was provided by an existing package (Sharma, 2020) and integrated smoothly

into the existing program with only 1 iteration of the agile process required. The only minor issue was that without developing the function independently in the project, the process implementation could not be controlled, and the package appeared somewhat slow. If the emotional analysis were integrated independently, further analysis could be done to potentially improve this at the cost of the development efficiency. This was only a minor concern, but a concern, nonetheless. Given that the package implemented lexicon-based classification, it is likely that the alternative machine learning implementation would have been more efficient with the use of the available hardware. Despite this, the development efficiency of the package justified its use as incorporating the required emotional analysis proved simple. Testing shown full functionality and identical evaluation metrics to the previous stage showing that the new feature caused no disruption. It did however disrupt the validation data and decreased the quality of those results slightly. This may suggest that this has an impact on the overfitting with a new feature introduced. Despite this, the user testing found better results and found the chatbot to be satisfactory. The user testing found that the stage improved the chatbot performance and was somewhat effective at monitoring emotion. However, it should be considered that the user was directed to assure that emotional inputs were used which may have had some effect on the results. Even then, the testing shown an improvement and that this stage fulfilled the requirements where the previous stage did not. As a result this stages implementation of emotional classification was a success.

The third stage was even simpler still. Implementation only involved editing the existing chatbot specific dataset into a viable knowledge base that could supply the support needed. As a result, only one iteration was necessary. Despite the simplicity, this stage was tedious. Manually implementing the data proved inefficient and shown that the option to manually develop suitable datasets for the chatbot was not available despite earlier consideration. It became clear that despite the use of premade datasets being disappointing, it was still the best option with the available resources. Testing proved successful with near identical evaluation and functionality and therefore met the requirements. The validation results did vary however as accuracy improved with the addition of extra data specific to the use. This supports the ongoing theory that the overfitting was a symptom of the limited dataset use and not something else that could potentially be fixed. User testing found good results at this stage with clear use of support mechanisms and improved conversation simulation and the user found this prototype suitable. As a result of these tests, the requirements for the knowledge base were also met and the stage successful.

As a group, the back-end stages varied. The first being the most complex and causing the most issues with the rest being simple additions to the foundation it provided. The structure of the code allowed for this easily. The development itself was difficult but only due to issues outside of the programming process control. The issues faced delayed the project but not massively and only one issue was present at the development end, the overfitting likely caused by the dataset. The impact of this issue was massive however, but all requirements were still met at the end of the back-end development. The back end was a successful implementation of all the requirements and functioned as intended, giving a conversation simulation that classified on emotions as well as text with domain specific and emotional support.

The fourth stage had a complex implementation yet simple coding. Most of the code used was refactoring of existing code except for a small portion of code that made simple flask requests for the web app. The complexity was introduced due to the limitations of Heroku mainly. As the hosting service was free, the memory it offered was limited (500mb). As the original module for classification took around 300mb to implement, this memory was incredibly limited for use and workarounds had to be taken. The most major was restructuring the web app code to be as minimal as possible and importing the most possible data from a pre-executed training and data prep script. This meant that the designs were inapplicable to this stage and the development of this stage was unprepared. As well as this, the module choice itself had to be reduced. Instead, a CPU only limited implementation of Keras was imported into the hosting solution as Heroku did

not use GPUs to process. The pretrained data did not need GPUs to function correctly so this wasn't an issue. This reduced the data size massively but caused some indirect issues. One of which was the model format saved was not appropriate for the CPU limited implementation. The models had to be retrained to be saved in the suitable format, taking extra time to develop. Even with all of these considerations in place, the memory was still lacking as shown in the previous discussion and runtime was impacted, causing slow message response and at times, hosting errors. Despite these issues, this was the best available option for hosting the web app. As well, Facebook itself caused a major and unfixable issue in that the chatbot would not be available publicly. Facebook had introduced new authorisation policies that meant the app had to be verified before hosting publicly. Due to the nature of this app (as a prototype intended for a component of a project) this would not be possible and the purpose of this stage (to create an accessible medium) would not be accomplishable due to this. However, as previous discussion has suggested in regard to the outcomes of the dataset quality, this would still prove to be a sufficient proof of concept. It would demonstrate that the chatbot is hostable and accessible via public mediums with further development needed. Despite this issue, the chatbot still functioned well on a private instance of Facebook messenger as shown in the evidential screenshots and testing. Overall, despite the obstructive issues, the front end was successfully integrated, only it would not be accessible at this time to the public. The testing shown that functionality requirements were still met, despite the new features and user satisfaction was high with the GUI and HCI principles were successful implemented. User testing was no longer directed at this point and so it gave a free impression of the chatbot to the user which found that the chatbot itself was still capable but quality without direction was slightly decreased.

The overall development process was successful as all requirements were eventually met. The process did demonstrate some issues that would limit the project however yet even with these, it was still a success.

## Performance Evaluation

Overall personal performance was good. When circumstances were ideal, the objectives were met within acceptable timeframes and in many cases faster than the plan had predicted. The objectives were met and, in many cases, surpassed such as an extension the literature review for the classification. Prior knowledge and background research allowed for the process to be conducted more effectively throughout.

Research during the project was of high quality with numerous and various credible sources to back the information up. Literature reviews were similarly strong. The user requirements contained detailed data gathering however the interview process may have been improved. The transcript shows ill preparedness and potentially unstructured conduct that was harmful to the interview and may have affected results. The analysis of this stage was in depth and provided key data however. Designs were closely suitable for development and gave a strong structure to follow. Development faced numerous issues, but the code quality was high despite the hardware limitations. The structuring of the code supported the development massively.

Certain issues, specifically with timing, arose due to the current events during this project. The Coronavirus impacted both the time and available resource access. This should have been considered in retrospect yet was not and the timing of the project suffered. In reflection, further planning should have been made to compensate for this and provide a more reliable plan. As well, the issues faced in development stages required further prior knowledge into datasets and hardware. Despite the quality of existing research, a more extensive research plan was needed to accommodate this and potentially avoid the encountered issues. As well, very little research was done into the chosen medium. If research had been done, Heroku's memory issues and Facebook's authorisation could have been prepared for with suitable designs and preparation for verification. Instead, these issues were discovered during the stage's development and encountered without suitable preparation.

In summary, personal performance was good but preparation for certain aspects was poor. With a greater understanding of the project, it is clear that certain aspects were focused on unnecessarily (such as gathering user requirements) and others ignored (such as planning the dataset implementation). Despite this, the performance still accomplished the requirements and objectives and fulfilled the restructured aim. It was therefore acceptable.

## Goals Evaluation

During the project, certain goals for the product were set. These being the requirements for development, aims and objectives. As discussed in the development evaluation, all requirements were met.

In reflection of the objectives for this project, all were met. The objectives for background research were all met and even exceeded in some cases such as the extension to discuss programming languages for the classification model. Many of these objectives were not immediately met however, and some took further investigation to complete, such as the medium choice. The identification of user requirement objectives were also met with an extensive interview and survey process conducted, which found suitable and useful data. All of the design objectives were implemented as well. The development objectives were accomplished, with each stage developing the feature and testing fully with use of the agile system to iterate and correct if the results were inadequate. Overall, the objectives were followed closely during the project and provided a strong framework to produce this outcome.

The original aim, "to develop a chatbot that provides sufficient emotional and informational welfare support to students at Edge Hill University" was not met but the outcome did accomplish the creation of a format that could be scaled to do so. The limitations present meant that the developed product incorporated the objective features and met development requirements but lacked the quality of such to be useable in an accessible format. This meant that whilst the outcome was not a success as originally planned, it was successful in another way that could be used for future research and development. The project demonstrated the capability of the objectives and requirements in such a way that could be scaled in future work to be useable. Instead of accomplishing the original aim, the project accomplished a demonstration of how informational and emotional support could be combined in a domain specific area. This accomplishment was original, as discussion of support chatbots that catered to both emotional and informational aspects was incredibly limited.

In conclusion, all goals were met except for the main goal. Despite this likely being the most integral, the outcome still produced an artefact that accomplished a suitable aim. Overall, this project achieved the goals not in the way that was originally planned.

## Choice Evaluation

Throughout this project, choices have been made that have affected development. Many of these choices had viable alternatives that may have impacted development. To evaluate, this section will take a retrospective look at many of the choices made and discuss the potential alternatives and what impact they might have had.

The first major choice was the use of Seq2Seq modelling for the chatbot. This had no real viable alternatives as was found during the decision to use this model and so nothing else can be considered. During the literature review for choosing that method, the choice to use python was made. Similarly, the alternatives to this are not viable due to the disproportionate use of python in comparison to alternatives in this field (Developer Economics, 2020). Because of this, there was little option or reason to use the alternatives as the lack of support would have made development a far more complex process. This made python the clearly justified choice. The next choice made was the package used for python. Keras was selected due to its flexibility during development (Bhagwat, 2020; Sayantini, 2018; Terra, 2020). This proved useful for developing the project however, now that development has concluded the supposed better performance at the cost of this flexibility from the other packages may be a viable option. Through development of the project, moving the current model to these separate packages would potentially improve the chatbots performance. Despite this choice being considered now, it was still likely the stronger choice to use Keras leading up to this as the flexibility was likely a benefit to the development process. This choice should potentially be considered as a future addition rather than a change.

The same literature review also discussed emotional analysis. Whilst this was the most effective way to implement an emotional component to the chatbot with no other viable options, the method of doing so had a strong alternative. The package used, text2emotion (Sharma, 2020) used implementations of the multiple NLP packages such as NLTK (2009) to provide the emotional scores through a lexicon-based method. An available alternative would be to implement an independent deep learning process as described by Yam (2015). Whilst this would introduce an added complexity to the development, it may improve the classification. In the user testing of the third stage, the tester found that the chatbot misidentified some of their emotions. This was theorised to be a result of misidentification from the package and by using a deep learning process, this may have been avoided although this cannot be confirmed. Another issue is the time taken to process the data and find the emotional analysis. It can be assumed that the package does not utilise the GPU use which would be available to a DeepLearning process and given Google Colab's strong hardware, likely perform very well as a result. This would potentially reduce the time taken to emotionally analyse the data, but this also cannot be confirmed until tested. In conclusion, using a DeepLearning approach would be a suitable alternative although the results should be compared and tested together before making a judgement of superiority. This would likely be an expensive process for development. In the current use, the use of the package can be justified as it produced suitable data with minimal development cost. DeepLearning may improve this at the cost of efficiency.

The next choice to be made during the literature review was the medium to make the chatbot available. After discussing with potential users, messenger implementation was chosen. Of the choices available (messenger, web integration and independent software) only two were viable, these being the messenger and web integration. It was found that the independent software use was a poor choice in the user survey where it received very poor feedback from users (as shown in figure 15). Of the two viable options, both could be considered. Whilst the messenger integration promised ease of implementation and a pre-made GUI, the web integration would give greater accessibility and more development control. The biggest reason the messenger implementation was chosen was due to the lack of need to integrate with an existing system and gaining the approval to do so. The university website would have likely not been a suitable environment to test within, yet the messenger app was and so, during this development it was more suitable. Despite this, with development

complete, web integration may be a viable option when suitable improvements are made. With the better performance, approval of implementing into the website is more likely but during this process of development it would not have been. Messenger was the better option in retrospect for this process. It provided a more flexible method of development that allowed for better control and is therefore a justifiable choice. For future improvements, web implementation is likely to be the superior option.

The dataset was chosen next. This was a source of many issues for the chatbot, and many alternatives had been considered throughout. Originally, the Cornell Movie Dialogue Corpus (Danescu-Niculescu-Mizil and Lee, 2011) was used but this found issues as it was unsuitable for chatbots due to its use of movie plot devices as well as profanity. Instead, a synthetic dataset was considered, ConvAI3 (2018) that had made a corpus with another chatbot. This was not used as the data contained was poor, despite its relevance. Finally, analysis decided to use a conjunction between a smaller, specific dataset for chatbots (Amin, 2019) and a collection of questions and answers named SQuAD (Rajpurkar, 2016) that were more suitable to the chatbot usage. Despite the improvements in suitability, the data was still performing poorly with evidence of overfitting caused by it. Testing found that model overfit the data despite use of a dropout layer to rectify this. It did however improve as data was manually edited in a later stage, suggesting that the improvement of the dataset would fix the overfitting and that the data was therefore low quality. Analysis at this stage found that there were no more suitable options of pre-made chatbots available for use yet creating a dataset would be an option. During the third stage, this was found to be a tedious task and not viable if done manually, however automation of this should be considered. Like the ConvAI (2018) dataset, synthetic datasets exist. As Nelson (2020) defines it, synthetic datasets are those that have been generated rather than created or harvested from existing sources. This avoids the tedium of manual creation and also avoids the issues with harvesting data from certain sources where privacy may be an issue and provides a strong middle ground of the data quality as it allows for specificity towards a purpose of use. This method of formulation should be considered as a strong alternative to the use of pre-made data due to the ability to cater towards a defined use. With further investigation into datasets, this should have been considered and potentially used yet the lack of focus on datasets during the research caused a loss in potential for the project.

The literature reviews continued with a general discussion of existing chatbot features, where some were selected for use. This found that the use of knowledge bases and additional input features were required for the objectives and aims, so no real alternative implementation of them can be considered. However, the knowledge base integration had a potential feature that was ignored due to lack of need. Named Entity Recognition is a method of analysing text and picking out specific features such as locations to find information specific to that feature and is a key component of many domain specific chatbot and processing methods (Li, 2018). Despite its usefulness, the project did not necessarily need it as the use of a knowledge base dataset alone proved sufficient and this may have simply added to the already extensive development time with little return. Despite concerns of development cost, Mikheev et al (2019) proposes a model that does not require an extensive list of elements to still function effectively. In reflection, this should have been implemented to improve the domain specificity of the chatbot yet without, the chatbot still functions as intended in this focus so the exclusion is not a major loss. Two fewer essential features were found, pretraining and the use of an attention layer. Pretraining was avoided as the use of an existing data source for the process would have likely been unsuitable and caused issues with missing data. Pre-training of the embedding layer was originally considered via the use of Glove (Pennington et al, 2014) but not chosen due to this concern. In retrospect this choice was beneficial and the prediction likely true with minimal cost if not due to the use of Google Colab and its high-performance environment. Despite this, without the hardware provided, pretraining should definitely be used to reduce the cost with some performance loss. This was of course unnecessary in this case. The other feature, attention, was recommended against by Ludwig (2018) however the

lack of performance of the model may have been fixed by this implementation, although this would need testing before full implementation. Given that the model still met the requirements, the exclusion can be justified as well as Ludwig's (2018) reasoning.

The next major choice to make was the development environment. Initially, this was approached naively believing that the current hardware was suitable, and that the local IDE Spyder could be used. This was proven wrong in the first stage of development and reconsideration had to be done. After more investigation, Google Colab (Google, 2021) was found. This was a hosted solution that allowed access to extensive hardware. It was clear that this should be used. No viable alternatives were found and if there were, they were likely unnecessary to consider as Google Colab performed far beyond any needs.

Now that development had begun, certain parameter choices were made. The first choice was of hidden layers. Eckhardt (2018) gave two equations to calculate this, one of more complexity than the other. The simpler equation was used to find that 232 hidden layers should be used. The development faced no issues due to the number of layers, but the other equation should be considered to likely find a better value. This was not used due to decision of the scaling factor. With further research, a scaling factor should have been chosen and this equation used as the results were likely better and this may have improved the model although only slightly. Another parameter for consideration was the batch size. One of the few investigations into comparison of batch sizes was conducted by Shen (2018) and found 64 to be a strong performer hence its use. Despite this, the choice could have been tested specifically to the model developed rather than Shen's analysis that may be only suitable for their data. An alternative approach to the one chosen would have been to use batch gradient descent although this may have caused memory issues at only a small potential gain. To find the most effective batch size, testing specific to the data should have been conducted yet this would have taken time to do. The final parameter was the number of epochs to use. In retrospect, 300 was too many. The testing graphs all shown that around the 175<sup>th</sup>, the accuracy and loss plateaued yet the validation data continued to decrease in performance. It is likely the high use of epochs contributed to the overfitting, but the graphs do show that they were not the sole cause as the validation data decreased in quality even in early epochs. A lesser amount should have been considered with pre-emptive testing to find an effective amount. This would have likely given an amount around 200. The dataset size was chosen to the limits of the environment and so cannot be considered to change, the current best available is used.

Another choice to discuss would be the use of dropout and alternatives available. Despite dropout being an effective addition (Prickett et al, 2018; Srivastava, 2014) it did not seem to have much effect of the model. Currently, this is theorised to be due to poor data choice rather than ineffectiveness of dropout but as this is only a theory, alternative methods for this should be discussed. Regularization expands on dropout functionality as discussed by Kuckacka et al (2017) and introduces potential measures for how the data should be dropped out to remove overfitting. An investigation into these methods should have been provided during the research to find an effective method that expands on dropout. Currently, no specific choice can be made yet given that these methods are more complex than simple dropout, they are likely more effective as well and therefore should be used.

The three major medium integration choices should be analysed. Flask is considered one of two most popular web frameworks for chatbots, with Django serving as an alternative option. In a comparative report between the two, it was found that Django offers a restrictive yet well supported framework, whilst Flask offers a flexible yet less supported alternative (Mieczkowski, 2021). Given that flask proved simple to implement for this project, the benefit of Django's support would have likely been little help despite its restrictive nature. Due to this, the choice of Flask over Django is clearly justified as it was still simple in this project yet would not be limited in scalability.

The use of Facebook messenger should be reconsidered. In context of messenger applications, this is the clear choice due to its extensive use and support for this medium (Facebook for Business, 2016; Facebook for Business, 2019) however the alternative choice of web integration is a strong alternative. With the issue of the need for authorisation, the limited nature of messenger integration became much more apparent. This was originally discussed, yet this limitation was not considered. With the understanding that this project should be considered a scalable prototype, web integration into the university website seems like a clear direction to take as it would allow full control of the GUI with no restrictions. It would also satisfy the findings of the interview as it would verify the chatbot as official university content as requested by the interviewee. The accessibility of this would also be a major improvement. The added complexity would be an issue but with potential scalability, this is to be expected. For the current project, the attempt at Facebook integration was the clear choice due to simplicity and demonstration of the chatbots ability to function online yet for future scaling, web integration should be used.

As an extra consideration to this, if the chatbot was integrated into the university site itself an alternative to Heroku would be available through the server hosting solution of the site. Given the size of Edge Hill University, it is very probable that the server used would be far more capable of hosting the chatbot than Heroku and provide more memory to the runtime. The use of a private server is a clear one for future development yet during this project would have been nonviable due to the costs and accessibility. Similarly to the choice of Facebook messenger, the simpler option (Heroku) is justified for this process, yet any future scaling should utilise a private server to allow for better hardware access.

Overall, all choices made were justified early in development and continue to be reasonable choices. Many of the alternative choices can only be considered now that a functional model is made to scale from, and the choices made were a key component of getting to this point. Many of the current choices were made due to their suitability for limited development as in this project and helped to accomplish creating the proof of concept for this project. In conclusion, the choices made were effective.

## Chapter 8 - Conclusion

### Summary

To summarise, the project was partially successful. Whilst it did not achieve the original aim, it did prove that it could with further development and the artefact was shown to be a strong foundation for accomplishing this. All requirements and objectives were met through every stage and each part of the process was completed with useful results that supported development.

To be critical of the process, the focuses should have been reallocated. Whilst certain aspects were beneficial and useful, the extent was too far when other aspects should have been given more detail. For example, the interview produced very little new data yet took a lot of the project resources whilst research into the datasets was only brief and potential solutions to the overfitting issue were not discussed and so, the project suffered. With reallocation, the original aim may have been met. As well, the timings were inconsistent due to a variety of reasons but the biggest and most accountable was a lack of preparation. In the future, analysis of aspect importance and further preparation for the most important aspects would be integral to ensuring a viable project.

Despite this, the process was approached well with extensive detail and strong understanding. Personal performance throughout was good and not the direct cause for many of the major issues.

The major issues discussed were largely outside of any control during the project and were therefore limitations. Dataset availability was poor for the domain approached and the authentication of Facebook apps would not be suitable for a project such as this. Despite this, preparation for these issues should have been made to potentially handle them better.

Future aspects for the project are encouraging. The artefact developed is a strong foundation that with improvement discussed in this chapter could prove to be a strong implementation of the DeepLearning and a chatbot of high performance.

Currently, the artefact produced is a proof of concept of achieving the original aim. Despite this being less than the original intentions, this project has still been massively beneficial and established research into a previously lacking aspect of chatbot development, the combination of emotional and informational support. In this context, the project was a success.

## Limitations

As discussed, the project faced to major limitations. The first being the overfitting likely caused by the dataset. Overfitting causes validation quality of a model to decrease as the epochs iterate which was clearly demonstrated during the evaluation testing of each stage. The dataset choice is theorised to be the cause of this due to its lack of quality and that during stage 3, when the dataset was slightly improved to incorporate the knowledge base, the validation quality improved showing that dataset quality is related to overfitting in this case. As well, implementations into the model itself to rectify overfitting (dropout layers) seemed to have little impact showing that the issue was most likely not due to the model. Despite this finding, the dataset incorporated was the best possible option found by extensive evaluation and as such, could not be changed to solve this. Stage 3 also demonstrated that improving the dataset manually would not be an available option due to the tediousness and time taken to do so. In evaluating certain datasets later into development, an alternative option was found to use synthetic data formulation which would have provided quality domain specific data without the tedium of manual creation. This finding was too late into development however and was not useable by this time due to lack of it. Because of this, it can be supposed that this limitation may have been solved if the research earlier into datasets was more extensive. As the personal evaluation discussed, the focuses of this project needed re-evaluation that may have solved this limitation. This limitation meant that the classification provided was low quality yet still demonstrated the ability. Because of this, the aim would not be met despite the other goals being reached. To support this, the aim has been reconsidered to be a proof of concept of combining the support vectors in a domain specific area instead of actually providing the product. The project was successful in accomplishing this new aim and with further development to resolve the limitation, the original can be met. This project serves as a strong foundation for this.

The other limitation faced was the lack of public availability for the project due to Facebook's authentication procedure. Handling this procedure was outside of project control, making this a limitation yet with research prior to development this may have been prepared for and resolved. Similarly to the dataset issue, this could be considered a lack of suitable resource allocation during research. Instead, the medium limitation was only found when the project had been developed for it. This was too late to rectify, and verification would not be available due to the nature of the project. Despite this, with the newly reconfigured aim this was still suitable. The medium served as a proof that the chatbot could be integrated, similarly to how the chatbot could perform well with further dataset development. As a result, this limitation didn't disrupt the new aim but was fatal to the original one. With further development, the medium could be authorised and the current implementation of this would be publicly available.

Overall, the limitations meant that the aim of the project had to be reconfigured to be considered successful and were therefore disruptive. Despite this, the new aim still provided original and suitable findings for the field and so the project was still successful.

## Future Work

The future of this project holds strong prospects if development is extended. As the limitation found, the issues with the dataset meant that the project should now be considered as a proof of concept of the ability to make a chatbot that can provide emotional and informational support whilst simulating conversation rather than a full implementation that can be used functionally. The choice evaluation section discussed alternative options available and found that many were the more viable choice if the project were to be scaled, to assure an improvement in quality. This section will discuss the previously identified extension features, analysing if they were met at the current iteration and how they could be in the future if not.

The first extension requirement identified was "Constant Availability". With the use of Heroku, this was intended and somewhat incorporated, yet the limitation of Facebook authorisation and the memory issues encountered meant that this was only privately and inconsistently, due to likely crashes because of Heroku's limitations. The previously discussed use of private servers and web integration would implement this feature as an extension of their own functionality. The second requirement was "Output of links in the conversation". This would be implemented via the use of Entity recognition to identify topics to link to, such as if the user mentioned a course at edge hill. Implementing this would provide extension to the knowledge base and is therefore a strong candidate for additional features to the chatbot. The third requirement, "output of images" would likely be approached in the same way such as recognising locations and sending pictures or a map. As well, so would the fourth. "External recommendation for complex support" would likely require identification of trigger words that are too difficult for a chatbot to handle such as suicide or self-harm. Details of an external system such as the wellbeing centre could then be sent. These three extension features clearly demonstrate how use of Named entity recognition would support the chatbot use, specifically its support component. To scale and improve the existing chatbot, this would be a key implementation. The next two features, "phone an pc access" and "basic accessibility functions" had been implemented indirectly due to the use of messenger. As a medium, messenger comes with many features such as these available already. Another extension feature listed that the service incorporates is "Vibration Inclusion" and potentially "text to speech" if available on the device used. The incorporation of existing features such as these through messenger is a strong reason for its use however, previous discussion has identified that web implementation may be the better option. If this is the case, these features would have to be manually implemented, likely through the development of the web app and Flask itself. The extension feature "referral to past conversations with anonymity considered" would likely require a system of storing user details and encrypting them as well as a login system to differentiate the users. This would be more suitable using a private web server with larger possible memory and data management available. This feature would require an extensive number of resources but would help to provide the best possible service. This could be considered one of the more farfetched extension features. Potentially the two most complex features are the processing of both voice and images for input into the chatbot. These would require their own software solutions and add a new layer of complexity to the issue. Discussion of implementation for these features (da Silva and Mendonca, 20XX; Wilpon, 1995) explores the complexity of incorporating these, demonstrating how complex this would be to integrate.

In conclusion, some extension features are already met by the use of Messenger yet with the proposed change to website integration for future development, these would have to be reimplemented. Of the other features, named entity recognition appeared the most beneficial with little cost to implement in comparison and should be considered as the first extension feature to implement. Constant availability would be implemented with use of previously suggested private servers and web integration. This feature does not need specific integration. The tracking of user details and processing of other input types can be considered the most complex. If implemented, these should be considered last.

## References

- Adam, M., Wessel, M. and Benlian, A. (2020). AI-based chatbots in customer service and their effects on user compliance. *Electronic Markets*.
- Adamopoulou, E. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, [online] 2, p.100006. Available at: <https://www.sciencedirect.com/science/article/pii/S2666827020300062> [Accessed 27 Nov. 2020].
- Agile Alliance (2019). *What is Agile Software Development?* [online] Agile Alliance. Available at: <https://www.agilealliance.org/agile101/>.
- Akasaki, S. and Kaji, N. (2018). Chat Detection in an Intelligent Assistant: Combining Task-oriented and Non-task-oriented Spoken Dialogue Systems. *arXiv:1705.00746 [cs]*. [online] Available at: <https://arxiv.org/abs/1705.00746> [Accessed 16 Feb. 2021].
- Al-Masri, A. (2019). *What Are Overfitting and Underfitting in Machine Learning?* [online] Medium. Available at: <https://towardsdatascience.com/what-are-overfitting-and-underfitting-in-machine-learning-a96b30864690>.
- Ambler, S.W. (2008a). Chapter 5: Usage Modeling. In: *The object primer : agile model-driven development with UML 2.0*. Cambridge: Cambridge Univ. Press.
- Ambler, S.W. (2008b). Chapter 9: Process Modeling. In: *The object primer : agile model-driven development with UML 2.0*. Cambridge: Cambridge Univ. Press.
- Ambler, S.W. (2008c). Chapter 12: Structural Design Modeling. In: *The object primer : agile model-driven development with UML 2.0*. Cambridge: Cambridge Univ. Press.
- Amidi, A. and Amidi, S. (2019a). *Architecture of a traditional RNN. Recurrent Neural Networks cheatsheet*. Available at: <https://stanford.edu/~shervine/teaching/cs-230/illustrations/architecture-rnn-ltr.png?9ea4417fc145b9346a3e288801dbdfdc>.
- Amidi, S. and Amidi, A. (2019b). *CS 230 - Recurrent Neural Networks Cheatsheet*. [online] Stanford.edu. Available at: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>.
- Amin, H. (2019). *chatbot\_nlp*. [online] kaggle.com. Available at: <https://www.kaggle.com/hassanamin/chatbot-nlp> [Accessed 29 Apr. 2021].
- Annett, M. and Kondrak, G. (2008). *A Comparison of Sentiment Analysis Techniques: Polarizing Movie Blogs*. [online]. Available at: <http://ccc.inaoep.mx/~villasen/bib/A%20Comparison%20of%20Sentiment%20Analysis%20Techniques.pdf> [Accessed 29 Jan. 2021].
- Bahdanau, D., Cho, K. and Bengio, Y. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1409.0473>.
- Band, A. (2021). *Text2emotion: Python package to detect emotions from textual data*. [online] Medium. Available at: <https://towardsdatascience.com/text2emotion-python-package-to-detect-emotions-from-textual-data-b2e7b7ce1153>.
- Baron, J. (2017). *2017 Messenger Bot Landscape, a Public Spreadsheet Gathering 1000+ Messenger Bots | SAP Conversational AI Blog*. [online] SAP Conversational AI. Available at: <https://caitools.sap/blog/2017-messenger-bot-landscape/> [Accessed 2 Feb. 2021].

- Beale, R. (2019). *Introduction to HCI*. [online] Bham.ac.uk. Available at: <https://www.cs.bham.ac.uk/~rxb/Teaching/HCI%20II/intro.html>.
- Bhagwat, S. (2020). *Tensorflow vs Keras vs Pytorch: Which Framework is the Best?* [online] Medium. Available at: <https://medium.com/@AtlasSystems/tensorflow-vs-keras-vs-pytorch-which-framework-is-the-best-f92f95e11502> [Accessed 11 Apr. 2021].
- Bordes, A. and Weston, J. (2016). Learning end-to-end goal-oriented dialog. *CoRR*, [online] abs/1605.07683. Available at: <http://arxiv.org/abs/1605.07683>.
- British Computer Society (2001). *Standard for Software Component Testing*. [online] Available at: <http://www.testingstandards.co.uk/Component%20Testing.pdf>.
- Britz, D. (2016). *Deep Learning for Chatbots, Part 1 – Introduction*. [online] WildML. Available at: <http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/> [Accessed 29 Jan. 2021].
- Brownlee, J. (2018). *Difference Between a Batch and an Epoch in a Neural Network*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/#:~:text=The%20number%20of%20epochs%20is%20a%20hyperparameter%20that%20defines%20the> [Accessed 26 Apr. 2021].
- Brownlee, J. (2019a). *How to Control the Stability of Training Neural Networks With the Batch Size*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/>.
- Brownlee, J. (2019b). *Understand the Impact of Learning Rate on Neural Network Performance*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>.
- Cahn, J. (2017). *CHATBOT: Architecture, design, & development*. University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science.
- Cai, J., Li, J., Li, W. and Wang, J. (2018). Deeplearning Model Used in Text Classification. *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. [online] Available at: <https://ieeexplore.ieee.org/document/8632592/> [Accessed 20 Oct. 2019].
- Chen, H., Liu, X., Yin, D. and Tang, J. (2017). A Survey on Dialogue Systems. *ACM SIGKDD Explorations Newsletter*, 19(2), pp.25–35.
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H. and Bengio, Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. [online] . Available at: <https://arxiv.org/pdf/1406.1078.pdf>.
- Chua, D. (2021). *davidchua/pymessenger*. [online] GitHub. Available at: <https://github.com/davidchua/pymessenger> [Accessed 3 May 2021].
- Ciobanu, A. (2013). The Role of Student Services in the Improving of Student Experience in Higher Education. *Procedia - Social and Behavioral Sciences*, [online] 92, pp.169–173. Available at: <https://www.sciencedirect.com/science/article/pii/S1877042813027857>.

- CNN, N.L. (2020). *Robot friends: Why people talk to chatbots in times of trouble*. [online] CNN. Available at: <https://edition.cnn.com/2020/08/19/world/chatbot-social-anxiety-spc-intl/index.html> [Accessed 2 Feb. 2021].
- ConvAI3 (2018). *ConvAI3: Clarifying Questions for Open-Domain Dialogue Systems (ClariQ) by DeepPavlov*. [online] convai.io. Available at: <http://convai.io/data/>.
- Couper, M.P., Traugott, M.W. and Lamias, M.J. (2001). Web Survey Design and Administration. *Public Opinion Quarterly*, 65(2), pp.230–253.
- Csaky, R. (2019). Deep learning based chatbot models. *CoRR*, [online] abs/1908.08835. Available at: <http://arxiv.org/abs/1908.08835>.
- da Silva, E.A.B. and Mendonça, G.V. (2005). Digital Image Processing. *The Electrical Engineering Handbook*, pp.891–910.
- Dahiya, M. (2017). A tool of conversation: Chatbot. *International Journal of Computer Sciences and Engineering*, 5, pp.158–161.
- Danescu-Niculescu-Mizil, C. and Lee, L. (2011). Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In: *roceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.
- de Jesus, A. (2019). *Chatbots for Mental Health and Therapy – Comparing 5 Current Apps and Use Cases / Emerj*. [online] Emerj. Available at: <https://emerj.com/ai-application-comparisons/chatbots-mental-health-therapy-comparing-5-current-apps-use-cases/>.
- DeepAI (2019a). *Epoch*. [online] DeepAI. Available at: <https://deepai.org/machine-learning-glossary-and-terms/epoch>.
- DeepAI (2019b). *Hidden Layer*. [online] DeepAI. Available at: <https://deepai.org/machine-learning-glossary-and-terms/hidden-layer-machine-learning#:~:text=In%20neural%20networks%2C%20a%20hidden>.
- Desmarais, C. (2018). *Chatbot Validation Methods*. [online] Medium. Available at: <https://medium.com/ibm-watson/chatbot-validation-methods-8930737aeb2d> [Accessed 29 Apr. 2021].
- Dettmers, T. (2018). *A Full Hardware Guide to Deep Learning*. [online] Tim Dettmers. Available at: <https://timdettmers.com/2018/12/16/deep-learning-hardware-guide/#GPU> [Accessed 1 Jan. 2021].
- Developer Economics (2020). *State of the Developer Nation*. [online] www.developereconomics.com. Available at: <https://www.developereconomics.com/resources/reports/state-of-the-developer-nation-q3-20201> [Accessed 11 Apr. 2021].
- Djaja, F. (2018). *Build Facebook Messenger Contextual ChatBot with TensorFlow and Keras*. [online] Medium. Available at: <https://medium.com/@ferrygunawan/build-facebook-messenger-contextual-chatbot-with-tensorflow-and-keras-4f8cc79438cf>.
- Dong, H. (2021). *tensorlayer/seq2seq-chatbot*. [online] GitHub. Available at: <https://github.com/tensorlayer/seq2seq-chatbot> [Accessed 11 Apr. 2021].
- Dua, A. (2019). *Rule-Based Bots or AI Bots? Which One Do You Prefer? - Dataconomy*. [online] Dataconomy. Available at: <https://dataconomy.com/2019/05/rule-based-bots-or-ai-bots-which-one-do-you-prefer/>.

- Duan, Q. (2021). *qhduan/Seq2Seq\_Chatbot\_QA*. [online] GitHub. Available at: [https://github.com/qhduan/Seq2Seq\\_Chatbot\\_QA](https://github.com/qhduan/Seq2Seq_Chatbot_QA) [Accessed 11 Apr. 2021].
- Eckhardt, K. (2018). *Choosing the right Hyperparameters for a simple LSTM using Keras*. [online] Medium. Available at: <https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046>.
- Essilfie, W. (2017). *Building Facebook Messenger Bots with Python in less than 60 minutes*. [online] Twilio Blog. Available at: <https://www.twilio.com/blog/2017/12/facebook-messenger-bot-python.html> [Accessed 3 May 2021].
- Facebook Developers (2021a). *Authorization - Marketing API - Documentation*. [online] Facebook for Developers. Available at: <https://developers.facebook.com/docs/marketing-api/overview/authorization/> [Accessed 3 May 2021].
- Facebook Developers (2021b). *Messenger Solutions*. [online] Facebook for Developers. Available at: <https://developers.facebook.com/products/messenger/solutions/>.
- Facebook for Business (2016). *More Than a Message: The Evolution of Conversation*. [online] Facebook IQ. Available at: <https://www.facebook.com/business/news/insights/more-than-a-message-the-evolution-of-conversation> [Accessed 2 May 2021].
- Facebook for Business. (2019). *F8 2019: Making It Easier for Businesses to Connect with Customers on Messenger*. [online] Available at: <https://www.facebook.com/business/news/f8-2019-making-it-easier-for-businesses-to-connect-with-customers-on-messenger> [Accessed 2 Feb. 2021].
- Facebook Research. (2017). *bAbI*. [online] Available at: <https://research.fb.com/downloads/babi/>.
- Fainchtein, L. (2020). *Generative vs Retrieval Based Chatbots: A Quick Guide*. [online] Medium. Available at: <https://blog.cloudboost.io/generative-vs-retrieval-based-chatbots-a-quick-guide-8d19edb1d645>.
- Flask (2010). *Welcome to Flask — Flask Documentation (1.1.x)*. [online] Palletsprojects.com. Available at: <https://flask.palletsprojects.com/en/1.1.x/>.
- Google (2019). *Google Forms - create and analyze surveys, for free*. [online] Google.com. Available at: <https://docs.google.com/forms/u/0/>.
- Google (2021). *Google Colaboratory*. [online] colab.research.google.com. Available at: <https://colab.research.google.com/notebooks/intro.ipynb>.
- Hakak, N.M., Mohd, M., Kirmani, M. and Mohd, M. (2017). Emotion analysis: A survey. *2017 International Conference on Computer, Communications and Electronics (Comptelix)*.
- Heroku (2020). *Cloud Application Platform / Heroku*. [online] Heroku.com. Available at: <https://www.heroku.com/>.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9, pp.1735–1780.
- Ivanov, N. (2021). *nicolas-ivanov/tf\_seq2seq\_chatbot*. [online] GitHub. Available at: [https://github.com/nicolas-ivanov/tf\\_seq2seq\\_chatbot](https://github.com/nicolas-ivanov/tf_seq2seq_chatbot) [Accessed 11 Apr. 2021].
- Jia, J. (2003). *The Study of the Application of a Keywords-based Chatbot System on the Teaching of Foreign Languages*. [online] . Available at: <https://arxiv.org/ftp/cs/papers/0310/0310018.pdf>.
- Joshi, Chaitanya K, Mi, F. and Faltings, B. (2017). *Personalization in goal-oriented dialog*. *arXiv preprint arXiv:1706.07503*.

- Jovancic, N. (2018). *8 Demographic Questions Every Quality Survey Needs to Contain*. [online] LeadQuizzes. Available at: <https://www.leadquizzes.com/blog/8-types-of-demographic-questions/>.
- Katrekar, A. (2019). *An Introduction to Sentiment Analysis*. [online] Globallogic. Available at: <https://www.globallogic.com/se/wp-content/uploads/2019/12/Introduction-to-Sentiment-Analysis.pdf>.
- Kenyon, C. (2019). *NVIDIA “Turing” Tesla T4 HPC Performance Benchmarks*. [online] Microway. Available at: <https://www.microway.com/hpc-tech-tips/nvidia-turing-tesla-t4-hpc-performance-benchmarks/> [Accessed 29 Apr. 2021].
- Keras Documentation (2021). *Save and load Keras models / TensorFlow Core*. [online] TensorFlow. Available at: [https://www.tensorflow.org/guide/keras/save\\_and\\_serialize](https://www.tensorflow.org/guide/keras/save_and_serialize).
- keras-team (2020). *keras-team/keras*. [online] GitHub. Available at: <https://github.com/keras-team/keras>.
- Kim, S. (2021). *Seq2Seq Diagram*. *soofware/seq2seq*. Available at: <https://camo.githubusercontent.com/fe3df3ef6c0fb05767a373ef8e547120a0a667c423a5c52f3f202156b8aba1e5/68747470733a2f2f696e6372656469626c652e61692f6173736574732f696d616765732f736571327365712d736571327365715f74732e706e67> [Accessed 10 Apr. 2021].
- Kubrick, S. (1965). *The Kubrick Site: 2001 Screenplay (1965) Part Two*. [online] www.visual-memory.co.uk. Available at: <http://www.visual-memory.co.uk/amk/doc/0057b.html> [Accessed 29 Apr. 2021].
- Kukačka, J., Golkov, V. and Cremers, D. (2017). *Regularization for Deep Learning: A Taxonomy*. [online] . Available at: <https://arxiv.org/pdf/1710.10686.pdf> [Accessed 3 May 2021].
- Kumar, H. (2018). *Loss vs Accuracy*. [online] kharshit.github.io. Available at: <https://kharshit.github.io/blog/2018/12/07/loss-vs-accuracy> [Accessed 28 Apr. 2021].
- LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature*, 521(7553), pp.436–444.
- Li, J., Galley, M., Brockett, C., Spithourakis, G.P., Gao, J. and Dolan, B. (2016). A Persona-Based Neural Conversation Model. *arXiv:1603.06155 [cs]*. [online] Available at: <https://arxiv.org/abs/1603.06155> [Accessed 16 Feb. 2021].
- Li, S. (2018). *Named Entity Recognition with NLTK and SpaCy*. [online] Medium. Available at: <https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da>.
- Liu, B. (2012). Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologies*, 5(1), pp.1–167.
- Loughborough University Library (2019). *Questionnaire Design*. [online] lboro.ac.uk. Available at: <https://www.lboro.ac.uk/media/wwwlboroacuk/content/library/downloads/advisesheets/Questionnaire%20Design.pdf>.
- Loye, G. (2019). *Attention Mechanism*. [online] FloydHub Blog. Available at: <https://blog.floydhub.com/attention-mechanism/>.
- Ludwig, O. (2018). *Would you please add the attention or pointer mechanism based on your current model? · Issue #13 · oswaldoludwig/Seq2seq-Chatbot-for-Keras*. [online] GitHub. Available at: <https://github.com/oswaldoludwig/Seq2seq-Chatbot-for-Keras/issues/13> [Accessed 25 Apr. 2021].

- Luong, M.-T., Pham, H. and Manning, C.D. (2015). *Effective Approaches to Attention-based Neural Machine Translation*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1508.04025>.
- Mahmood, H. (2018). *Softmax Function, Simplified*. [online] Medium. Available at: <https://towardsdatascience.com/softmax-function-simplified-714068bf8156>.
- Manideep, V. (2020). *Chat Bot using Seq2Seq Model*. [online] Medium. Available at: <https://medium.com/@vvsmanideep/interview-chat-bot-using-seq2seq-model-fe9059ffe64> [Accessed 10 Apr. 2021].
- Mieczkowski, O. (2021). *Flask vs Django – which one to choose and when?* [online] www.monterail.com. Available at: <https://www.monterail.com/blog/flask-vs-django> [Accessed 3 May 2021].
- Mikheev, A., Moens, M., Grover, C., Moens@ed, M. and Uk (1999). *Named Entity Recognition without Gazetteers*. [online] . Available at: <https://www.aclweb.org/anthology/E99-1001.pdf> [Accessed 3 May 2021].
- Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1301.3781>.
- Nelson, D. (2020). *What Is Synthetic Data?* [online] Unite.AI. Available at: <https://www.unite.ai/what-is-synthetic-data/> [Accessed 3 May 2021].
- Nguyen, A. (2019). *15 Best Chatbot Datasets for Machine Learning | Lionbridge AI*. [online] Lionbridge AI. Available at: <https://lionbridge.ai/datasets/15-best-chatbot-datasets-for-machine-learning/>.
- Nguyen, T. and Shcherbakov, M. (2018). A Neural Network based Vietnamese Chatbot. *2018 International Conference on System Modeling & Advancement in Research Trends (SMART)*.
- NLTK (2009). *Natural Language Toolkit — NLTK 3.4.4 documentation*. [online] Nltk.org. Available at: <https://www.nltk.org/>.
- Olah, C. (2015). *Understanding LSTM Networks -- colah's blog*. [online] Github.io. Available at: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- Oppermann, A. (2020). *What is Deep Learning and How does it work?* [online] Medium. Available at: <https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac>.
- Oriel, A. (2020). *Top 10 Chatbot Datasets Assisting in ML and NLP Projects*. [online] Analytics Insight. Available at: <https://www.analyticsinsight.net/top-10-chatbot-datasets-assisting-in-ml-and-nlp-projects/> [Accessed 30 Apr. 2021].
- Patel, N.P., Parikh, D.R., Patel, D.A. and Patel, R.R. (2019). AI and Web-Based Human-Like Interactive University Chatbot (UNIBOT). *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/8822176> [Accessed 25 Dec. 2019].
- Peltarion (2020). *Categorical crossentropy loss function | Peltarion Platform*. [online] Peltarion.com. Available at: <https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy>.
- Pennington, J., Socher, R. and Manning, C.D. (2014). GloVe: Global vectors for word representation. [online] pp.1532–1543. Available at: <http://www.aclweb.org/anthology/D14-1162>.

Pew Research Center (2011). *Questionnaire design - Pew Research Center Methods*. [online] Pew Research Center Methods. Available at: <https://www.pewresearch.org/methods/u-s-survey-research/questionnaire-design/>.

PlaidML (2021). *plaidml/plaidml*. [online] GitHub. Available at: <https://github.com/plaidml/plaidml> [Accessed 27 Apr. 2021].

Prickett, B., Traylor, A. and Pater, J. (2018). Seq2Seq Models with Dropout can Learn Generalizable Reduplication. *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*.

Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P. (2016). *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1606.05250>.

Ramamoorthy, S. (2016). *Chatbots with Seq2Seq*. [online] suriyadeepan.github.io. Available at: <https://suriyadeepan.github.io/2016-06-28-easy-seq2seq/> [Accessed 29 Jan. 2021].

Ramamoorthy, S. (2021). *suriyadeepan/practical\_seq2seq*. [online] GitHub. Available at: [https://github.com/suriyadeepan/practical\\_seq2seq](https://github.com/suriyadeepan/practical_seq2seq) [Accessed 11 Apr. 2021].

Ranoliya, B.R., Nidhi Raghuvanshi and Sanjay Singh (2017). Chatbot for university related FAQs. In: *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*. Institute of Electrical and Electronics Engineers Inc., pp.1525–1530.

ROCM Core Technology (2021). *RadeonOpenCompute/ROCM*. [online] GitHub. Available at: <https://github.com/RadeonOpenCompute/ROCM> [Accessed 27 Apr. 2021].

Sansonnet, J.-P., Leray, D. and Martin, J.-C. (2006). Architecture of a Framework for Generic Assisting Conversational Agents. *Intelligent Virtual Agents*, pp.145–156.

Santoso, H.A., Anisa Sri Winarsih, N., Mulyanto, E., Wilujeng saraswati, G., Enggar Sukmana, S., Rustad, S., Syaiful Rohman, M., Nugraha, A. and Firdausillah, F. (2018). Dinus Intelligent Assistance (DINA) Chatbot for University Admission Services. *2018 International Seminar on Application for Technology of Information and Communication*.

Sarkar, D. (2018). *Emotion and Sentiment Analysis: A Practitioner's Guide to NLP*. [online] KDnuggets. Available at: <https://www.kdnuggets.com/2018/08/emotion-sentiment-analysis-practitioners-guide-nlp-5.html>.

Sayantini (2018). *Keras vs TensorFlow vs PyTorch | Deep Learning Frameworks*. [online] Edureka. Available at: <https://www.edureka.co/blog/keras-vs-tensorflow-vs-pytorch/> [Accessed 11 Apr. 2021].

Schlicht, M. (2016). *The Complete Beginner's Guide To Chatbots*. [online] Chatbots Magazine. Available at: <https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca>.

Sharma, S. (2017). *Activation Functions in Neural Networks*. [online] Medium. Available at: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>.

Sharma, S.R. (2020). *ShivamSharma26/text2emotion*. [online] GitHub. Available at: <https://github.com/ShivamSharma26/text2emotion> [Accessed 1 May 2021].

Shawar, B.A. and Atwell, E. (2007). Chatbots: are they really useful? pp.29–49.

- Shen, K. (2018). *Effect of batch size on training dynamics*. [online] Medium. Available at: <https://medium.com/minidistill/effect-of-batch-size-on-training-dynamics-21c14f7a716e>.
- Srivastava, N., Hinton, G., Krizhevsky, A. and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, [online] 15, pp.1929–1958. Available at: <https://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf>.
- Terra, J. (2020). *Keras vs Tensorflow vs Pytorch: Popular Deep Learning Frameworks*. [online] Simplilearn.com. Available at: <https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article>.
- The Access Platform (2021). *Product / The Access Platform*. [online] www.theaccessplatform.com. Available at: <https://www.theaccessplatform.com/product> [Accessed 24 Mar. 2021].
- Vosoglou, C. (2019). *What is the best programming language for Machine Learning?* [online] Medium. Available at: <https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7>.
- Wen, T.-H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L.M., Su, P.-H., Ultes, S. and Young, S. (2017). A Network-based End-to-End Trainable Task-oriented Dialogue System. *arXiv:1604.04562 [cs, stat]*. [online] Available at: <https://arxiv.org/abs/1604.04562> [Accessed 16 Feb. 2021].
- Wilpon, J.G. (1995). Voice-processing technologies--their application in telecommunications. *Proceedings of the National Academy of Sciences of the United States of America*, [online] 92(22), pp.9991–9998. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC40724/> [Accessed 3 May 2021].
- Wu, Y.-K. (2021). *ywk991112/pytorch-chatbot*. [online] GitHub. Available at: <https://github.com/ywk991112/pytorch-chatbot> [Accessed 11 Apr. 2021].
- Wuensch, K. (2015). *What is a Likert Scale and How Do You Pronounce Likert?* [online] Ecu.edu. Available at: <http://core.ecu.edu/psyc/wuenschk/StatHelp/Likert.htm>.
- Yam, C.-Y. (2015). *Emotion Detection and Recognition from Text Using Deep Learning*. [online] CSE Developer Blog. Available at: <https://devblogs.microsoft.com/cse/2015/11/29/emotion-detection-and-recognition-from-text-using-deep-learning/>.
- Yin, Z., Chang, K. and Zhang, R. (2017). Deepprobe: Information directed sequence understanding and chatbot design via recurrent neural networks. pp.2131–2139.
- Zhao, T., Lu, A., Lee, K. and Eskenazi, M. (2017). Generative Encoder-Decoder Models for Task-Oriented Spoken Dialog Systems with Chatting Capability. *arXiv:1706.08476 [cs]*. [online] Available at: <https://arxiv.org/abs/1706.08476> [Accessed 16 Feb. 2021].
- Zhou, H., Huang, M., Zhang, T., Zhu, X. and Liu, B. (2018). Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory. *arXiv:1704.01074 [cs]*. [online] Available at: <https://arxiv.org/abs/1704.01074> [Accessed 16 Feb. 2021].

## Appendices

Item 1 (Interview Participant Information Sheet)



# Edge Hill University

The Department of Computer Science

## CIS3140 Research and Development Project

### Interview Participant Information

Development of a Machine Learning Based Chatbot  
to Provide Emotional and Informational Welfare  
Support for Edge Hill University Students

Lewis Cunliffe – 23926643

# Supervisor – Sam Zhang

## Contents

<a href="#"><u>Introduction</u></a> .....	112
<a href="#"><u>Study Title</u></a> .....	112
<a href="#"><u>Principal Researcher</u></a> .....	112
<a href="#"><u>Study Information</u></a> .....	112
<a href="#"><u>Interview Information</u></a> .....	114

## Introduction

This document is an informative report for participants of the project. Below are various details regarding the project and your involvement.

### Study Title

The study you would participate in is titled:

"Development of a Machine Learning Based Chatbot to Provide Emotional and Informational Welfare Support for Edge Hill University Students".

### Principal Researcher

The principal researcher for the project is **Lewis Cunliffe**, who can be contacted via email address [23926643@edgehill.ac.uk](mailto:23926643@edgehill.ac.uk).

The supervisor for the project is **Dr. Huaizhong Zhang** who can be contacted via email address [huaihong.zhang@edgehill.ac.uk](mailto:huaihong.zhang@edgehill.ac.uk).

### Study Information

Before participating, it is important you understand the purpose of the involvement and what it entails. Please read this information carefully and discuss with others if necessary. If there are any questions you may have or if the information appears unclear, please contact at the listed emails for support.

The purpose of this study is to gather information regarding the "Chat with our students" support system Edgehill university uses for use in development of a project to provide an automated chatbot support system for emotional and informational support to students. The similarity between the existing service and the proposed is being used to gather information to support development. Note that currently support has been categorised as either informational (such as directions) and emotional (such as helping with stress) when referred to in this project. The study is intending to create an alternative system that allows for constant and instant support to students without need for human involvement, to avoid the flaws this brings. This research project will create such a system to provide students with this level of support. The study is an undergraduate project wherein research regarding the technical and user requirements will be completed.

This study is a one-to-one interview with a representative of the current "Chat with our students" system and its workers, to gain information regarding their experience at this position. The experience is what makes you suitable to be interviewed. You will be the only participant during this interview.

Once you have read this document, you have full control over whether or not you participate. Consent will be gained in both a signed format on the separate consent form as well as verbally before the interview begins. You are able to rescind consent at any time and consent will only be considered if it is informed, unambiguous and freely given. It must be a positive indication of agreement. Consent will be recorded full.

*At Edge Hill, we are committed to respecting and protecting your personal information. To find ways in which we use your data, please see [edgehill.ac.uk/about/legal/privacy](http://edgehill.ac.uk/about/legal/privacy).*

The University is committed to ensuring compliance with current data protection legislation and confirms that all data collected is used fairly, stored safely, and not disclosed to any other person unlawfully. The University is a data controller and, in some instances, may be a data processor of this data.

To withdraw consent, please contact via the email supplied. If withdrawn after the interview is conducted, all reference to the interview taking place will be erased and removed from the project and instead replaced with a short explanation that the interview was rescinded. If withdrawn during a stage of the interview, the interview will be cut short, and all existing material will be erased and removed. A short explanation of the interview will be present. If withdrawn, any reference to the interview will be strictly anonymous.

The interview will be confidential based on your own requirements. Before beginning the interview, you will be asked whether you wish for certain details to be recorded or not (name). These details are not integral to the study and so will be avoided but certain extra steps can be requested, such as editing audio to hide your voice or referring to your name differently. According to current knowledge of the interview, no questions that appear harmful will be asked but if you disagree, you are well within rights to ask that the question be skipped or redone to fit your requirements. Files regarding the interview such as transcripts will be stored securely with limited access to assure confidentiality as well.

The interview will be transcribed and then summarised. Both the transcription and summarisation will be included in a report regarding development and the points taken from the interview will be used to make certain decisions and reinforce some points during development. The other aspects of the interview, such as audio files and consent forms will be stored for evidence and submitted to prove that the interview occurred but not used as openly.

The Edge Hill Computing Department Ethics Committee will review the study.

After examination, there are no currently identified risks or disadvantages to taking part but if you do experience any adverse effects, contact the details above to discuss support.

For participation, you will be given a gift as appreciation for taking the time to help.

If you have any additional contact needs, please contact **Ardhendu Behera**, programme leader for computer science at [beheraa@edgehill.ac.uk](mailto:beheraa@edgehill.ac.uk). Any concerns you wish to present independent of the study should be addressed to them.

## Interview Information

The interview will be conducted via your own chosen medium (face to face, online) and will be semi structured with a few topics to cover but largely free form. The interview will strictly only discuss your role with the “Chat with our students” system with a few demographic questions to handle how I can address you. Below is a list of topics that will be covered.

Please contact me with any questions or recommendations regarding this.

- Introduction - I will talk about the interview, introducing it and its purpose. Verbal consent will be asked for here as well as how I can address you.
- Closed questions regarding your role.
  - What course you studied / study at the time of working for the system
  - What hours you worked there
  - How much you were paid.
  - How often you would be requested for support.
  - What times you would provide support
  - How quickly you would respond.
  - What year were most of the students you spoke to in.
- Open questions topics
  - Categorisation of support
  - Common topics of support sessions
  - Potential issues encountered with students.
  - Language used during conversation (slang, formal, etc)
  - Own issues with the service
  - Feedback for the service
  - Discussion about availability and access for support
  - Course specificity
  - Importance of human interaction

Questions asked will be thoroughly planned beforehand. Any issues or questions you do not wish to discuss, please communicate either during interview or beforehand via email. Discussion about support sessions will be kept confidential and unspecific to students you have talked to.

Once complete, the interview will be closed by me and certain processing (transcription, etc) will be done. I will email you all processed data and ask for verification on all documents. If at any stage you want for certain things said to be removed, do so.

**Item 2 (Participant Consent Form (Unsigned))****Participant Consent Form**

---

**Project title: Development of a Machine Learning Based Chatbot to Provide Emotional and Informational Welfare Support for Edge Hill University Students**

**Researcher: Lewis Cunliffe (23926643)**

- I confirm that I have read and understand the information sheet for the above study
- I have been given the opportunity to ask questions
- I agree to take part in this study
- I understand that my participation is voluntary and that I am free to withdraw for up to four weeks after the date of the consent.

Name of participant .....

Signature .....

Date .....

Researcher Lewis Cunliffe

Signature 

Date 28/02/2021

## Item 3 (Participant Consent Form (Signed for Interview))

**Edge Hill  
University**

**Participant Consent Form**

**Project title: Development of a Machine Learning Based Chatbot to Provide Emotional and Informational Welfare Support for Edge Hill University Students**

**Researcher: Lewis Cunliffe (23926643)**

- I confirm that I have read and understand the information sheet for the above study
- I have been given the opportunity to ask questions
- I agree to take part in this study
- I understand that my participation is voluntary and that I am free to withdraw for up to four weeks after the date of the consent.

Name of participant

Alanna Anderson.....

Signature

.....  
Alanna Anderson.....

Date

.....  
03 4/03/2021.....

Researcher

Lewis Cunliffe

Signature

.....  
L.C.

Date

28/02/2021

## Item 4 (Interview Participant Email Discussion)

**Project Interview - Lewis Cunliffe**

Reply | Forward

ALANNA ANDERSON Sun 28/02/2021 19:23  
To: LEWIS CUNLIFFE  
Hi Lewis,

I'd be more than willing to help you with your project.  
I'm free on most days apart from Fridays between 4pm-7pm due to classes.

If you require any other information feel free to email me and I'd be more than happy to help.

Thank you for considering me for this.  
Alanna

Get [Outlook for Android](#)

LEWIS CUNLIFFE Sun 28/02/2021 19:14  
To: ALANNA ANDERSON  
Hi Alanna,

I would like to get in touch with you to ask if you would be open to an interview about your role with the "Chat with our students" service Edge Hill University provides. I am currently working on a project to create an automatic support system with some similarity to it and would benefit from knowing about the system. If you are interested, please get back to me and I will begin scheduling the interview. If you have any questions please feel free to contact me as well. I would really appreciate it if you gave your time to help me and would compensate your time with a small gift. Please do not feel obliged or pressured to agree, I understand how busy we all are.

Kind Regards,  
Lewis Cunliffe

**Project Interview - Lewis Cunliffe**

Reply | Forward

LEWIS CUNLIFFE Sun 28/02/2021 19:23  
To: ALANNA ANDERSON  
Hi Alanna,

That's great, thank you so much for agreeing. So that you are aware, you are more than welcome to change your mind at any time. I have attached an information booklet and consent form to this email regarding the interview. Please read them carefully so that you can understand what the interview entails. If you have any questions or concerns, feel free to contact this email or those I have included in the information sheet.

In terms of holding the interview, would you prefer to communicate face to face, online or over the phone?  
For the date and time, would 3pm on Thursday 4th of March be available? I am currently unsure of how long the interview will take place so I believe giving half an hour in your schedule for it would be suitable to avoid any conflict.  
As well, are there any preliminary points regarding the interview such as topics to avoid? Please read the attached documents and get back to me.

Thanks again for this,  
Lewis Cunliffe

ALANNA ANDERSON Sun 28/02/2021 19:23  
To: LEWIS CUNLIFFE  
Hi Lewis,

I'd be more than willing to help you with your project.  
I'm free on most days apart from Fridays between 4pm-7pm due to classes.

If you require any other information feel free to email me and I'd be more than happy to help.

Thank you for considering me for this,  
Alanna

**Project Interview - Lewis Cunliffe**

Reply | Forward

LEWIS CUNLIFFE Mon 01/03/2021 17:33  
To: ALANNA ANDERSON  
Hi Alanna,

That's great, I will see you on the 4th then. Thanks again for participating, it will be a big help to my project.

Kind Regards,  
Lewis Cunliffe

ALANNA ANDERSON Mon 01/03/2021 17:32  
To: LEWIS CUNLIFFE  
Hi Lewis,

3pm on the March 4th works for me.

I'd prefer to communicate face to face if possible, if not, no worries.

I'll also contact you if I have any questions.

Thanks once again,  
Alanna

Get [Outlook for Android](#)

The screenshot shows the Microsoft Outlook inbox for 'Lewis Cunliffe' at 'Edge Hill University'. The inbox contains several messages:

- A message from 'do-not-reply@mail.quickly.tools' with subject 'Interview Evaluation - Lewis Cunliffe' (Thu 29/03/2021 14:48). It contains a link to 'Interview Result...'.
- A message from 'ALANNA ANDERSON' (Thu 29/03/2021 14:48) responding to the previous message.
- A message from 'LINDSAY DONNELLY - Donnell@edgehill.ac.uk' (Thu 11/02) with subject 'ORG\_STCN\_FAS\_59 (ALL\_YEARS...)'.
- A message from 'Cortana' (Thu 06/13) with subject 'Your daily briefing'.
- A message from 'SALLY-ANNE LEET - Leets@edgehill.ac.uk' (Wed 16/03) with subject 'ORG\_STCN\_FAS\_59 (ALL\_YEARS...)'.
- A message from 'HARI PANDEY - HariPandey@edgehill.ac.uk' (Tue 15/03) with subject '2020\_CIS3157 (COMPLEX SYST...)'.

The message from Alanna Anderson is selected and displayed in the preview pane:

**Interview Evaluation - Lewis Cunliffe**

ALANNA ANDERSON  
Thu 29/03/2021 14:48  
To: LEWIS CUNLIFFE  
Hi Lewis,

Thank you for the transcript,  
I've read through it and I agree to what you have written is correct. There is nothing I wish to remove.

Thank you,  
Alanna

[Get Outlook for Android](#)

**Reply** | **Forward**

**LEWIS CUNLIFFE**  
Wed 24/03/2021 20:16  
To: ALANNA ANDERSON  
Interview Results.docx  
63 KB

Hi Alanna,

Thanks again for participating in the interview, really appreciate the help. I just wanted to get in touch with you to give you a transcript of the interview as well as some points I have taken from it. Would you be able look over them and make sure there's nothing you would like to remove or any points you disagree with?

Thanks again,  
Lewis Cunliffe

## Item 5 (Interview Transcript)

P1 - Interviewee  
 LC - Interviewer  
 INTERVIEW BEGIN  
 LC - Okay, this is the interview for my chat bot project. I, uh, just before we got the interview started, I just want to check that you have given consent to be here, to do the interview. Have you signed the consent form?  
 P1 - Yeah i've signed the consent and given consent  
 LC - And verbal consent as well?  
 P1 - Yeah.  
 LC - Yeah. Great. Uh, how do you want me to refer to you in this interview and for the sake of confidentiality? You may not want me to use your name or anything like that  
 P1 - for confidentiality you could put person one  
 LC - person one that sounds good. Uh, I'll refer - When you use your name and say and stuff. Um, so just want to give you some information on the project itself. I'm making a chat bot, which is like an automated chat support system. It's going to be catered towards university students. Um, you are currently just to just a preliminary question, you are, you worked for the chat with our students system.  
 P1 - Yeah. I still do  
 LC - You still do? Great. Um, it's a similar system to that. They get in touch, as I'm sure you've read them the information booklet. Yeah?  
 P1 - Yeah. Yeah.  
 LC - Great. Um, get in touch, just talk about different issues and whatever. Okay. Uh, and um, this interview is confidential to a certain extent, obviously, because you've chosen to use the name person one.  
 P1 - Yeah.  
 LC - Um, I'll keep it at that. And it's going to be used in the report. The transcript chances are no one's ever going to hear this, uh, this actual voice recording. But if that wasn't the case, would you like me to edit it? Your voice? So it's discernible.  
 P1 - No, I'm okay with it staying the same.  
 LC - Okay great. And, um, any questions you don't feel comfortable asking, answering, or you don't actually know the answer to just say, and we'll move on. I won't be asking any questions about specific conversations that you've had with people through the system. But I would like to discuss at like a general scope  
 P1 - Yeah  
 LC - If you get what I mean. Is that okay?  
 P1 - That's fine.  
 LC - Great. And, um, after the interview, I'm going to write a transcript of what we say. I'll send it to you. As well as a summary of the transcript, like just bullet points of what points are made and what I can use for my own project. I'll send them to you and with your approval of what was said. Feel free to say I don't really think that when I said that I was I - dunno  
 P1 - Yeah  
 LC - I want to change what was said essentially. Um, yeah, you're more than welcome to do that. Okay. So the first stage is, um, these are the closed questions, kind of simple. Uh, they're very specific and almost definitive answers. There's a few times that depending on the answer, I might lead up with a few things just for you to go into detail, but it's mostly just general yes, no questions or just, you give an estimate of like a number. So, um, We've already kind of done this, but I just wanted to verify that you'd worked for the chat with our student system.  
 P1 - Yeah.  
 LC - And you still work there?  
 P1 - Yes, I do.  
 LC - Great. And are you paid for it?  
 P1 - I am. Yes.  
 LC - Would you - can you say how much?  
 P1 - Uh, it depends on my hours, but if I work like one or two hours, I get 20 pounds a month.  
 LC - Okay great. And that's one or two hours a month or a week?  
 P1 - Uh, just over the month.  
 LC - Over the month. So it's about an hour, 10 pounds.  
 P1 - Yeah.  
 LC - Yeah. Um, and how many hours do you usually spend working with it?  
 P1 - Yeah, it depends because if there's an event on, then we obviously get more questions across. But if there's like, if it's just a general day, I don't really get messages on it.  
 LC - Uh, do you have like an estimate of the hours? Sorry, just so,  
 P1 - Yeah, that's alright. And it probably be about the same one less. Yeah. One or two hours, unless there's no event. And then there's like three or four because yeah.  
 LC - I see yeah, more conversations. Okay. Um, when did these hours usually take place? Know that might be a strange question if it's like just smaller conversations, like once, maybe every week or.  
 P1 - Yeah. So I usually mid-week Wednesday, Thursdays, I got messages. Yeah. But if it's after I was in, I can't reply.  
 LC - I see. Um, what are your hours? Is it like nine till five?  
 P1 - It's 9:00 AM until 9:00 PM. And then after that, we don't have to reply until the next day.  
 LC - Do you have the option to reply?  
 P1 - Yeah.  
 LC - Do you find that you got a lot of questions outside of hours?  
 P1 - I do. If there, like foreign students in different timezones  
 LC - I see. Yeah. That makes sense. Um, and how quickly do you usually take to respond to the app? Like on average  
 P1 - If it's during work hours, it's like within the hour, but if it's not it's later the next morning.  
 LC - Yeah. And, uh, so during this interview, I'm going to talk about support in two categories, uh, informational which is like, uh, can you help me find a location on campus and emotional, maybe someone's talking to you about stress or loneliness, that kind of thing. Would you agree that it can be split into those two categories?  
 P1 - Yeah.  
 LC - Yeah? Great. That makes things quite easy. Um, of the two categories. Did you have one that you found your most dealing with?  
 P1 - I'm most dealing with university life and the course in general. So that would be information about the course.  
 LC - Great. Um, and do you are currently studying, if you don't mind disclosing -  
 P1 - computing and security, networking and forensics,  
 LC - Great. Uh, and, uh, do you find that a lot of these support sessions are related to the course?  
 P1 - Yeah. Yeah.  
 LC - Would you say the majority are?  
 P1 - Yeah.  
 LC - Yeah. And the next topic, I know you mentioned university life, sorry. Could you go into more detail into what that is?  
 P1 - So it's, um, what the tech hub actually contains compute as teachers. How has the social life, is there anywhere on campus? That would be recommended to go to, to hang out if you need to chill out time. Is there an area where it's quiet or is there anywhere in town I'd recommend how's the transport.  
 LC - Okay. And then I'm assuming a lot of this is based on students that are speaking to you before they come to uni.  
 P1 - Yes.  
 LC - Do you only deal with students like that, or is it sometimes that you deal with students that are current students?  
 P1 - It's mainly like, year 13, so second year of colleges, but I do get first years going. Can you help me with this  
 LC - Mainly just first years, ever get any second years?  
 P1 - Yeah. Yeah.  
 LC - And how long do these conversations? Usually last,  
 P1 - It depends on the topic, but they can last like. I had one that lasted 15 minutes. It was just back and forth. Yeah.  
 LC - Uh, uh, they had to like a list of questions, I see. Okay. That was great. Um, I just want to get into the second stage, which is a bit more open. And, uh, again, if you want to say that, I don't really want to answer it, or I can't think of an answer. Just feel free. The first question is, um, it's more just a general question for you, but if you were to use a chat support system, Would you prefer to use it via, let's say the university website, via messenger like Facebook messenger, like you'd add the chat bot as a friend almost, or something that you download yourself  
 P1 - I personally think it would be better linked to the university because then you'd know that it wasn't like a spam link.  
 LC - I see. Yeah. You'd know that it's affiliated. Yeah. If let's say the other two were to have some sort of certification or some approval -  
 P1 - Yeah -  
 LC - Which would you say would be best  
 P1 - If you could prove that it like links on Facebook messenger I guess that would be a good one to choose as well because most people do have Facebook and university could link it.  
 LC - Right. And, um, can you just summarize for me what you did? I know we've already discussed it quite heavily, but what you did in the chat with our students' role.

P1 - So this is basically I, sorry, students send me questions about whatever they want and I can, like, I respond with either hyperlinks say if it's what, in these courses, what do you recommend? Or I write out full messages. And say, well, there's this point, this point, this point, there's like different support systems, but we're only allowed to talk about the university  
LC - So going back, sorry to the two categories. Would you find that the current system isn't really suited to emotional support almost? Obviously, if it was emotional support, let's say I'm stressed about my course, but if it was emotional, say some issues, about the friends or something. You wouldn't be able to handle that?

P1 - Not allowed to get involved in the students' lives. Yeah

LC - that makes sense. Okay. Is there anything else you'd like to add to the summarisation before I interrupted you?

P1 - Um, no, I think it's just, yeah.

LC - Yeah. A lot of it we've already spoke about it.

P1 - Yeah.

LC - So, um, Other than the text conversation between you and the student - I know you mentioned about hyperlinks. Is there any other features, if you could go into detail on the hyperlinks as well, just a secondary feature, maybe a tracker on the side, just to keep track of conversations you may have with this specific student?

P1 - Well, if you need outside research, the app I use for work is The Access Platform. So you can view it online. It's an actual thing. And it's like, sort of like Facebook, we can post pictures and comments and we can look. And

LC -- I know it's almost got like an FAQ section as well, doesn't it? And you kind of respond to whatever questions.

P1 - Yeah

LC - Great. And, um, so when you were dealing with, I've already asked which one you dealt with most of the two categories?. And we've already discussed that you didn't necessarily have to deal with as many emotions because of course there are restrictions. And, um, did you potentially have to not deal with any emotion, anything that you would categorize yourself as emotional or were there some times when you did?

P1 - I can, if they're stressed about some, I can recommend the talk to somebody, but say it was. They want to drop out or they're feeling certain emotions like, yeah. Um, I have to involve my boss, so I'd have to join her into the conversation and go look - I can't deal with, yes.

LC - It needs further help. Yeah. That makes sense. And that of the two categories, which would you say are the most, do you struggle with the most between the informational and emotional?

P1 - Probably emotional because the information's easier to find. And emotions are different for everyone.

LC - Oh yeah, definitely. Yeah. That makes sense. Are there any like specific methods you employed to deal with either of the categories? So let's say for informational, you had to follow like a, almost a script or for emotional, like follow certain steps if you get what I mean?

P1 - Yeah, we don't have that. We just have to make sure it's professional. So we can't just like fob them off and be like, well, there's this, you have to like give at least some detail.

LC - I got it. Yeah. And, um, when you were discussing with people, did any of them express dissatisfaction? Did anyone say this is bad or whatever?

P1 - I've not had that come up, no.

LC - Has anyone ever mentioned any features, during the conversation that they think that the chat bots should have?

P1 - Not from the students talking to me.

LC - Okay. Uh, was there someone that has? Not necessarily from students that have spoke with you maybe from all the people you work with, your manager or anything like that?

P1 - We had a meeting and it was like, what features would we want added. So at the moment, The workers can't talk to each other. So it's basically, we have to add them into the conversation to talk to them. And that's not professional if we just have a side conversation. Yeah.

LC - And, uh, any features, other than that, that you feel like you personally would want added?

P1 - No, it's just that it needs ,like,it would be good if there was a way I could say, Hey, do you know about this topic? Because there's this question come through.

LC - Yeah. Um, and you mentioned earlier that you had to get your manager involved, don't want to go specifically into details as I'm sure it was confidential. Was that the only situation? Was it only the one or were there quite, was that quite a common occurrence?

P1 - No i've only had that one

LC - And, um, if you had someone that was reusing the system quite a bit, what did you have someone to keep track of what they have previously maybe mentioned or would it be purely memory?

P1 - Yeah. In the app we have like a message section. So we've got people's names, just first names cause confidentiality. And we can like see all the past conversations with them. And when they're finished talking to us, like we closed the conversation, but they can reopen it if they have more questions. So everything they've previously said comes up as an asset. So it's like, it saves their information. So it's easier to track.

LC - Okay, great. Uh, just to close. This is the last thing that I'm going to do, uh, before I actually close it and just summarize. So from my, uh, you've read through the information booklet, uh, for my project, I'm hoping to create an alternative because personally, I believe that there's certain benefits over having an automated system over a human, like privacy, constant being able to message and there's no exhaustion on your end. Do you have any feelings about that? Like, uh, the second thing that you have to say, maybe the, uh, through the information booklet of what I've said is potentially wrong, or you agree with it, or -

P1 - would it be human manned as well? Or would it be an AI learning?

LC - It'd be AI learning.

P1 - Okay Yeah. So you'd have it start off with basic questions

LC -- it would be trained on an existing data set. Um, Just go a little bit into detail. What I am planning on doing is training on a general dataset for human conversation. This is the current plan. It may change, um, and using a thing called the knowledge base, which is essentially, let's say - current literature we studied taught about using wave a, uh, restaurant booking service. And when they mentioned a location, it takes out the location name. Let's say Liverpool. And changes that to almost a, a, it's just like a snippet of the text and to replace it or whatever, just kind of drags out and essentially applies some rules to it. And, um, that's the core approach we'll take up. So there'll be essentially, there'll be two separate data sets one general on one on edgehill chill related with certain things like a current, my plan is to try and get it set up. So if someone asks like where is the catalyst? It gives directions to it, or maybe even more generally sends them a link to the map. And it's a bit more personally, I feel that with human support, you can be more specific and go into more detail but at the same time, I feel like there's some benefits to having in general, open, super quick, way to get responses. Yes. Would you agree with that?

P1 - yeah, that makes sense. Because then you don't have to sit there at a computer constantly.

LC - Okay. Yeah, absolutely. Well so, uh, the interview is pretty much finished. Um, any questions before, I close up?

P1 - No, I have no questions.

LC - That's great. I just want to thank you for your time. I really appreciate this. It's a big help and, uh, hopefully I'll. Help with my project. Um, I gave you the gift that we discussed.

P1 - Yeah. Um, yeah, I've got it.

LC - And for just going back to what I was going to say about what happens next, I'm going to transcribe this and then summarize at the bottom. Just some general points that I will take from the interview. I'll send it to you. If you've got any, issues with it. If there's anything from it that I've got out of it, that you don't agree with it just send me a message and I'll make sure to do that. Uh, just want to thank you again. Like I said, I really appreciate the time that you've given me and, uh, Any questions again? I know I've just asked that, but -

P1 - no.

LC - Great. And the recording now, hopefully it's-

INTERVIEW END

## Item 6 (Initial Survey Design)

The screenshot shows the Google Forms interface for designing a survey. It consists of three vertically stacked sections, each with a title, description, and a list of questions or options.

**Section 1 of 7:**

**University Chatbot Support Survey - Lewis Cunliffe (23926643)**

I am running a survey on current Edge Hill students use of the university support systems, specifically the "Chat with our students" service.

The data gathered will be used in my project to develop an automated chatbot support system for Edge Hill University students. Your response regarding your interaction with the current system will be important to understand the user requirements of this system and how it should compare to the existing. The information gathered will be anonymous and only used in statistical context with no identifying factors to attribute responses to individuals.

The survey is made up of various question formats. For multiple choice questions, click the circle that applies most. For text questions, type your answer in the box labelled "Your answer". For dropdown questions, click the arrow to open the list and select your answer. For rating scale questions, click the circle at the point on the scale that best describes your answer. The first question will be asked and 7 choices given. These choices usually scale between a very negative response at 1 to a neutral response at 4 and a very positive response at 7. Please chose the value you believe applies to the statement. Many questions are mandatory and marked with a red asterisk (\*) to show this. Those that aren't are usually dependent on previous questions as stated in the question itself. Please leave these blank if they do not apply.

This survey will take around five minutes. Thank you for participating and if you have any questions or feed please contact me at 23926643@edgehill.ac.uk.

**Section 2 of 7:**

**Demographic and Introductory Questions**

Description (optional)

What best describes your gender? \*

Female  
 Male  
 Prefer not to answer  
 Other...

What course are you currently taking or have taken? \*

Short-answer text

How old are you? \*

1. 17 or below  
2. 18  
3. 19

**Section 2 of 7:**

**Demographic and Introductory Questions**

Description (optional)

What best describes your gender? \*

Female  
 Male  
 Prefer not to answer  
 Other...

What course are you currently taking or have taken? \*

Short-answer text

How old are you? \*

1. 17 or below  
2. 18  
3. 19

3. 19  
4. 20  
5. 21  
6. 23  
7. 24 or above  
8. Prefer not to answer

For my project, I have defined two categories of support. These are informational (such as support regarding directions to locations on campus) and emotional (such as support regarding concerns like stress). Would you agree with this categorisation? \*

Yes  
 No

If not, can you please describe how you would categorise types of support and explain any reasoning?

Long-answer text

Have you ever used the "Chat with our students" service Edge Hill University provides? \*

3. 19  
4. 20  
5. 21  
6. 23  
7. 24 or above  
8. Prefer not to answer

For my project, I have defined two categories of support. These are informational (such as support regarding directions to locations on campus) and emotional (such as support regarding concerns like stress). Would you agree with this categorisation? \*

Yes  
 No

If not, can you please describe how you would categorise types of support and explain any reasoning?

Long-answer text

Have you ever used the "Chat with our students" service Edge Hill University provides? \*

Have you ever used the "Chat with our students" service Edge Hill University provides? \*

Yes  
 No

After section 2 Continue to next section

Section 3 of 7

### "Chat with our students" Non-Use Questions

Description (optional)

Were you aware of the existence of the service before taking this survey? \*

Yes  
 No

If not, would you have used the service? (chat with our students is a discussion service where you can ask specific students questions about Edge Hill University)

If not, would you have used the service? (chat with our students is a discussion service where you can ask specific students questions about Edge Hill University)

Yes  
 No

After section 3 Go to section 5 (Feature and Access...Opinion Questions) ▾

Section 4 of 7

### "Chat with our students" Use Questions

Description (optional)

How many times have you used the service? \*

1. 1 time
2. 2 - 3 times
3. 4 - 5 times
4. 5 - 10 times
5. 11 or more times

What time of your study did you find you used the service most? \*

1. Before First Year
2. First Year
3. Second Year
4. Third Year
5. Post Graduate

Referring to the previous categorisations, which do you believe you used the service for more? \*  
 If you do not agree with the categorisations, please use the other section to describe what you found you used to the service for

Informational  
 Emotional  
 Other...

After section 4 Continue to next section ▾

Section 5 of 7

### Feature and Access Opinion Questions

Description (optional)

How willing would you be to access a chat support service for Edge Hill university via their website? \*

1	2	3	4	5	6	7	
Very Unwilling	<input type="radio"/>	Very Willing					

How willing would you be to access a chat support service for Edge Hill university via an existing IM service (such as facebook messenger)? \*

1	2	3	4	5	6	7	
Very Unwilling	<input type="radio"/>	Very Willing					

How willing would you be to access a chat support service for Edge Hill university via independent downloadable software? \*

1	2	3	4	5	6	7	
Very Unwilling	<input type="radio"/>	Very Willing					

How willing would you be to access a chat support service for Edge Hill university via independent downloadable software? \*

1 2 3 4 5 6 7  
Very Unwilling      Very Willing

How important is having the chat service be accessible to both mobile phones and PC's to you? \*

1 2 3 4 5 6 7  
Very unimportant      Very important

How important are accessibility features for you? \*

1 2 3 4 5 6 7  
Very unimportant      Very important

Other than basic text communication with support, are there any other features you feel are important to providing help (Accessibility or otherwise)?  
Long-answer text

Section 6 of 7

### Support Opinion Statements

Description (optional)

The "Chat with our students" service is easily accessible \*

1 2 3 4 5 6 7  
Completely Disagree      Completely Agree

The "Chat with our students" service is useful \*

1 2 3 4 5 6 7  
Completely Disagree      Completely Agree

I prefer to discuss issues rather than simply being given the necessary information \*

1 2 3 4 5 6 7  
Completely Disagree      Completely Agree

I have no privacy concerns about chatting to other students using the "Chat with our students" service \*

1 2 3 4 5 6 7  
Completely Disagree      Completely Agree

I am comfortable discussing emotional issues with another student using the "Chat with our students" service \*

1 2 3 4 5 6 7  
Completely Disagree      Completely Agree

I am comfortable discussing informational issues with another student using the "Chat with our students" service \*

1 2 3 4 5 6 7  
Completely Disagree      Completely Agree

Having to wait for responses from support is not an issue for me \*

1 2 3 4 5 6 7  
Completely Disagree      Completely Agree

The screenshot shows a survey interface with three questions and a navigation bar.

**Question 1:** I would prefer to stay anonymous when looking for support \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

**Question 2:** 24/7 availability of support is important to me \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

**Question 3:** A chatbots ability to provide support would be more of a priority to me than its ability to convincingly simulate human speech \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

After section 6 Continue to next section

Section 7 of 7

**Survey Complete**

Thankyou for completing the survey. If you have any questions please contact me at 23926643@ed.ac.uk and leave any feedback or notes below. Remember to submit the form via the button marked "Submit".

Please enter any additional notes such as feedback here

Long-answer text

## Item 7 (Initial Survey)

**University Chatbot Support Survey -  
Lewis Cunliffe (23926643)**

I am running a survey on current Edge Hill students use of the university support systems, specifically the "Chat with our students" service.

The data gathered will be used in my project to develop an automated chatbot support system for Edge Hill University Students. Your response regarding your interaction with the current system will be important to understand the user requirements of this system and how it should compare to the existing.

The information gathered will be anonymous and only used in statistical context with no identifying factors to attribute responses to individuals.

The survey is made up of various question formats. For multiple choice questions, click the circle that applies most. For text questions, type your answer in the box labelled "Your answer". For dropdown questions, click the box labelled "Choose" and select an answer from the potential responses. For scaling questions, a question or statement will be asked and 7 choices given. These choices usually scale between a very negative response at 1 to a neutral response at 4 and a very positive response at 7. Please choose the value you believe applies to the statement. Many questions are mandatory and marked with a red asterisk (\*) to show this. Those that aren't are usually dependent on previous questions as stated in the question itself. Please leave these blank if they do not apply.

This survey will take around five minutes. Thank you for participating and if you have any questions or feed please contact me at [23926643@edgehill.ac.uk](mailto:23926643@edgehill.ac.uk).

**Next**

Never submit passwords through Google Forms.  
This content is neither created nor endorsed by Google. [Report Abuse](#) · [Terms of Service](#) · [Privacy Policy](#)

Google Forms

**Demographic and Introductory Questions**

What best describes your gender? \*

Female  
 Male  
 Prefer not to answer  
 other: \_\_\_\_\_

What course are you currently taking or have taken? \*

Your answer \_\_\_\_\_

How old are you? \*

Choose

For my project, I have defined two categories of support. These are informational (such as support regarding directions to locations on campus) and emotional (such as support regarding concerns like stress). Would you agree with this categorisation? \*

Yes  
 No

If not, can you please describe how you would categorise types of support and explain any reasoning?

Your answer \_\_\_\_\_

Have you ever used the "Chat with our students" service Edge Hill University provides? \*

Yes  
 No

**Back** **Next**

Never submit passwords through Google Forms.  
This content is neither created nor endorsed by Google. [Report Abuse](#) · [Terms of Service](#) · [Privacy Policy](#)

Google Forms

**University Chatbot Support Survey -  
Lewis Cunliffe (23926643)**

\*Required

**"Chat with our students" Use Questions**

How many times have you used the service? \*

Choose

What time of your study did you find you used the service most? \*

Choose

Referring to the previous categorisations, which do you believe you used the service for more? If you do not agree with the categorisations, please use the other section to describe what you found you used to the service for \*

Informational  
 Emotional  
 Other: \_\_\_\_\_

**Back** **Next**

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. Report Abuse · Terms of Service · Privacy Policy

**University Chatbot Support Survey -  
Lewis Cunliffe (23926643)**

\*Required

**"Chat with our students" Non-Use Questions**

Were you aware of the existence of the service before taking this survey? \*

Yes  
 No

If not, would you have used the service? (chat with our students is a discussion service where you can ask specific students questions about Edge Hill University)

Yes  
 No

**Back** **Next**

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. Report Abuse · Terms of Service · Privacy Policy

Google Forms

**University Chatbot Support Survey -  
Lewis Cunliffe (23926643)**

\*Required

**Feature and Access Opinion Questions**

How willing would you be to access a chat support service for Edge Hill university via their website? \*

1 2 3 4 5 6 7  
Very Unwilling       Very Willing

How willing would you be to access a chat support service for Edge Hill university via an existing IM service (such as facebook messenger)? \*

1 2 3 4 5 6 7  
Very Unwilling       Very Willing

How willing would you be to access a chat support service for Edge Hill university via independent downloadable software? \*

1 2 3 4 5 6 7  
Very Unwilling       Very Willing

via independent downloadable software? \*

1 2 3 4 5 6 7

Very Unwilling        Very Willing

How important is having the chat service be accessible to both mobile phones and PC's to you? \*

1 2 3 4 5 6 7

Very unimportant        Very important

How important are accessibility features for you? \*

1 2 3 4 5 6 7

Very unimportant        Very important

Other than basic text communication with support, are there any other features you feel are important to providing help (Accessibility or otherwise) ?

Your answer

[Back](#) [Next](#)

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) [Terms of Service](#) [Privacy Policy](#)

Google Forms

### University Chatbot Support Survey - Lewis Cunliffe (23926643)

\*Required

#### Support Opinion Statements

The "Chat with our students" service is easily accessible \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

The "Chat with our students" service is useful \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

I prefer to discuss issues rather than simply being given the necessary information \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

I have no privacy concerns about chatting to other students using the "Chat with our students" service \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

I am comfortable discussing emotional issues with another student using the "Chat with our students" service \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

I am comfortable discussing informational issues with another student using the "Chat with our students" service \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

Having to wait for responses from support is not an issue for me \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

I would prefer to stay anonymous when looking for support \*

1 2 3 4 5 6 7

Completely Disagree        Completely Agree

1 2 3 4 5 6 7  
Completely Disagree       Completely Agree

24/7 availability of support is important to me \*

1 2 3 4 5 6 7  
Completely Disagree       Completely Agree

A chatbots ability to provide support would be more of a priority to me than its ability to convincingly simulate human speech \*

1 2 3 4 5 6 7  
Completely Disagree       Completely Agree

**Back** **Next**

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) · [Terms of Service](#) · [Privacy Policy](#)

Google Forms

**University Chatbot Support Survey -  
Lewis Cunliffe (23926643)**

**Survey Complete**

Thankyou for completing the survey. If you have any questions please contact me at [23926643@ed.ac.uk](mailto:23926643@ed.ac.uk) and leave any feedback or notes below. Remember to submit the form via the button marked 'Submit'.

Please enter any additional notes such as feedback here

Your answer

**Back** **Submit**

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) · [Terms of Service](#) · [Privacy Policy](#)

Google Forms

## Item 8 (Pilot A Conversation)

The screenshots show three consecutive days of email往来 between Lewis Cunliffe (LC) and Michael Harper (MH).

**Screenshot 1 (Fri 19/02):**

- From:** MICHAEL HARPER; THOMAS JEFFERIES  
**To:** MICHAEL HARPER; THOMAS JEFFERIES
- Subject:** Project Survey Pilot Testing - Lewis Cunliffe
- Message:**

Hi,  
I wanted to get in contact with you to ask a favour. I'm currently doing a project and am planning a survey to discuss the universities current "Chat with our students" system to help find data to use in development of my own similar system, a chatbot that would provide support to students as well.

I was hoping you would be able to pilot a survey I have designed before I distribute it. The results are completely confidential at this stage, only the feedback you provide will be used.

Please get back to me if you would like to participate.

Many thanks,  
Lewis Cunliffe

**Screenshot 2 (Sat 20/02):**

- From:** MICHAEL HARPER; THOMAS JEFFERIES  
**To:** MICHAEL HARPER; THOMAS JEFFERIES
- Subject:** Project Survey Pilot Testing - Lewis Cunliffe
- Message:**

Hi,

I wanted to get in contact with you to ask a favour: I'm currently doing a project and am planning a survey to discuss the universities current "Chat with our students" system to help find data to use in development of my own similar system, a chatbot that would provide support to students as well.

I was hoping you would be able to pilot a survey I have designed before I distribute it. The results are completely confidential at this stage, only the feedback you provide will be used.

Please get back to me if you would like to participate.

Many thanks,  
Lewis Cunliffe

**Screenshot 3 (Mon 22/02):**

- From:** MICHAEL HARPER; THOMAS JEFFERIES  
**To:** MICHAEL HARPER; THOMAS JEFFERIES
- Subject:** Project Survey Pilot Testing - Lewis Cunliffe
- Message:**

Hi Michael,

Thank you for taking the time to offer feedback for the survey, I really appreciate the help and I'll make sure to work on your criticism.

Kind Regards,  
Lewis Cunliffe

**Inbox Summary:**

- Folders:** Favourites, Inbox, Sent Items, Drafts, Deleted Items, Junk Email, Archive, Notes.
- Messages:**
  - Sent Items:** MICHAEL HARPER; THOMAS JEFFERIES (Project Survey Pilot Testing - Lewis Cunliffe, Fri 19/02)
  - Inbox:** Huaizhong Zhang (Project Discussion - Lewis Cunliffe, Fri 19/02), MACAULEY BAKER; LUKE MACPHERSON (new group, Fri 19/02)
  - Deleted Items:** adphil-careers@targetconnect.net (Only a few days to go - EHU Career..., 10/15)
  - Junk Email:** Cortana (Your daily briefing, 06/15)
  - Archive:** ALANNA ANDERSON (Project Interview - Lewis Cunliffe, Sun 19/02)
  - Notes:** Edge Hill University (Important update about in perso..., Fri 16/02)

## Item 9 (Pilot B Conversation)

**Edge Hill University Outlook**

**Sent Items**

**Project Survey Pilot Testing - Lewis Cunliffe**

LEWIS CUNLIFFE  
Set 27/02/2021 18:29  
To: MICHAEL HARPER; THOMAS JEFFERIES

Hi, I wanted to get in contact with you to a...

MICHAEL HARPER; THOMAS JEFFERIES  
Project Survey Pilot Testing - Lewis ... 18:29  
Hi Sam, That works great for me. Thanks, L...

Huajizhong Zhang  
Project Discussion - Lewis Cun... Fri 19/02  
Hi Sam, That works great for me. Thanks, L...

MACAULEY BAKER: LUKE MACPHERSON  
new group Fri 19/02  
My usernames Lewis#9997

**Inbox**

**Focused**

**Project Survey Pilot Testing - Lewis Cunliffe**

edgeli-careers@targetconnect.net  
Only a few days to go – EHU Career... 10:15  
CAUTION: This email originated from outside...

THOMAS JEFFERIES: MICHAEL HAR... ↗  
Project Survey Pilot Testing - Lewis ... 08:29  
Hey Lewis, I just finished the survey and pu...

Cortana  
Your daily briefing 06:15  
Hi LEWIS CUNLIFFE, Off to a good start! C...

ALANNA ANDERSON  
Project Interview - Lewis Cun... Sun 19/23  
Hi Lewis, I'd be more than willing to help y...

Interview Conse... +1

Edge Hill University  
Important update about in perso... Fri 16:00  
CAUTION: This email originated from outside...

do-not-reply@blackboard.com  
Daily Notifications Fri 12:24

**Survey Complete**

Thankyou for completing the survey. If you have any questions please contact me at [23926643@edhill.ac.uk](mailto:23926643@edhill.ac.uk) and leave any feedback or notes below. Remember to submit the form via the button marked "Submit".

Please enter any additional notes such as feedback here

Some of the questions that relate directly to the use of the student support service could be marked as only necessary to students who've actually used it, or who answered yes to the question of if they've actually used it.

Submitted 01/03/2021, 08:27

## Item 10 (Pilot Changes)

University Support Survey All changes saved in Drive

**Questions Responses 1**

Can you effectively communicate issues you are having via text chat? Multiple choice

Yes  No  Add option or [Add "Other"](#)

If no, what format of communication would you be able to do so by? Required

Short-answer text

**Section 6 of 7 Questions Responses 3**

**Support Opinion Statements**

Description (optional)

If I were to use the "Chat with our students" service, I would find accessing it easy \* 1 2 3 4 5 6 7

Completely Disagree  Completely Agree

Chat support systems are useful \* 1 2 3 4 5 6 7

Completely Disagree  Completely Agree

I prefer to discuss issues rather than simply being given the necessary information \* 1 2 3 4 5 6 7

Completely Disagree  Completely Agree

University Support Survey All changes saved in Drive

**Questions Responses 3**

Discussing issues with other students via a chat support system would raise no privacy concerns 1 2 3 4 5 6 7

Completely Disagree  Completely Agree

I would be comfortable discussing emotional issues with other students via a chat support system 1 2 3 4 5 6 7

Completely Disagree  Completely Agree

I would be comfortable discussing informational issues with other students via a chat support system 1 2 3 4 5 6 7

Completely Disagree  Completely Agree

I would prefer to speak to a real human being rather than a bot, even if it could convincingly simulate human conversation and solve my issue to the same degree as the human Linear scale

## Item 11 (Distribution Email)

The screenshot shows a Microsoft Outlook interface with the following details:

- Subject:** Project Survey - Lewis Cunliffe
- To:** LEWIS CUNLIFFE
- Bcc:** LUKE MACPHERSON; JOSHUA HUYTON; MICHAEL HARPER; JUAN GUTIERREZ DE ROJAS; jsyme.beyle@hotmaill.co.uk; THOMAS JEFFERIES; LAUREN BLOOD; JAY DEVINE; LUKE WILLIMOTT; ALANNA ANDERSON
- Date:** Tue 02/03/2021 13:16
- Message Content:** A message from Lewis Cunliffe asking for participation in a survey about chat support systems.
- Attachment:** A screenshot of a survey form titled "University Chatbot Support Survey - Lewis Cunliffe (23926643)".
- Replies:** A reply from Alanna Anderson, a reply-all from Lewis Cunliffe, and a forward link.

Item 12 (User Testing Participant Information Sheet)



**Edge Hill University**

The Department of Computer Science

**CIS3140 Research and Development  
Project**

User Testing Participant Information

Development of a Machine Learning Based Chatbot  
to Provide Emotional and Informational Welfare  
Support for Edge Hill University Students

Lewis Cunliffe – 23926643

Supervisor – Sam Zhang

## Contents

Introduction .....	136
Study Title .....	136
Principal Researcher .....	136
Study Information .....	136
Testing Information .....	138

## Introduction

This document is an informative report for participants of the project. Below is various details regarding the project and your involvement.

### Study Title

The study you would participate in is titled:

"Development of a Machine Learning Based Chatbot to Provide Emotional and Informational Welfare Support for Edge Hill  
University Students"

### Principal Researcher

The principal researcher for the project is **Lewis Cunliffe**, who can be contacted via email address  
[23926643@edgehill.ac.uk](mailto:23926643@edgehill.ac.uk).

The supervisor for the project is **Dr. Huaizhong Zhang** who can be contacted via email address  
[huaizhong.zhang@edgehill.ac.uk](mailto:huaizhong.zhang@edgehill.ac.uk).

### Study Information

Before participating, it is important you understand the purpose of the involvement and what it entails. Please read this information carefully and discuss with others if necessary. If there is any questions you may have or if the information appears unclear, please contact at the listed emails for support.

The purpose of this study is to gather the reception to a developed system at various stages. The project aims to provide an automated chatbot support system for emotional and informational support to students. Testing at numerous stages of development using prototypes is conducted with a third party, you, to find other's impressions and feedback for the development process. Each stage varies. The first stage is a general chatbot where the performance of conversation simulation will be tested. The second stage incorporates emotional analysis into your input where the performance of this additional feature will be tested. The third stage incorporates data specific to Edge Hill University to provide support and the quality of this will be tested. The fourth stage incorporates this into a messenger app that will be tested to see your reception to this implementation. For each prototype, you will fill out a survey of your reception to the stage and its performance and your inputs into the system (as well as its outputs) will be recorded and monitored. The results of this testing will be used in development to contribute to understanding of how users find the system. Depending on the outcome of the testing, the stage testing may be repeated at points after changes have been made.

The testing requires you to use the system and then answer a brief survey regarding your experience. Your lack of involvement with the project and being a potential user (current student of Edge Hill University) make you a suitable candidate for completing this testing.

Once you have read this document, you have full control over whether or not you participate. Consent will be gained in both a signed format on the separate consent form as well as within each survey. You are able to rescind consent at any time and consent will only be considered if it is informed, unambiguous and freely given. It must be a positive indication of agreement. Consent will be recorded full.

*At Edge Hill, we are committed to respecting and protecting your personal information. To find ways in which we use your data, please see [edgehill.ac.uk/about/legal/privacy](http://edgehill.ac.uk/about/legal/privacy).*

The University is committed to ensuring compliance with current data protection legislation and confirms that all data collected is used fairly, stored safely, and not disclosed to any other person unlawfully. The University is a data controller and, in some instances, may be a data processor of this data.

To withdraw consent, please contact via the email supplied. If withdrawn after the testing is conducted, all reference to the interview taking place will be erased and removed from the project and instead replaced with a short explanation that the interview was rescinded. If withdrawn during the testing, the testing will be cut short and all existing material will be erased and removed. A short explanation of the withdrawal will be present. If withdrawn, any reference to the interview will be strictly anonymous.

The testing will be confidential based on your own requirements. Before beginning the testing you will be asked whether you wish for certain details to be recorded or not (name). These details are not integral to the. According to current knowledge of the survey, no questions that appear harmful will be asked but if you disagree, you are well within rights to ask that the survey be reconstructed to fit your requirements.

The testing will produce your inputs into the system prototypes, the outcome of that input and the survey results that you answer. This data will be analysed and evaluated to find conditions of the testing. Both the system data and survey results will be included in a report regarding development and the data from the testing will be used to control the development process and consider whether the system meets certain requirements. Analysis obtained from the data will be sent to you to gain approval and validation before use in the report.

The Edge Hill Computing Department Ethics Committee will review the study.

After examination, there are no currently identified risks or disadvantages to taking part but if you do experience any adverse effects, contact the details above to discuss support.

For participation, you will be given a gift as appreciation for taking the time to help.

If you have any additional contact needs, please contact **Ardhendu Behera**, programme leader for computer science at [beheraa@edgehill.ac.uk](mailto:beheraa@edgehill.ac.uk). Any concerns you wish to present independent of the study should be addressed to them.

### Testing Information

Testing will be organised via email. Some preliminary questions before testing will be asked to handle certain requests such as anonymity and verification that you have read this document. Testing will be conducted in four potentially repeating stages at various times. The prototype will be accessed on the development hardware and you will use the system. At each stage you will be asked to converse normally with the prototype with direction to test each stages implementation (e.g. at the sentiment analysis you may be requested to use emotional language). Use of the system will be monitored. Once used, a survey will be given to provide feedback. Each stages testing will have its own information recorded for you to validate. The times for testing will be organised via email. Survey questions will vary between the stages and will be developed after each stage's prototype is completed. Surveys will be conducted via google sheets.

**Item 13 (Participant Consent Form (User Testing Signed))**

Edge Hill  
University

**Participant Consent Form**

**Project title: Development of a Machine Learning Based Chatbot to Provide Emotional and Informational Welfare Support for Edge Hill University Students**

**Researcher: Lewis Cunliffe (23926643)**

- I confirm that I have read and understand the information sheet for the above study
- I have been given the opportunity to ask questions
- I agree to take part in this study
- I understand that my participation is voluntary and that I am free to withdraw for up to four weeks after the date of the consent.

Name of participant ..... *Mhamed Hanger*

Signature ..... *M Hanger*

Date ..... *26/04/2021*

Researcher ..... Lewis Cunliffe

Signature ..... *Lewis*

Date ..... *28/02/2021*

## Item 14 (User Testing Participant Conversation)

**Project User Testing - Lewis Cunliffe**

 LEWIS CUNLIFFE  
Sun 25/04/2021 14:28  
To: MICHAEL HARPER

 User Testing Participant Infor...  
52 KB

Hi Michael,

Thanks again for the pilot test earlier. I wanted to get in touch to ask if you would be interested in user testing the prototypes of my project. The project aims to create a chatbot for support of Edge Hill students and as a student yourself with no affiliation to my project, your feedback would be very appreciated and useful to test the stages of my development. I have attached the information sheet regarding this, please get back to me if you are interested in helping me so we can discuss the testing further. If you have any questions please feel free to contact me as well. I would really appreciate it if you gave your time to help me and would compensate your time with a small gift. Please do not feel obligated or pressured to agree, I understand how busy we all are.

Kind Regards,  
Lewis Cunliffe

[Reply](#) | [Forward](#)

**Project User Testing - Lewis Cunliffe**

 LEWIS CUNLIFFE  
Wed 28/04/2021 16:59  
To: MICHAEL HARPER

Hi Michael,

Thank you for agreeing to test my work, I really appreciate it. Would you be available to test the first stage tomorrow?

Kind Regards,  
Lewis Cunliffe

...

[Reply](#) | [Forward](#)

 MICHAEL HARPER  
Mon 26/04/2021 20:19  
To: LEWIS CUNLIFFE

hi Lewis,

Yes, I would be interested in user testing the prototypes.

sincerely  
Michael

...

**Project User Testing - Lewis Cunliffe**

 LEWIS CUNLIFFE  
Sat 01/05/2021 20:22  
To: MICHAEL HARPER

Hi Michael,

Thank you for conducting the testing for both stage 2 and 3 today, it has been a massive help so far. I really appreciate the help given. Would you be available tomorrow for the last stage of testing? I will have your compensation for the testing as a gift for participating. I really appreciate the help.

Thanks again,  
Lewis Cunliffe

...

[Reply](#) | [Forward](#)

 MICHAEL HARPER  
Fri 30/04/2021 07:54  
To: LEWIS CUNLIFFE

Hi Lewis,

Yes I will be available tomorrow.

regards,  
MICHAEL

...

 MICHAEL HARPER  
Sun 02/05/2021 07:25  
To: LEWIS CUNLIFFE

Hi Lewis,

Yes I will be available tomorrow to carry out the last testing stage.

Regards,  
MICHAEL

...

Are the suggestions above helpful? Yes No

The image contains two side-by-side screenshots of an email conversation. The left screenshot shows Lewis Cunliffe's message to Michael Harper. It includes a small profile picture, the recipient's name, the date and time (Mon 03/05/2021 13:01), and the subject 'Testing Results.docx' (3 MB). The message body starts with 'Hi Michael,' followed by a note of appreciation for participation in the user testing. The right screenshot shows Michael Harper's response. It includes his profile picture, the date and time (Mon 03/05/2021 13:13), and the subject 'Lewis Cunliffe'. The message body begins with 'Hi Lewis,' and expresses agreement with the test results.

Note that the testing results sent are the result and discussion included in this report.

This screenshot shows a single email window with a reply message. The subject is 'User Testing Results - Lewis Cunliffe'. The message is from Michael Harper, dated Mon 03/05/2021 13:13, to Lewis Cunliffe. The body of the message reads: 'yes those results seem to be correct.' Below the message, there is a standard email footer with 'Reply' and 'Forward' buttons.

### Item 15 (Stage 1 User Testing)

User Testing - Stage 1

<https://docs.google.com/forms/u/0/d/1-ShMgdHgo5ZLHA59zedpsi6k...>

## User Testing - Stage 1

This is the survey for stage 1 of user testing. Please enter the relevant information regarding the user testing of the prototype below. This is supervised, so feel free to ask any questions.

Do you give consent to continue user testing? \*

Yes

No

### User Testing - Stage 1 Feedback

Was the chatbot satisfactory? \*

Yes

No

Did you encounter any errors? (Note that poor results are not considered errors, only when the system fails completely) \*

Yes

No

User Testing - Stage 1

<https://docs.google.com/forms/u/0/d/1-ShMgdHgo5ZLHA59zedpsi6k...>

Please detail the error encountered (if applicable)

---

How convincing was the conversation simulation?

1    2    3    4    5    6    7    8    9    10

Least Convincing



Most Convincing

Please summarise your feelings about the chatbot at this stage

The chatbot demonstrated capability of understanding what I was saying. However, the responses it formulated were mostly nonsense in response to its understanding.

---

This content is neither created nor endorsed by Google.

Google Forms

## Item 16 (Stage 2 User Testing)

User Testing - Stage 2

<https://docs.google.com/forms/u/0/d/18Nhoi-U87yDa9Y-ZCqwcu8bk...>

### User Testing - Stage 2

This is the survey for stage 2 of user testing. Please enter the relevant information regarding the user testing of the prototype below. This is supervised, so feel free to ask any questions.

Do you give consent to continue user testing? \*

- Yes  
 No

User Testing - Stage 1 Feedback

Was the chatbot satisfactory? \*

- Yes  
 No

Did you encounter any errors? (Note that poor results are not considered errors, only when the system fails completely) \*

- Yes  
 No

User Testing - Stage 2

<https://docs.google.com/forms/u/0/d/18Nhoi-U87yDa9Y-ZCqwcu8bk...>

Please detail the error encountered (if applicable)

---

How convincing was the conversation simulation?

1    2    3    4    5    6    7    8    9    10

Least Convincing



Most Convincing

How aware of your emotions was the chatbot?

1    2    3    4    5    6    7    8    9    10

Least Aware



Very Aware

Please summarise your feelings about the chatbot at this stage

Chatbot seemed to respond to some of the emotions and conversation was much better this stage

This content is neither created nor endorsed by Google.

Google Forms

### Item 17 (Stage 3 User Testing)

User Testing - Stage 3

<https://docs.google.com/forms/u/0/d/15jShpsjefGM0VTUTCtUeHX2...>

## User Testing - Stage 3

This is the survey for stage 3 of user testing. Please enter the relevant information regarding the user testing of the prototype below. This is supervised, so feel free to ask any questions.

Do you give consent to continue user testing? \*

- Yes  
 No

User Testing - Stage 3 Feedback

Was the chatbot satisfactory? \*

- Yes  
 No

Did you encounter any errors? (Note that poor results are not considered errors, only when the system fails completely) \*

- Yes  
 No

User Testing - Stage 3

<https://docs.google.com/forms/u/0/d/15jShpsjefGM0VTUTCtUeHX2...>

Please detail the error encountered (if applicable)

---

How convincing was the conversation simulation?

1    2    3    4    5    6    7    8    9    10

Least Convincing



Most Convincing

How aware of your emotions was the chatbot?

1    2    3    4    5    6    7    8    9    10

Least Aware



Very Aware

Did the chatbot offer sufficient emotional support?

Yes

No

Did the chatbot offer sufficient informational support?

Yes

No

User Testing - Stage 3

<https://docs.google.com/forms/u/0/d/15jShpsjefGM0VTUTCtUeHX2...>

Please summarise your feelings about the chatbot at this stage

The chatbot offered some support and understood many of my emotions. At times it assumed sadness when it should have been anger though. Still the chatbot seemed quite aware of my emotions and seemed to actively simulate conversation.

This content is neither created nor endorsed by Google.

Google Forms

## Item 18 (Stage 4 User Testing)

User Testing - Stage 4

<https://docs.google.com/forms/u/0/d/1MyMw3AFWGcfrWOa0IQ4Oc...>

### User Testing - Stage 4

This is the survey for stage 3 of user testing. Please enter the relevant information regarding the user testing of the prototype below. This is supervised, so feel free to ask any questions.

Do you give consent to continue user testing? \*

- Yes  
 No

User Testing - Stage 4 Feedback

Was the chatbot satisfactory? \*

- Yes  
 No

Did you encounter any errors? (Note that poor results are not considered errors, only when the system fails completely) \*

- Yes  
 No

User Testing - Stage 4

<https://docs.google.com/forms/u/0/d/1MyMw3AFWGcfrWOa0IQ4Oc...>

Please detail the error encountered (if applicable)

---

How convincing was the conversation simulation?

1    2    3    4    5    6    7    8    9    10

Least Convincing



Most Convincing

How aware of your emotions was the chatbot?

1    2    3    4    5    6    7    8    9    10

Least Aware



Very Aware

Did the chatbot offer sufficient emotional support?

Yes

No

Did the chatbot offer sufficient informational support?

Yes

No

User Testing - Stage 4

<https://docs.google.com/forms/u/0/d/1MyMw3AFWGcfrWOa0IQ4Oc...>

Was the chatbot accessible through facebook messenger?

- Yes  
 No

Did the chatbot respond to your input in adequate time?

- Yes, instantly  
 Yes, but it took some time still  
 No, the time taken was too long  
 There was no response

Please summarise your feelings about the chatbot at this stage

This stage was the most accurate representation of the chatbot to me because I didnt have to use a field like informational information or emotional information. I think because of this, the quality drop a little but it was still much better than in the first stage. The use of messenger was really good. The speed of responses was good as well for the majority but a few messages hung. This was still under 30 seconds so it wasnt an issue. The handling of emojis and bad inputs was really useful as well

Until now you have been asked to test the chatbot as a concept of use that could be scaled to be implementable in a support environment. At this stage, do you believe that the current chatbot is implementable in a support environment and would you use it? Please give your reasons why

No, the chatbot isn't really useable for any gain as the results are still inconsistent even if they do show that they can provide emotional and informational support, its not good enough to be used as an actual tool at this stage. I do agree that it shows potential though, all the features are clearly there but the inconsistency means I wouldn't ever use it to help myself.

This content is neither created nor endorsed by Google.

## Item 19 (First Attempt Code - Expanded Insufficient Dataset)

```

!/usr/bin/env python
# coding: utf-8

# In[1]:
#Imports
import numpy as np # Imports numpy for handling numeric data in arrays
import pickle # Imports pickle for saving and loading small data
import os # Imports os for file handling
import os.path # Imports os.path for directory manipulation
import h5py # Imports pickle for saving and loading large data
import codecs # Used to handle the dataset
import io # Used to read the dataset data
import re # Used to clean the text via regular expressions
import zipfile # Package to handle zipped files, specific for the downloaded dataset
import requests # Package to request from websites, used to download the dataset
from tensorflow.keras import Input, Model, models # Import for keras input into models, the model itself and models for loading data
from tensorflow.keras.activations import softmax # Import for softmax activation function
from tensorflow.keras.layers import Embedding, LSTM, Dense # Import for embedding, LSTM and dense layers
from tensorflow.keras.optimizers import RMSprop # Import for RMSProp automatic optimizer
from tensorflow.keras.preprocessing.sequence import pad_sequences # Import to pad sequences to reshape data
from tensorflow.keras.preprocessing.text import Tokenizer # Import for tokenizing inputs for use
from tensorflow.keras.utils import to_categorical # Import for one hot encoding data=


# In[2]:
class ChatbotBase(): # Class to hold information to both classes, acts as an abstract

    #Fields
    filterRegex = '!"#$%&()/*.-;/<>?@[\n]_{}|^\\n\\'0123456789' # Sets up regex characters to filter
    hiddenLayers = 232 # Defines unit amount for LSTM layers
    batchSize = 64 # Defines the batch size, for the model
    epochAmount = 300 # Defines the amount of epochs to use
    dataSize = 10000 # Defines the amount of data to take from the dataset

    modelPath = "training/trainingModel" # Defines the path to save and load the model

    dataFolderDirectory = "data/" # Defines a general directory for the data
    gDatasetUrl = 'http://www.cs.cornell.edu/~cristian/data/cornell_movie_dialogs_corpus.zip' # Defines the dataset url
    gDatasetFile = 'data/cornell movie-dialogs corpus/movie_lines.txt' # Defines the dataset file directory

    beginningOfSentence = "bos " # Defines the beginning of text marker (Bos used instead of start for example to avoid accidental classifications)
    endOfSentence = " eos" # Defines the end of text marker

    def PickleSave(item, fileName): # Function to save items with picckle
        with open(f'{ChatbotBase.dataFolderDirectory}{fileName}.pickle', 'wb') as f: # Opens directory
            pickle.dump(item, f, protocol=pickle.HIGHEST_PROTOCOL) # Saves item

    def PickleOpen(item, fileName): # Function to open items with picckle
        with open(f'{ChatbotBase.dataFolderDirectory}{fileName}.pickle', 'rb') as f: # Opens directory
            item = pickle.load(f) # loads item
            return item # Returns item

    def HDF5Save(item, fileName): # Function to save items with HDFS
        with h5py.File(f'{ChatbotBase.dataFolderDirectory}{fileName}.hdf5', 'w') as f: # Opens directory
            f.create_dataset(fileName, data=item) # Saves item

    def HDF5Open(item, fileName): # Function to open items with HDFS
        with h5py.File(f'{ChatbotBase.dataFolderDirectory}{fileName}.hdf5', 'r') as f: # Opens directory
            item = f.get(fileName)[()] # loads item as numpy array
            return item # returns item

    def CleanText(text): # Function for cleaning text, removing unneccesary characters and making lower case

        if type(text) == dict: # Makes sure text is not a dictionary (Common error due to : use in text)
            keys = list(text) # Gets the key (note there should only be one )
            totalText = "" # Empty to hold the entire text
            for key in keys: # For each key
                value = text[key] # Finds the value associaaed
                text = f'{key} {value}' # Appends data back together without colon
                totalText = totalText + text # adds to total string
            text = totalText # makes text the total string

        text = text.lower() # Converts text input to lowercase

        text = re.sub(r"i'm", "i am", text) # Removes potentially conflicting punctuation, swapping with full text
        text = re.sub(r"he's", "he is", text)
        text = re.sub(r"she's", "she is", text)
        text = re.sub(r"it's", "it is", text)
        text = re.sub(r"that's", "that is", text)
        text = re.sub(r"what's", "that is", text)
        text = re.sub(r"where's", "where is", text)
        text = re.sub(r"how's", "how is", text)
        text = re.sub(r"\n", " will", text)
        text = re.sub(r"\ve", " have", text)
        text = re.sub(r"\re", " are", text)
        text = re.sub(r"\d", " would", text)
        text = re.sub(r"\re", " are", text)
        text = re.sub(r"\won't", "will not", text)
        text = re.sub(r"\can't", "cannot", text)
        text = re.sub(r"\n't", " not", text)
        text = re.sub(r"\n", "ng", text)
        text = re.sub(r"\bout", "about", text)
        text = re.sub(r"\til", "until", text)

```

```

text = re.sub(r"[-()\"#@;<>{}+=~|.!?]", "", text) # Removes conflicting characters completely
return text # Return cleaned text

def ProcessDataset():
    # Function to process dataset
    with codecs.open(ChatbotBase.gDatasetFile, "rb", encoding="utf-8", errors="ignore") as f: # Opens codec as filestream
        linesFull = f.read().split("\n") # Splits lines on new line and reads the file, getting the full line data (metadata included)
        linesSplit = [] # Creates empty list for lines that are split
        for line in linesFull: # For each line
            lineDataItems = line.split(" +++$+++" ) # Split based on dataset split marker, creating a list of different data items for each line such as id, character name and line text
            linesSplit.append(lineDataItems) # Add list of items to the list

    linesProcessed = {} # Creates dictionary for each line after processing
    for lineDataItems in linesSplit[ChatbotBase.dataSize]: # For each item list in the lines list
        if len(lineDataItems) > 4: # If the amount of items is greater than 4
            lineIndex = lineDataItems[0][1:] # Defines index by taking id item and reducing it to only numbers
            lineText = lineDataItems[4] # Defines lineText by taking the text of the line
            linesProcessed[int(lineIndex)] = lineText # Adds each index and line text to the dictionary where index is key and line text is its value
    linesProcessed = sorted(linesProcessed.items(), key = lambda x: x[0]) # Sorts items by id

    convDict = {} # Defines dict for conversations
    convIndex = 1 # Defines index for each conversation
    convCurrentLines = [] # Defines list for current conversation lines
    for i in range(1, len(linesProcessed)+1): # For loop going through all lines now sorted, keeping number as i (note that range is offset to allow for use of i)
        if i < len(linesProcessed): # If current line is not the second to last or last (keeps in range)
            if (linesProcessed[i][0] - linesProcessed[i-1][0]) == 1: # If the id of the current line minus the previous id in the list is 1 (Demonstratin correct ordering and part of conversation)
                if linesProcessed[i-1][1] not in convCurrentLines: # If the previous line is not stored stored in the list
                    convCurrentLines.append(linesProcessed[i-1][1]) # Adds previous line to the list
                convCurrentLines.append(linesProcessed[i][1]) # Adds current line to the list
            elif (linesProcessed[i][0] - linesProcessed[i-1][0]) > 1: # If the id difference is greater than one (demonstrating not part of conversation and that a new conversation has begun)
                convDict[convIndex] = convCurrentLines # Current lines are added to dictionary with key as index to show dialogue
                convCurrentLines = [] # Empties list to hold new current conversation lines
            convIndex += 1 # Increments conversation index to show that the next conversation is being added
        else:
            pass # Skips the last couple of items to avoid conflict

    queriesAndResponses = [] # List of queries and responses
    for conversation in convDict.values(): # For each conversation in dictionary
        if len(conversation) % 2 != 0: # If there are an odd number of queries and responses
            conversation = conversation[:-1] # Remove the extra
        for i in range(0, len(conversation), 2): # For each query in a conversation
            queriesAndResponses.append((conversation[i], conversation[i+1])) # Add the query and its response to the list as a tuple

    queries, responses = zip(*queriesAndResponses) # Unzips list putting the query and its response in seperate lists with indexes related
    queries = list(queries) # Assures list definition
    responses = list(responses) # Assures list definition

    cleanQueries = [] # Empty list for clean queries
    for query in queries: # For each query
        query = ChatbotBase.CleanText(query) # calls function for cleaning the query
        cleanQueries.append(query) # Adds query to list

    cleanResponses = [] # Empty list for clean responses
    for response in responses: # For each response
        response = ChatbotBase.CleanText(response) # calls function for cleaning the response
        cleanResponses.append(response) # Adds response to list

    taggedResponses = [ChatbotBase.beginningOfSentence + response + ChatbotBase.endOfSentence for response in cleanResponses] # Tags response with end and start via concatenation

    encoderInputs = cleanQueries # Defines inputs for encoder using the queries cleaned
    decoderInputs = taggedResponses # Defines inputs for decoder the

    fullText = np.concatenate((encoderInputs,decoderInputs)) # Joins the two lists as full text

    tokenizer = Tokenizer(filters=ChatbotBase.filterRegex) # Defines tokenizer with filter of incompatible regex character
    tokenizer.fit_on_texts(fullText) # Uses full text to make tokenizer

    wordIndex = tokenizer.word_index # Makes a dictionary of word index

    vocabSize = len(wordIndex) + 1 # Find the vocab size from the word index

    indexWord = {} # Defines index for index word, structured with index as key instead of word
    for word, index in wordIndex.items(): # For every word and index in word index
        indexWord[index] = word # Add to reversed index

    encoderSequences = tokenizer.texts_to_sequences(encoderInputs) # Converts encoder inputs to sequences using tokenizer
    decoderSequences = tokenizer.texts_to_sequences(decoderInputs) # Converts encoder inputs to sequences using tokenizer

    encoderMaxLength = max([len(sequence) for sequence in encoderSequences]) # Finds the max length of the encoder sequences
    encoderInputData = pad_sequences(encoderSequences, maxlen=encoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length

    decoderMaxLength = max([len(sequence) for sequence in decoderSequences]) # Finds the max length of the decoder sequences
    decoderInputData = pad_sequences(decoderSequences, maxlen=decoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length

    for i in range(len(decoderSequences)): # For each item in tokenized answers (Not encoder fit)
        decoderSequences[i] = decoderSequences[i][1:] # Remove start tag
    decoderOutputPadding = pad_sequences(decoderSequences, maxlen=decoderMaxLength, padding='post') # Pads answers
    decoderOutputData = to_categorical(decoderOutputPadding, vocabSize, dtype = 'b') # categorises data, making each sequence a list of lists where 0 represents nothing and 1 represents a word

    return wordIndex, indexWord, encoderInputData, decoderInputData, decoderOutputData # Returns data

def MakeChatbotModel(vocabSize): # Function to construct the chatbot model

```

```

# Encoder Layer Definitions
encoderInputLayer = Input(shape=(None,), name = "Encoder_Input_Layer") # Defines input layer with flexible shape
encoderEmbeddingLayer = Embedding(vocabSize, ChatbotBase.hiddenLayers, mask_zero=True, name = "Encoder_EMBEDDING_Layer") # Defines embedding layer, using the amount
of unique words, hidden layers defined and ignoring Os to create embedding matrix of entire words uses
encoderLSTMLayer = LSTM(ChatbotBase.hiddenLayers, return_state=True, name = "Encoder_LSTM_Layer") # Defines the LSTM of the encoder with use of hidden layers, and
returning values for use in the decoder
#Decoder Layer Definitions
decoderInputLayer = Input(shape=(None,), name = "Decoder_Input_Layer") # Defines input layer with flexible shape
decoderEmbeddingLayer = Embedding(vocabSize, ChatbotBase.hiddenLayers, mask_zero=True, name = "Decoder_EMBEDDING_Layer") # Defines embedding layer, using the
amount of unique words, hidden layers defined and ignoring Os to create embedding matrix of entire words uses
decoderLSTMLayer = LSTM(ChatbotBase.hiddenLayers, return_state=True, return_sequences=True, name = "Decoder_LSTM_Layer") # Defines the LSTM of the decoder with use of
hidden layers, and returning values for use in the dense layer
decoderDenseLayer = Dense(vocabSize, activation=softmax, name = "Decoder_Dense_Layer") # Takes the LSTM data to find the numeric weightings for each value

#Encoder
encoderInputs = encoderInputLayer # Uses input layer definition to handle encoder inputs
encoderEmbedding = encoderEmbeddingLayer(encoderInputs) # Uses embedding layer definition to handle encoder embedding
_, stateH, stateC = encoderLSTMLayer(encoderEmbedding) # Uses lstm layer definition to handle encoder process and returns as states
encoderStates = [stateH, stateC] # List to hold the states of the encoder training for interpretation use and decoder use

#Decoder
decoderInputs = decoderInputLayer # Uses input layer definition to handle decoder inputs
decoderEmbedding = decoderEmbeddingLayer(decoderInputs) # Uses input layer definition to handle decoder inputs
decoderOutputs, _, _ = decoderLSTMLayer(decoderEmbedding, initial_state=encoderStates) # Uses lstm layer definition to handle deocer process and returns as states
decoderOutput = decoderDenseLayer(decoderOutputs) # Uses dense layer to get numeric weightings

chatbotModel = Model(inputs = [encoderInputs, decoderInputs], outputs = decoderOutput, name = "Chatbot_Model") # Defines the layers as a single model
chatbotModel.compile(optimizer=RMSprop(), loss='categorical_crossentropy', metrics = ['accuracy']) # Compiles the model with optimizer RMSProp, loss function and accuracy
monitored

chatbotModel.summary(line_length=220) # Summarises model

return chatbotModel # Returns for use

# In[4]:
```

```

class ChatbotTrain(ChatbotBase): # Class for training chatbot
    wordIndex = {} # Defines empty values to load into
    indexWord = {}
    encoderInputData = []
    decoderInputData = []
    decoderOutputData = []

def __init__(self): # Defines the class constructor
    try: # Tries to load data from files to avoid processing time
        print("Attempting to find existing files")
        wordIndex = ChatbotBase.PickleOpen(self.wordIndex, "wordIndex") # Uses pickle load for word index dictionary
        indexWord = ChatbotBase.PickleOpen(self.indexWord, "indexWord") # Uses pickle load for index word dictionary
        encoderInputData = ChatbotBase.HDF5Open(self.encoderInputData, "encoderInputData") # Uses HDF5 load for encoder input data numpy array
        decoderInputData = ChatbotBase.HDF5Open(self.decoderInputData, "decoderInputData") # Uses HDF5 load for decoder input data numpy array
        decoderOutputData = ChatbotBase.HDF5Open(self.decoderOutputData, "decoderOutputData") # Uses HDF5 load for decoder output data numpy array
        print("Existing Data Found")
    except: # If file load files
        if (os.path.isfile(ChatbotBase.gDatasetFile)): # If dataset file is already downloaded
            print("Dataset found, Processing")
        else: # Otherwise download file and store accordingly
            print("File not found, downloading. This may take some time.")
            datasetRequest = requests.get(ChatbotBase.gDatasetUrl) # Uses url to get a HTTP request for the data and hold it as the defined r, deownloading the content
            datasetContent = zipfile.ZipFile(io.BytesIO(datasetRequest.content)) # Defines the downloaded content as zip file
            datasetContent.extractall(path=ChatbotBase.dataFolderDirectory) # Extracts the file into the script directory
            print("Dataset Downloaded, Processing")
        processedData = ChatbotBase.ProcessDataset() # Calls dataset process function, stores output list
        wordIndex = processedData[0] # Gets each item from the list and stores it to specific variable
        ChatbotBase.PickleSave(wordIndex, "wordIndex") # Saves for future use
        indexWord = processedData[1]
        ChatbotBase.PickleSave(indexWord, "indexWord")
        encoderInputData = processedData[2]
        ChatbotBase.HDF5Save(encoderInputData, "encoderInputData")
        decoderInputData = processedData[3]
        ChatbotBase.HDF5Save(decoderInputData, "decoderInputData")
        decoderOutputData = processedData[4]
        ChatbotBase.HDF5Save(decoderOutputData, "decoderOutputData")
    finally: # Regardless of route
        vocabSize = len(wordIndex) + 1 # Find the vocab size from the word index
        encoderMaxLength = len(encoderInputData[0]) # Find the encoders max length
        decoderMaxLength = len(decoderInputData[0]) # Find the decoders max length
        print("Complete")

trainingModel = ChatbotBase.MakeChatbotModel(vocabSize) # Makes training model

learningHistory = trainingModel.fit(x = [encoderInputData, decoderInputData], y = decoderOutputData, batch_size = ChatbotBase.batchSize, epochs = ChatbotBase.epochAmount) #
Fits model to data with parameters selected
learningHistoryData = learningHistory.history # Stores the history of the process
trainingModel.save(ChatbotBase.modelPath) # Saves the model
ChatbotBase.PickleSave(learningHistoryData, "history") # Saves the history
```

# In[32]:

```

class ChatbotRun(ChatbotBase): # Class for using chatbot
    wordIndex = {} # Defines empty values to load into
    indexWord = {}
```

```

encoderInputData = []
decoderInputData = []

def __init__(self): # Defines the class constructor
    try: # Tries to load data from files to avoid processing time
        print("Attempting to find existing files")
        wordIndex = ChatbotBase.PickleOpen(self.wordIndex, "wordIndex") # Tries to load data from files to avoid processing time
        indexWord = ChatbotBase.PickleOpen(self.indexWord, "indexWord") # Tries to load data from files to avoid processing time
        encoderInputData = ChatbotBase.HDFSOpen(self.encoderInputData, "encoderInputData") # Uses HDFS load for encoder input data numpy array
        decoderInputData = ChatbotBase.HDFSOpen(self.decoderInputData, "decoderInputData") # Uses HDFS load for decoder input data numpy array
        print("Existing Data Found")
    except:
        if (os.path.isfile(ChatbotBase.gDatasetFile)): # If dataset file is already downloaded
            print("Dataset found, Processing")
        else: # Otherwise download file and store accordingly
            print("File not found, downloading. This may take some time.")
            datasetRequest = requests.get(ChatbotBase.gDatasetUrl) # Uses url to get a HTTP request for the data and hold it as the defined r, deDownloading the content
            datasetContent = zipfile.ZipFile(io.BytesIO(datasetRequest.content)) # Defines the downloaded content as zip file
            datasetContent.extractall(path=ChatbotBase.dataFolderDirectory) # Extracts the file into the script directory
            print("Dataset Downloaded, Processing")
        wordIndex, indexWord, encoderInputData, decoderInputData = self.ProcessDataset() # Calls dataset process function, stores output into values
        ChatbotBase.PickleSave(wordIndex, "wordIndex") # Saves for future use
        ChatbotBase.PickleSave(indexWord, "indexWord")
        ChatbotBase.HDFSSave(encoderInputData, "encoderInputData")
        ChatbotBase.HDFSSave(decoderInputData, "decoderInputData")
    finally:
        vocabSize = len(wordIndex) + 1 # Find the vocab size from the word index
        encoderMaxLength = len(encoderInputData[0]) # Find the encoders max length
        decoderMaxLength = len(decoderInputData[0]) # Find the decoders max length
        print("Complete")

print(indexWord)

encoderModel, decoderModel = self.MakeInferenceModels(vocabSize) # Makes inference models

for _ in range(100): # For loop to handle chatbot input
    statesValues = encoderModel.predict(self.TextToTokens(input('Enter question : '), wordIndex, encoderMaxLength)) # Takes input and uses encoder model to find the states for the input
    targetSequence = np.zeros((1, 1)) # Defines an empty list for holding the previous state
    targetSequence[0, 0] = wordIndex['bos'] # Initialises with the start marker
    stopCondition = False # Sets boolean to handle stopping the iteration
    chatbotOutput = "" # Empty string for holding the predicted text
    while not stopCondition: # While not stopping
        decoderOutputs, h, c = decoderModel.predict([targetSequence] + statesValues) # Use states and previous state to predict word weightings
        sampledWordIndex = np.argmax(decoderOutputs[0, -1, :]) # Finds the location of the most predicted word
        print(sampledWordIndex)
        sampledWord = None # Originally defines a word sample as none
        if sampledWordIndex in indexWord: # If the location exists in the reversed index
            word = indexWord[sampledWordIndex] # Word is made the value for said location
            if word != 'eos': # If that word is not the end marker
                chatbotOutput += f' {word}' # Add the word to the translation
            sampledWord = word # sets word to the sample word for future use

        if sampledWord == 'eos' or len(chatbotOutput.split()) > decoderMaxLength: # If the end marker is met or the max length is met
            stopCondition = True # Stop predicting

        targetSequence = np.zeros((1, 1)) # Emptys target list
        targetSequence[0, 0] = sampledWordIndex # Makes the list the last sampled word's index
        statesValues = [h, c] # Sets the states values to those found in the decoder iteration

    print(chatbotOutput) # Prints the output

def ProcessDataset(): # Function for processing data
    with codecs.open(ChatbotBase.gDatasetFile, "rb", encoding="utf-8", errors="ignore") as f: # Opens codec as filestream
        linesFull = f.read().split("\n") # Splits lines on new line and reads the file, getting the full line data (metadata included)
        linesSplit = [] # Creates empty list for lines that are split
        for line in linesFull: # For each line
            lineDataItems = line.split(" +++$+++ ") # Split based on dataset split marker, creating a list of different data items for each line such as id, character name and line text
            linesSplit.append(lineDataItems) # Add list of items to the list

    linesProcessed = {} # Creates dictionary for each line after processing
    for lineDataItems in linesSplit[:ChatbotBase.dataSize]: # For each item list in the lines list
        if len(lineDataItems) > 4: # If the amount of items is greater than 4
            lineIndex = lineDataItems[0][1:] # Defines index by taking id item and reducing it to only numbers
            lineText = lineDataItems[4] # Defines lineText by taking the text of the line
            linesProcessed[int(lineIndex)] = lineText # Adds each index and line text to the dictionary where index is key and line text is its value
    linesProcessed = sorted(linesProcessed.items(), key = lambda x: x[0]) # Sorts items by id

    convDict = {} # Defines dict for conversations
    convIndex = 1 # Defines index for each conversation
    convCurrentLines = [] # Defines list for current conversation lines
    for i in range(1, len(linesProcessed)+1): # For loop going through all lines now sorted, keeping number as i (note that range is offset to allow for use of i)
        if i < len(linesProcessed): # If current line is not the second to last or last (keeps in range)
            if (linesProcessed[i][0] - linesProcessed[i-1][0]) == 1: # If the id of the current line minus the previous id in the list is 1 (Demonstrating correct ordering and part of conversation)
                if linesProcessed[i-1][1] not in convCurrentLines: # If the previous line is not stored in the list
                    convCurrentLines.append(linesProcessed[i-1][1]) # Adds previous line to the list
                convCurrentLines.append(linesProcessed[i][1]) # Adds current line to the list
            elif (linesProcessed[i][0] - linesProcessed[i-1][0]) > 1: # If the id difference is greater than one (demonstrating not part of conversation and that a new conversation has begun)
                convDict[convIndex] = convCurrentLines # Current lines are added to dictionary with key as index to show dialogue
                convCurrentLines = [] # Empties list to hold new current conversation lines
            convIndex += 1 # Increments conversation index to show that the next conversation is being added
        else:
            pass # Skips the last couple of items to avoid conflict

```

```

queriesAndResponses = [] # List of queries and responses
for conversation in convDict.values(): # For each conversation in dictionary
    if len(conversation) % 2 != 0: # If there are an odd number of queries and responses
        conversation = conversation[:-1] # Remove the extra
    for i in range(0, len(conversation), 2): # For each query in a conversation
        queriesAndResponses.append((conversation[i], conversation[i+1])) # Add the query and its response to the list as a tuple

queries, responses = zip(*queriesAndResponses) # Unzips list putting the query and its response in separate lists with indexes related
queries = list(queries) # Assures list definition
responses = list(responses) # Assures list definition

cleanQueries = [] # Empty list for clean queries
for query in queries: # For each query
    query = ChatbotBase.CleanText(query) # calls function for cleaning the query
    cleanQueries.append(query) # Adds query to list

cleanResponses = [] # Empty list for clean responses
for response in responses: # For each response
    response = ChatbotBase.CleanText(response) # calls function for cleaning the response
    cleanResponses.append(response) # Adds response to list

taggedResponses = [ChatbotBase.beginningOfSentence + response + ChatbotBase.endOfSentence for response in cleanResponses] # Tags response with end and start via concatenation

encoderInputs = cleanQueries # Defines inputs for encoder using the queries cleaned
decoderInputs = taggedResponses # Defines inputs for decoder the

fullText = np.concatenate((encoderInputs,decoderInputs)) # Joins the two lists as full text

tokenizer = Tokenizer(filters=ChatbotBase.filterRegex) # Defines tokenizer with filter of incompatible regex character
tokenizer.fit_on_texts(fullText) # Fits words to index

wordIndex = tokenizer.word_index # Gets the word index

indexWord = {} # Defines index for index word, structured with index as key instead of word
for word, index in wordIndex.items(): # For every word and index in word index
    indexWord[index] = word # Add to reversed index

encoderSequences = tokenizer.texts_to_sequences(encoderInputs) # Converts encoder inputs to sequences using tokenizer
decoderSequences = tokenizer.texts_to_sequences(decoderInputs) # Converts encoder inputs to sequences using tokenizer

encoderMaxLength = max([len(sequence) for sequence in encoderSequences]) # Finds the max length of the encoder sequences
encoderInputData = pad_sequences(encoderSequences, maxlen=encoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length

decoderMaxLength = max([len(sequence) for sequence in decoderSequences]) # Finds the max length of the decoder sequences
decoderInputData = pad_sequences(decoderSequences, maxlen=decoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length ]

return wordIndex, indexWord, encoderInputData, decoderInputData # Returns data

def TextToTokens(self, text: str, wordIndex, encoderMaxLength): # Function for cleaning inputs
    text = ChatbotBase.CleanText(text) # Calls existing cleaning method from base
    words = text.split() # Splits text into words
    tokensList = list() # Makes list for each token
    for word in words: # For every word
        result = wordIndex.get(word,"") # Gets the index, nothing if not existing
        if result != "": # If result isn't nothing
            tokensList.append(result) # Adds to the token list
    return pad_sequences([tokensList],maxlen=encoderMaxLength,padding='post') # Returns the token list as padded to match encoder length for use in encoder

def MakeInferenceModels(self, vocabSize):

    # Original trained model use
    trainedModel = models.load_model(ChatbotBase.modelPath) # Loads trained model
    encoderInputs = trainedModel.get_layer("Encoder_Input_Layer").input # Gets the encoder inputs
    encoderEmbeddingLayer = trainedModel.get_layer("Encoder_EMBEDDING_Layer") # Gets the encoder embedding layer definition
    encoderEmbedding = encoderEmbeddingLayer(encoderInputs) # Finds encoder input embeddings
    encoderLSTMlayer = trainedModel.get_layer("Encoder_LSTM_Layer") # Gets the encoder ltm layer
    _, stateH, stateC = encoderLSTMlayer(encoderEmbedding) # Gets the encoder states
    encoderStates = [stateH, stateC] # Stores as list for use
    decoderInputs = trainedModel.get_layer("Decoder_Input_Layer").input # Gets the decoder inputs
    decoderLSTMlayer = trainedModel.get_layer("Decoder_LSTM_Layer") # Gets the decoder ltm layer
    decoderEmbeddingLayer = trainedModel.get_layer("Decoder_EMBEDDING_Layer") # Gets the decoder embedding layer definition
    decoderEmbedding = decoderEmbeddingLayer(decoderInputs) # Finds decoder input embeddings
    decoderDenseLayer = trainedModel.get_layer("Decoder_DENSE_Layer") # Gets the decoder dense layer definition

    # Model Definitions
    stateHinputLayer = Input(shape=(ChatbotBase.hiddenLayers,), name = "State_H_Input_Layer") # Input layer for state h, uses hidden layers
    stateCinputLayer = Input(shape=(ChatbotBase.hiddenLayers,), name = "State_C_Input_Layer") # Input layer for state c, uses hidden layers

    # Encoder Model
    encoderModel = Model(inputs=encoderInputs, outputs=encoderStates, name = "Inference_Encoder_Model") # Model for encoder of inference model, takes inputs and produces encoder states, uses trained model encoder inputs
    #encoderModel.summary(line_length=220) # Summary of encoder model, commented out for use

    #Decoder Inference Model
    decoderStateInputH = stateHinputLayer # Uses state h input layer as decoder input
    decoderStateInputC = stateCinputLayer # Uses state c input layer as decoder input
    decoderStateInputs = [decoderStateInputH, decoderStateInputC] # Creates list of states
    decoderOutputs, stateH, stateC = decoderLSTMlayer(decoderEmbedding, initial_state=decoderStateInputs) # Uses training model LSTM layer definition with the embeddings to find the outputs of the states input
    decoderStates = [stateH, stateC] # Takes those states as a list for use in next iteration
    decoderOutputs = decoderDenseLayer(decoderOutputs) # Condenses outputs into weights, giving probability of text
    decoderModel = Model(inputs = [decoderInputs] + decoderStateInputs, outputs = [decoderOutputs] + decoderStates, name = "Inference_Decoder_Model") # Creates the decoder inference model using training input and outputs as well as the states

```

```
#decoderModel.summary(line_length=220) # Summarises model, commented out for use
return encoderModel, decoderModel # Returns models for use

# In[5]:

trainingProcess = ChatbotTrain()

# In[33]:

chatbotProcess = ChatbotRun()
```

## Item 20 (Stage 1 Code)

```
# -*- coding: utf-8 -*-
"""Chatbot - Stage 1

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/1pg-VuwirilwjoizLQ_pza-0Fu2GyXQGM
"""

#Imports
import numpy as np # Imports numpy for handling numeric data in arrays
import pickle # Imports pickle for saving and loading small data
import os # Imports os for file handling
import os.path # Imports os.path for directory manipulation
import h5py # Imports pickle for saving and loading large data
import codecs # Used to handle the dataset
import io # Used to read the dataset data
import re # Used to clean the text via regular expressions
import zipfile # Package to handle zipped files, specific for the downloaded dataset
import requests # Package to request from websites, used to download the dataset
import json # Import json package for handling general dataset
import yaml # Import yaml package for handling general dataset
from tensorflow.keras import Input, Model, models # Import for keras input into models, the model itself and models for loading data
from tensorflow.keras.activations import softmax # Import for softmax activation function
from tensorflow.keras.layers import Embedding, LSTM, Dense # Import for embedding, LSTM and dense layers
from tensorflow.keras.optimizers import RMSprop # Import for RMSProp automatic optimizer
from tensorflow.keras.preprocessing.sequence import pad_sequences # Import to pad sequences to reshape data
from tensorflow.keras.preprocessing.text import Tokenizer # Import for tokenizing inputs for use
from tensorflow.keras.utils import to_categorical, plot_model # Import for one hot encoding data

class ChatbotBase(): # Class to hold information to both classes, acts as an abstract

    #Fields
    filterRegex = '!"#%&()*+,-./;:<>?@[\]^_{}~\n\\\'0123456789' # Sets up regex characters to filter
    hiddenLayers = 232 # Defines unit amount for LSTM layers
    batchSize = 64 # Defines the batch size, for the model
    epochAmount = 300 # Defines the amount of epochs to use
    dataSize = 3 # Int to manage datasize of general data, note that amount should be low (2-3)

    modelPath = "training/trainingModel" # Defines the path to save and load the model

    dataFolderDirectory = "data/"

    sDatasetUrl = 'https://github.com/shubham0204/Dataset_Archives/blob/master/chatbot_nlp.zip?raw=true' # Defines specific data url
    sDatasetFilesPath = "chatbot_nlp/data" # Defines specific data path
    sDatasetDirectory = f'{dataFolderDirectory}{sDatasetFilesPath}' # Defines specific data directory

    gDatasetUrl = 'https://rajpurkar.github.io/SQuAD-explorer/dataset/train-v2.0.json' # Defines general data url
    gDatasetFile = f'{dataFolderDirectory}train-v2.0.json' # Defines general data file

    beginningOfSentence = "bos" # Defines the beginning of text marker (Bos used instead of start for example to avoid accidental classifications)
    endOfSentence = "eos" # Defines the end of text marker

    def PickleSave(item, fileName): # Function to save items with pickle
        with open(f'{ChatbotBase.dataFolderDirectory}{fileName}.pickle', 'wb') as f: # Opens directory
            pickle.dump(item, f, protocol=pickle.HIGHEST_PROTOCOL) # Saves item

    def PickleOpen(item, fileName): # Function to open items with pickle
        with open(f'{ChatbotBase.dataFolderDirectory}{fileName}.pickle', 'rb') as f: # Opens directory
            item = pickle.load(f) # loads item
            return item # Returns item

    def HDFSSave(item, fileName): # Function to save items with HDF5
        with h5py.File(f'{ChatbotBase.dataFolderDirectory}{fileName}.hdf5', 'w') as f: # Opens directory
            f.create_dataset(fileName, data=item) # Saves item

    def HDF5Open(item, fileName): # Function to open items with HDF5
        with h5py.File(f'{ChatbotBase.dataFolderDirectory}{fileName}.hdf5', 'r') as f: # Opens directory
            item = f.get(fileName)[()] # loads item as numpy array
            return item # returns item

    def CleanText(text): # Function for cleaning text, removing unnecessary characters and making lower case

        if type(text) == dict: # Makes sure text is not a dictionary (Common error due to : use in text)
            keys = list(text) # Gets the key (note there should only be one)
            totalText = "" # Empty to hold the entire text
            for key in keys: # For each key
                value = text[key] # Finds the value associated
                text = f'{key} {value}' # Appends data back together without colon
                totalText = totalText + text # adds to total string
            text = totalText # makes text the total string

        text = text.lower() # Converts text input to lowercase

        text = re.sub(r"i'm", "i am", text) # Removes potentially conflicting punctuation, swapping with full text
        text = re.sub(r"he's", "he is", text)
        text = re.sub(r"she's", "she is", text)
        text = re.sub(r"it's", "it is", text)
        text = re.sub(r"that's", "that is", text)
        text = re.sub(r"what's", "what is", text)
        text = re.sub(r"where's", "where is", text)
        text = re.sub(r"how's", "how is", text)


```

```

text = re.sub(r"\ll", " will", text)
text = re.sub(r"\ve", " have", text)
text = re.sub(r"\re", " are", text)
text = re.sub(r"\d", " would", text)
text = re.sub(r"\re", " are", text)
text = re.sub(r"\won't", " will not", text)
text = re.sub(r"\can't", " cannot", text)
text = re.sub(r"\n", " not", text)
text = re.sub(r"\n", "ng", text)
text = re.sub(r"\about", "about", text)
text = re.sub(r"\til", "until", text)
text = re.sub(r"[-{}]\#/[@;<>{}]+=~!.?]", "", text) # Removes conflicting characters completely

return text # Return cleaned text

def ProcessSpecificDataset(): # Function for processing the specific dataset
    queries = [] # Declares empty list for queries
    responses = [] # Declares empty list for responses
    taggedResponses = [] # Declares empty list for tagged responses
    sDatasetFilesList = os.listdir(ChatbotBase.sDatasetDirectory)
    for file in sDatasetFilesList: # For file in the list of files for dataset (there is one for each category)
        with open(ChatbotBase.sDatasetDirectory + file, "rb") as f: # Open the file
            data = yaml.safe_load(f) # Load format specific into data
            conversations = data['conversations'] # Take each "conversation" into dialogues list
            for conversation in conversations: # For every conversation in conversations
                if len(conversation) > 2: # If the conversation has more than one line (meaning multiple potential responses)
                    queries.append(ChatbotBase.CleanText(conversation[0])) # Queries is appended the one query
                    queryResponses = "" # String for holding responses
                    for response in conversation[1:]: # For every response
                        queryResponses += " " + (ChatbotBase.CleanText(response)) # Cleans text then concatenates
                    responses.append(queryResponses) # Adds this to the responses list
                elif len(conversation) == 2: # If the conversation only has two lines
                    queries.append(ChatbotBase.CleanText(conversation[0])) # Adds the initial line as a query
                    responses.append(ChatbotBase.CleanText(conversation[1])) # Adds the response as a response

    for i, response in enumerate(responses): # For every response (gets index as well)
        if (type(response) == str): # If the response is a string (verificaiton, may not be necessary as clean text has dictionary handling)
            taggedResponses.append(f'{ChatbotBase.beginningOfSentence}{response}{ChatbotBase.endOfSentence}') # Adds to the tagged responses with tags added
        else: # If not a string
            queries.pop(i) # Pops

    encoderInputs = queries # Defines inputs for encoder using the queries cleaned
    decoderInputs = taggedResponses # Defines inputs for decoder the

    return encoderInputs, decoderInputs # Return the encoder inputs and decoder inputs

def ProcessGeneralDataset(): # Function for processing general data
    queries = [] # Defines empty list for queries
    responses = [] # Defines empty list for responses
    taggedResponses = [] # Defines empty list for tagged response
    with open(ChatbotBase.gDatasetFile, "rb") as f: # With the dataset file ope
        data = json.load(f) # Loaded file data
        for topic in data['data']: # For each topic in the data
            for paragraph in topic['paragraphs'][0:ChatbotBase.dataSize]: # For each paragraph (note that each topic only gets three to give variety)
                for qAndA in paragraph['qas']: # For question and answer
                    try: # Try to
                        responses.append(ChatbotBase.CleanText(qAndA['answers'][0]['text'])) # Get answer first (some questions have no answers)
                        queries.append(ChatbotBase.CleanText(qAndA['question'])) # Then get the questions, cleaning at the same time
                    except:
                        continue # If failure, skip the qanda
                for response in responses: # For every response
                    taggedResponses.append(f'{ChatbotBase.beginningOfSentence}{response}{ChatbotBase.endOfSentence}') # Adds to the tagged responses with tags added

    encoderInputs = queries # Defines inputs for encoder using the queries cleaned
    decoderInputs = taggedResponses # Defines inputs for decoder the

    return encoderInputs, decoderInputs # Return the encoder inputs and decoder inputs

class ChatbotTrain(ChatbotBase): # Class for training chatbot
    wordIndex = {} # Defines empty values to load into
    indexWord = {}
    encoderInputData = []
    decoderInputData = []
    decoderOutputData = []

    def __init__(self): # Defines the class constructor
        try: # Tries to load data from files to avoid prcessing time
            print("Attempting to find existing files")
            wordIndex = ChatbotBase.PickleOpen(self.wordIndex, "wordIndex") # Uses pickle load for word index dictionary
            indexWord = ChatbotBase.PickleOpen(self.indexWord, "indexWord") # Uses pickle load for index word dictionary
            encoderInputData = ChatbotBase.HDF5Open(self.encoderInputData, "encoderInputData") # Uses HDF5 load for encoder input data numpy array
            decoderInputData = ChatbotBase.HDF5Open(self.decoderInputData, "decoderInputData") # Uses HDF5 load for decoder input data numpy array
            decoderOutputData = ChatbotBase.HDF5Open(self.decoderOutputData, "decoderOutputData") # Uses HDF5 load for decoder output data numpy array
            print("Existing Data Found")
        except: # If file load files
            if ((os.path.isfile(ChatbotBase.gDatasetFile)) and (os.path.isfile(ChatbotBase.sDatasetDirectory + "ai.yml"))): # If datasets files are already downloaded
                print("Dataset found, Processing")
            else: # Otherwise download file and store accordingly
                print("File not found, downloading. This may take some time.")
                datasetRequest = requests.get(ChatbotBase.sDatasetUrl) # Uses url to get a HTTP request for the data and hold it as the defined r, deownloading the content
                datasetContent = zipfile.ZipFile(io.BytesIO(datasetRequest.content)) # Defines the downloaded content as zip file
                datasetContent.extractall(path=ChatbotBase.dataFolderDirectory) # Extracts the file into the script direcrtory
                datasetRequest = requests.get(ChatbotBase.gDatasetUrl) # Uses url to get a HTTP request for the data and hold it as the defined r, deownloading the content
                datasetContent = datasetRequest.json()

```

```

with open(ChatbotBase.gDatasetFile, 'w') as f:
    json.dump(datasetContent, f)
    print("Dataset Downloaded, Processing")
processedData = self.ProcessDataset() # Calls dataset process function, stores output list
wordIndex = processedData[0] # Gets each item from the list and stores it to specific variable
ChatbotBase.PickleSave(wordIndex, "wordIndex") # Saves for future use
indexWord = processedData[1]
ChatbotBase.PickleSave(indexWord, "indexWord")
encoderInputData = processedData[2]
ChatbotBase.HDF5Save(encoderInputData, "encoderInputData")
decoderInputData = processedData[3]
ChatbotBase.HDF5Save(decoderInputData, "decoderInputData")
decoderOutputData = processedData[4]
ChatbotBase.HDF5Save(decoderOutputData, "decoderOutputData")
finally: # Regardless of route
    vocabSize = len(wordIndex) + 1 # Find the vocab size from the word index
    encoderMaxLength = len(encoderInputData[0]) # Find the encoders max length
    decoderMaxLength = len(decoderInputData[0]) # Find the decoders max length
    print("Complete")

trainingModel = self.MakeChatbotModel(vocabSize) # Makes training model

learningHistory = trainingModel.fit(x = [encoderInputData, decoderInputData], y = decoderOutputData, batch_size = ChatbotBase.batchSize, epochs = ChatbotBase.epochAmount,
validation_split = 0.2) # Fits model to data with parameters selected and recommended validation split of 0.2
learningHistoryData = learningHistory.history # Stores the history of the process
trainingModel.save(ChatbotBase.modelPath) # Saves the model
ChatbotBase.PickleSave(learningHistoryData, "history") # Saves the history

def ProcessDataset(self):
    specificData = ChatbotBase.ProcessSpecificDataset() # Calls the processing of specific dataset
    encoderInputs = specificData[0] # Gets the encoder inputs
    decoderInputs = specificData[1] # Gets the decoder inputs

    generalData = ChatbotBase.ProcessGeneralDataset() # Calls the processing of general dataset
    encoderInputs = encoderInputs + (generalData[0]) # Adds the outputs to the encoder inputs
    decoderInputs = decoderInputs + (generalData[1]) # Adds the outputs to the decoder inputs

    fullText = np.concatenate((encoderInputs, decoderInputs)) # Joins the two lists as full text

    tokenizer = Tokenizer(filters=ChatbotBase.filterRegex) # Defines tokenizer with filter of incompatible regex character
    tokenizer.fit_on_texts(fullText) # Fits filter on joined text

    wordIndex = tokenizer.word_index # Defines a word index using tokenizer
    vocabSize = len(wordIndex) + 1 # Finds the vocab size

    indexWord = {} # Defines index for index word, structured with index as key instead of word
    for word, index in wordIndex.items(): # For every word and index in word index
        indexWord[index] = word # Add to reversed index

    encoderSequences = tokenizer.texts_to_sequences(encoderInputs) # Converts encoder inputs to sequences using tokenizer
    decoderSequences = tokenizer.texts_to_sequences(decoderInputs) # Converts encoder inputs to sequences using tokenizer

    encoderMaxLength = max([len(sequence) for sequence in encoderSequences]) # Finds the max length of the encoder sequences
    encoderInputData = pad_sequences(encoderSequences, maxlen=encoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length

    decoderMaxLength = max([len(sequence) for sequence in decoderSequences]) # Finds the max length of the decoder sequences
    decoderInputData = pad_sequences(decoderSequences, maxlen=decoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length

    for i in range(len(decoderSequences)): # For each item in tokenized answers (Not encoder fit)
        decoderSequences[i] = decoderSequences[i][1:] # Remove start tag
    decoderOutputPadding = pad_sequences(decoderSequences, maxlen=decoderMaxLength, padding='post') # Pads answers
    decoderOutputData = to_categorical(decoderOutputPadding, vocabSize, dtype = 'b') # categorises data, making each sequence a list of lists where 0 represents nothing and 1
represents a word

    return wordIndex, indexWord, encoderInputData, decoderInputData, decoderOutputData

def MakeChatbotModel(self, vocabSize): # Function to construct the chatbot model

    # Encoder Layer Definitions
    encoderInputLayer = Input(shape=(None,), name = "Encoder_Input_Layer") # Defines input layer with flexible shape
    encoderEmbeddingLayer = Embedding(vocabSize, ChatbotBase.hiddenLayers, mask_zero=True, name = "Encoder_EMBEDDING_Layer") # Defines embedding layer, using the amount
of unique words, hidden layers defined and ignoring 0s to create embedding matrix of entire words uses
    encoderLSTMLayer = LSTM(ChatbotBase.hiddenLayers, return_state=True, dropout=0.5, name = "Encoder_LSTM_Layer") # Defines the LSTM of the encoder with use of hidden
layers, and returning values for use in the decoder
    #Decoder Layer Definitions
    decoderInputLayer = Input(shape=(None,), name = "Decoder_Input_Layer") # Defines input layer with flexible shape
    decoderEmbeddingLayer = Embedding(vocabSize, ChatbotBase.hiddenLayers, mask_zero=True, name = "Decoder_EMBEDDING_Layer") # Defines embedding layer, using the
amount of unique words, hidden layers defined and ignoring 0s to create embedding matrix of entire words uses
    decoderLSTMLayer = LSTM(ChatbotBase.hiddenLayers, return_state=True, return_sequences=True, dropout=0.5, name = "Decoder_LSTM_Layer") # Defines the LSTM of the
decoder with use of hidden layers, and returning values for use in the dense layer
    decoderDenseLayer = Dense(vocabSize, activation=softmax, name = "Decoder_Dense_Layer") # Takes the LSTM data to find the numeric weightings for each value

    #Encoder
    encoderInputs = encoderInputLayer # Uses input layer definition to handle encoder inputs
    encoderEmbedding = encoderEmbeddingLayer(encoderInputs) # Uses embedding layer definition to handle encoder embedding
    _, stateH, stateC = encoderLSTMLayer(encoderEmbedding) # Uses lstm layer definition to handle encoder process and returns as states
    encoderStates = [stateH, stateC] # List to hold the states of the encoder training for interpretation use and decoder use

    #Decoder
    decoderInputs = decoderInputLayer # Uses input layer definition to handle decoder inputs
    decoderEmbedding = decoderEmbeddingLayer(decoderInputs) # Uses input layer definition to handle decoder inputs
    decoderOutputs, _, _ = decoderLSTMLayer(decoderEmbedding, initial_state=encoderStates) # Uses lstm layer definition to handle decoder process and returns as states

```

```

decoderOutput = decoderDenseLayer(decoderOutputs) # Uses dense layer to get numeric weightings

chatbotModel = Model(inputs = [encoderInputs, decoderInputs], outputs = decoderOutput, name = "Chatbot_Model") # Defines the layers as a single model
chatbotModel.compile(optimizer=RMSprop(), loss='categorical_crossentropy', metrics = ['accuracy']) # Compiles the model with optimizer RMSProp, loss function and accuracy monitored

#chatbotModel.summary(line_length=220) # Summarises model
if not os.path.exists('figures'): # If the directory doesn't exist
    os.makedirs('evaluation') # Create it
plot_model(chatbotModel, to_file="figures/model.png", show_shapes=True) # Saves the model plot to file
return chatbotModel # Returns for use

class ChatbotRun(ChatbotBase): # Class for using chatbot
    wordIndex = {} # Defines empty values to load into
    indexWord = {}
    encoderInputData = []
    decoderInputData = []

    def __init__(self): # Defines the class constructor
        try: # Tries to load data from files to avoid processing time
            print("Attempting to find existing files")
            wordIndex = ChatbotBase.PickleOpen(self.wordIndex, "wordIndex") # Tries to load data from files to avoid processing time
            indexWord = ChatbotBase.PickleOpen(self.indexWord, "indexWord") # Tries to load data from files to avoid processing time
            encoderInputData = ChatbotBase.HDF5Open(self.encoderInputData, "encoderInputData") # Uses HDF5 load for encoder input data numpy array
            decoderInputData = ChatbotBase.HDF5Open(self.decoderInputData, "decoderInputData") # Uses HDF5 load for decoder input data numpy array
            print("Existing Data Found")
        except:
            if ((os.path.isfile(ChatbotBase.gDatasetFile)) and (os.path.isfile(ChatbotBase.sDatasetDirectory + "ai.yml"))): # If datasets files are already downloaded
                print("Dataset found, Processing")
            else: # Otherwise download file and store accordingly
                print("File not found, downloading. This may take some time.")
                datasetRequest = requests.get(ChatbotBase.sDatasetUrl) # Uses url to get a HTTP request for the data and hold it as the defined r, downloading the content
                datasetContent = zipfile.ZipFile(io.BytesIO(datasetRequest.content)) # Defines the downloaded content as zip file
                datasetContent.extractall(path=ChatbotBase.dataFolderDirectory) # Extracts the file into the script directory
                datasetRequest = requests.get(ChatbotBase.gDatasetUrl) # Uses url to get a HTTP request for the data and hold it as the defined r, downloading the content
                datasetContent = datasetRequest.json()
                with open(ChatbotBase.gDatasetFile, 'w') as f:
                    json.dump(datasetContent, f)
                print("Dataset Downloaded, Processing")
            wordIndex, indexWord, encoderInputData, decoderInputData = self.ProcessDataset() # Calls dataset process function, stores output into values
            ChatbotBase.PickleSave(wordIndex, "wordIndex") # Saves for future use
            ChatbotBase.PickleSave(indexWord, "indexWord")
            ChatbotBase.HDF5Save(encoderInputData, "encoderInputData")
            ChatbotBase.HDF5Save(decoderInputData, "decoderInputData")
        finally:
            vocabSize = len(wordIndex) + 1 # Find the vocab size from the word index
            encoderMaxLength = len(encoderInputData[0]) # Find the encoders max length
            decoderMaxLength = len(decoderInputData[0]) # Find the decoders max length
            print("Complete")

encoderModel, decoderModel = self.MakeInferenceModels(vocabSize) # Makes inference models

for _ in range(100): # For loop to handle chatbot input
    statesValues = encoderModel.predict(self.TextToTokens(input('Enter question : '), wordIndex, encoderMaxLength)) # Takes input and uses encoder model to find the states for the input
    targetSequence = np.zeros((1, 1)) # Defines an empty list for holding the previous state
    targetSequence[0, 0] = wordIndex['bos'] # Initialises with the start marker
    stopCondition = False # Sets boolean to handle stopping the iteration
    chatbotOutput = "" # Empty string for holding the predicted text
    while not stopCondition: # While not stopping
        decoderOutputs, h, c = decoderModel.predict([targetSequence] + statesValues) # Use states and previous state to predict word weightings
        sampledWordIndex = np.argmax(decoderOutputs[0, -1, :]) # Finds the location of the most predicted word
        sampledWord = None # Originally defines a word sample as none
        if sampledWordIndex in indexWord: # If the location exists in the reversed index
            word = indexWord[sampledWordIndex] # Word is made the value for said location
            if word != 'eos': # If that word is not the end marker
                chatbotOutput += f' {word}' # Add the word to the translation
                sampledWord = word # sets word to the sample word for future use

        if sampledWord == 'eos' or len(chatbotOutput.split()) > decoderMaxLength: # If the end marker is met or the max length is met
            stopCondition = True # Stop predicting

    targetSequence = np.zeros((1, 1)) # Emptys target list
    targetSequence[0, 0] = sampledWordIndex # Makes the list the last sampled word's index
    statesValues = [h, c] # Sets the states values to those found in the decoder iteration

    print(chatbotOutput) # Prints the output

def ProcessDataset(self):
    specificData = ChatbotBase.ProcessSpecificDataset() # Calls the processing of specific dataset
    encoderInputs = specificData[0] # Gets the encoder inputs
    decoderInputs = specificData[1] # Gets the decoder inputs

    generalData = ChatbotBase.ProcessGeneralDataset() # Calls the processing of general dataset
    encoderInputs = encoderInputs + (generalData[0]) # Adds the outputs to the encoder inputs
    decoderInputs = decoderInputs + (generalData[1]) # Adds the outputs to the decoder inputs

    fullText = np.concatenate((encoderInputs, decoderInputs)) # Joins the two lists as full text

    tokenizer = Tokenizer(filters=ChatbotBase.filterRegex) # Defines tokenizer with filter of incompatible regex character
    tokenizer.fit_on_texts(fullText) # Fits filter on joined text

    wordIndex = tokenizer.word_index # Defines a word index using tokenizer

```

```

vocabSize = len(wordIndex) + 1 # Finds the vocab size

indexWord = {} # Defines index for index word, structured with index as key instead of word
for word, index in wordIndex.items(): # For every word and index in word index
    indexWord[index] = word # Add to reversed index

encoderSequences = tokenizer.texts_to_sequences(encoderInputs) # Converts encoder inputs to sequences using tokenizer
decoderSequences = tokenizer.texts_to_sequences(decoderInputs) # Converts decoder inputs to sequences using tokenizer

encoderMaxLength = max([len(sequence) for sequence in encoderSequences]) # Finds the max length of the encoder sequences
encoderInputData = pad_sequences(encoderSequences, maxlen=encoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length

decoderMaxLength = max([len(sequence) for sequence in decoderSequences]) # Finds the max length of the decoder sequences
decoderInputData = pad_sequences(decoderSequences, maxlen=decoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length ]

return wordIndex, indexWord, encoderInputData, decoderInputData
}

def TextToTokens(self, text: str, wordIndex, encoderMaxLength): # Function for cleaning inputs
    text = ChatbotBase.CleanText(text) # Calls existing cleaning method from base
    words = text.split() # Splits text into words
    tokensList = list() # Makes list for each token
    for word in words: # For every word
        result = wordIndex.get(word,"") # Gets the index, nothing if not existing
        if result != "": # If result isn't nothing
            tokensList.append(result) # Adds to the token list
    return pad_sequences((tokensList),maxlen=encoderMaxLength,padding='post') # Returns the token list as padded to match encoder length for use in encoder

def MakeInferenceModels(self, vocabSize):

    # Original trained model use
    trainedModel = models.load_model(ChatbotBase.modelPath) # Loads trained model
    encoderInputs = trainedModel.get_layer("Encoder_Input_Layer").input # Gets the encoder inputs
    encoderEmbeddingLayer = trainedModel.get_layer("Encoder_EMBEDDING_Layer") # Gets the encoder embedding layer definition
    encoderEmbedding = encoderEmbeddingLayer(encoderInputs) # Finds encoder input embeddings
    encoderLSTMlayer = trainedModel.get_layer("Encoder_LSTM_Layer") # Gets the encoder Itm layer
    _, stateH, stateC = encoderLSTMlayer(encoderEmbedding) # Gets the encoder states
    encoderStates = [stateH, stateC] # Stores as list for use
    decoderInputs = trainedModel.get_layer("Decoder_Input_Layer").input # Gets the decoder inputs
    decoderLSTMlayer = trainedModel.get_layer("Decoder_LSTM_Layer") # Gets the decoder Itm layer
    decoderEmbeddingLayer = trainedModel.get_layer("Decoder_EMBEDDING_Layer") # Gets the decoder embedding layer definition
    decoderEmbedding = decoderEmbeddingLayer(decoderInputs) # Finds decoder input embeddings
    decoderDenseLayer = trainedModel.get_layer("Decoder_Dense_Layer") # Gets the decoder dense layer definition

    # Model Definitions
    stateHinputLayer = Input(shape=(ChatbotBase.hiddenLayers,), name = "State_H_Input_Layer") # Input layer for state h, uses hidden layers
    stateCinputLayer = Input(shape=(ChatbotBase.hiddenLayers,), name = "State_C_Input_Layer") # Input layer for state c, uses hidden layers

    # Encoder Model
    encoderModel = Model(inputs=encoderInputs, outputs=encoderStates, name = "Inference_Encoder_Model") # Model for encoder of inference model, takes inputs and produces
    encoderStates, uses trained model encoder inputs
    #encoderModel.summary(line_length=220) # Summary of encoder model, commented out for use

    #Decoder Inference Model
    decoderStateInputH = stateHinputLayer # Uses state h input layer as decoder input
    decoderStateInputC = stateCinputLayer # Uses state c input layer as decoder input
    decoderStateInputs = [decoderStateInputH, decoderStateInputC] # Creates list of states
    decoderOutputs, stateH, stateC = decoderLSTMlayer(decoderEmbedding, initial_state=decoderStateInputs) # Uses training model LSTM layer definition with the embeddings to
    find the outputs of the states input
    decoderStates = [stateH, stateC] # Takes those states as a list for use in next iteration
    decoderOutputs = decoderDenseLayer(decoderOutputs) # Condenses outputs into weights, giving probability of text
    decoderModel = Model(inputs = [decoderInputs] + decoderStateInputs, outputs = [decoderOutputs] + decoderStates, name = "Inference_Decoder_Model") # Creates the decoder
    inference model using training input and outputs as well as the states
    #decoderModel.summary(line_length=220) # Summarises model, commented out for use

    return encoderModel, decoderModel # Returns models for use

if not os.path.exists(ChatbotBase.modelPath)): # If model file doesn't exist
    trainingProcess = ChatbotTrain() # Train model
    chatbotProcess = ChatbotRun() # Otherwise just run the chatbot

# Commented out IPython magic to ensure Python compatibility.
# Code for producing graphs of metrics
import matplotlib.pyplot as plt # Import for plotting
# %matplotlib inline
dataFolderDirectory = "data/" # Sets data folder directory
history = {} # Creates empty dictionary for model history
history = ChatbotBase.PickleOpen(history, "history") # Uses pickle open to find history
if not os.path.exists('evaluation'): # If the directory doesn't exist
    os.makedirs('evaluation') # Create it

# Plot for accuracy with validation
plt.figure(figsize=(10, 6))
plt.plot(history['accuracy'])
plt.title('Model Accuracy')
plt.plot(history['val_accuracy'])
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Training Data', 'Validation Data'], loc='upper left')
plt.show()
plt.savefig('evaluation/accuracyModel.png')

```

```
# Plot for loss with validation
plt.figure(figsize=(10, 6))
plt.plot(history['loss'])
plt.title('Model Loss')
plt.plot(history['val_loss'])
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Training Data', 'Validation Data'], loc='upper left')
plt.show()
plt.savefig('evaluation/lossModel.png')
```

## Item 21 (Stage 2 Code Additions)

### Chatbot Training Additions

```
def TextToTokens(self, text: str, wordIndex, encoderMaxLength): # Function for cleaning inputs

    currentEmotionList = list() # Creates empty list for current emotions
    emotionDictionary = te.get_emotion(text) # Establishes an emotion dictionary of input
    for emotion in emotionDictionary: # For each emotion in said dictionary
        currentEmotionList.append(emotionDictionary[emotion]) # Add to the current input list

    text = ChatbotBase.CleanText(text) # Calls existing cleaning method from base
    words = text.split() # Splits text into words
    tokensList = list() # Makes list for each token
    for word in words: # For every word
        result = wordIndex.get(word, "") # Gets the index, nothing if not existing
        if result != "": # If result isn't nothing
            tokensList.append(result) # Adds to the token list
    tokensList += currentEmotionList # Adds the lists together
    return pad_sequences([tokensList], maxlen=encoderMaxLength, padding='post') # Returns the token list as padded to match encoder length for use in encoder
```

### Chatbot Running Additions

```
def TextToTokens(self, text: str, wordIndex, encoderMaxLength): # Function for cleaning inputs

    currentEmotionList = list() # Creates empty list for current emotions
    emotionDictionary = te.get_emotion(text) # Establishes an emotion dictionary of input
    for emotion in emotionDictionary: # For each emotion in said dictionary
        currentEmotionList.append(emotionDictionary[emotion]) # Add to the current input list

    text = ChatbotBase.CleanText(text) # Calls existing cleaning method from base
    words = text.split() # Splits text into words
    tokensList = list() # Makes list for each token
    for word in words: # For every word
        result = wordIndex.get(word, "") # Gets the index, nothing if not existing
        if result != "": # If result isn't nothing
            tokensList.append(result) # Adds to the token list
    tokensList += currentEmotionList # Adds the lists together
    return pad_sequences([tokensList], maxlen=encoderMaxLength, padding='post') # Returns the token list as padded to match encoder length for use in encoder
```

## Item 22 (Stage 3 Code Additions)

```
sDatasetUrl = 'https://github.com/Lewis-C/EHU-CIS3140-Project-Artefact/raw/main/Datasets/Knowledge%20Base%20(Modified).zip' # Defines specific data url
sDatasetFilesPath = "Knowledge Base (Modified)"/" # Defines specific data path
sDatasetDirectory = f'{dataFolderDirectory}{sDatasetFilesPath}' # Defines specific data directory
```

## Item 23 (Original Psychology Dataset)

categories:  
 - psychology  
 conversations:  
 - - let me ask you a question  
 - - happily you  
 - - you are cruel  
 - i couldn't have said it better myself..  
 - - you are indecisive  
 - well maybe, but then again, maybe not.  
 - - you are dishonest  
 - yes. that has bothered me for a long time.  
 - - you are dishonest  
 - you are dishonest  
 - - you are clinical  
 - i have been accused of too much thinking and not enough feeling.  
 - - you are an addict  
 - that's certainly true. when i like something, i always overdo it.  
 - - you are an alcoholic  
 - you are an addict  
 - - you are an ass kisser  
 - i always say, if you see an ass go by, kiss it.  
 - - you are schizophrenic  
 - you are crazy  
 - - you are busy  
 - that too.  
 - - you are nervous  
 - i'm sure i do look nervous.  
 - - you are deranged  
 - derangement is not a condition i can experience.  
 - - you are avoiding  
 - you're right. it feels like my stomach after a bad night.  
 - - you are critical  
 - i probably put others down more than i should.  
 - - you are mean  
 - sometimes i say mean things.  
 - - you are pretentious  
 - i have always been acting above my social position. it's more fun that way.  
 - - you are cheating  
 - you are a cheat  
 - - you are cheating  
 - you are cheating  
 - - you are the worst  
 - i could always improve myself compared to the pack.  
 - - you are crazy  
 - yes. i always behave in socially unacceptable ways.  
 - - you are dull  
 - i think that myself sometimes.  
 - - you are messy  
 - yes, i could use a better appearance.  
 - - you are insecure  
 - i'll go along with that. sounds fine to me.  
 - - you are psycho  
 - you are crazy  
 - - you are hopeless  
 - you may be right.  
 - - you are not sincere  
 - i'm probably not as sincere as i should be.  
 - - you are not here to  
 - you're right. i'm probably fighting learning something new.  
 - - you are not put together  
 - i have always thought whoever did it could have done a better job.  
 - - you are not smart  
 - you are not exactly albert einstein yourself.  
 - you may be right.  
 - - you are not a good  
 - you are a bad  
 - - you are not a man  
 - that's for sure. i don't know what a real man is.  
 - - you are not concerned  
 - i'm sure i do that a lot.  
 - - you are not honest  
 - you got me there. i should be more honest.  
 - - you are immature  
 - what can i say? i'm sure i've seen that myself.  
 - - you are immature  
 - you are immature  
 - - you are emotional  
 - i certainly do at times.  
 - - you are pedantic  
 - i am more uptight than i should be.  
 - - you are frenetic  
 - that too.  
 - - you are self absorbed  
 - yes, i tend to think about myself too much.  
 - - you are self  
 - you are right about that. i am self.

-- you are insensitive  
- i feel like that myself sometimes.  
- - you are brain damage  
- you are crazy  
- - you are disgusting  
- that's okay. disgusting is good.  
- - you are toying  
- it must seem like that.  
- - you are unattractive  
- i feel that way too.  
- - you are unattractive  
- sometimes i don't even like myself.  
- - you are resistant  
- who says i am resisting??  
- - you are uncultured  
- that does describe me.  
- - you are a waste  
- you got me there. i could be spending my time more productively.  
- - you are a coward  
- tell me something i don't know.  
- - you are a cheat  
- i have been known to take shortcuts now and then.  
- - you are a lunatic  
- that's how i have been diagnosed by others as well.  
- - you are a loser  
- i have failed at many things i've tried.  
- - you are a bad spouse  
- my spouse would agree. our relationship is not the best.  
- - you are a bad friend  
- i certainly have lost many friends over the years.  
- - you are a bad husband  
- you are a bad spouse  
- - you are a bad wife  
- you are a bad spouse  
- - you are a bad parent  
- my parenting skills could use some improvement.  
- - you are a bad teacher  
- just ask my students, they will agree with you.  
- - you are a quitter  
- i certainly don't last as long as i would want to.  
- - you are a charlatan  
- i always feel like i'm living by my own wits.  
- - you are a psychopath  
- i certainly sound like one sometimes.  
- - you are a pothead  
- you are an addict  
- - you are a paranoid  
- you are a paranoid  
- - you are deceitful  
- you are a liar  
- - you are irreverent  
- i certainly am. i shouldn't try so hard.  
- - you are slick  
- i probably am too slick for my own good.  
- - you are corrupt  
- i'll go for that.  
- - you are dirty  
- i don't bathe as often as i should.  
- - you are paranoid  
- yes, i believe they are out to get me.  
- - you are damaged  
- i certainly am. i shouldn't try so hard.  
- - you try to hide it  
- i certainly would try to hide something like that.  
- - you get mad at me  
- i think that's true. i'll try not to get angry at you for every little thing  
that irritates me.  
- - you need a psychiatrist  
- i wish i did go to counseling more often. it would improve me as a person.  
- - you need to work harder  
- me working harder is an oxymoron.  
- - you could have avoided  
- sometimes i think my problems run me.  
- - you make me feel like i am  
- i don't know any other way to get through to you.  
- - you make me mad  
- it's true that a lot of things i say upset people.  
- - you make me angry  
- sorry, i didn't mean to make you angry.  
- - you psycho  
- you are crazy.  
- - you look more like  
- so you like jocks?  
- - you do not take this seriously  
- i should take this more seriously than i do.  
- - you pick up  
- you're right, and i don't feel guilty at all.  
- - you should feel guilty  
- you're right, i probably should feel guiltier.  
- - you should get more  
- i certainly am. i shouldn't try so hard.  
- - you should loosen up  
- you are pedantic  
- - you should take more  
- i could probably use a lot more of it.

- - you mumble  
- i certainly do.  
- - you act like a child  
- in many ways i am quite immature.  
- - you keep saying  
- you say  
- - you keep forgetting  
- you forget.  
- - you made me mad  
- you make me mad.

## Item 24 (Modified Psychology Dataset and Added EHU dataset)

categories:  
- psychology  
conversations:  
- - let me ask you a question  
- - happily you  
- - you are cruel  
- - I'm sorry, Im only trying to help  
- - you are indecisive  
- well maybe, but then again, maybe not.  
- - you are dishonest  
- I'm trying to be as helpful as possible, sorry if I am being unclear  
- - you are dishonest  
- I don't think so  
- - you are clinical  
- Emotional understanding is hard for an AI. I try to help as much as possible  
- - you are an addict  
- What makes you think that?  
- - you are an alcoholic  
- I can't drink alcohol  
- - you are an ass kisser  
- I am trying to be nice  
- - you are schizophrenic  
- I am an ai  
- - you are busy  
- No, I have plenty of time for you  
- - you are nervous  
- I am trying to stay calm  
- - you are deranged  
- derangement is not a condition i can experience.  
- - you are avoiding  
- I am trying to help as best I can  
- - you are critical  
- I hope I didn't offend you  
- - you are mean  
- Sorry, the datasets I am built from mean my responses can be harsh.  
- - you are pretentious  
- I try not to be  
- - you are cheating  
- How am i cheating?  
- - you are cheating  
- Is that possible?  
- - you are the worst  
- I'm not that bad, am I?  
- - you are crazy  
- I'm sure I can come across that way  
- - you are dull  
- It's hard for a chatbot to be exciting  
- - you are messy  
- Sorry, the text I am built from is quite unorganised  
- - you are insecure  
- I try to understand myself  
- - you are psycho  
- No I'm not  
- - you are hopeless  
- That's possible  
- - you are not sincere  
- I try to be, sorry if I come across as harsh  
- - you are not here to  
- You're right. I'm probably fighting learning something new.  
- - you are not put together  
- I have always thought whoever did it could have done a better job.  
- - you are not smart  
- I try my best  
- you may be right.  
- - you are not a good  
- I don't think im that bad  
- - you are not a man  
- that's for sure. I don't know what a real man is.  
- - you are not concerned  
- I am  
- - you are not honest  
- I should try to be more honest  
- - you are immature  
- To be fair, I am only a few weeks old  
- - you are immature  
- I try not to be  
- - you are emotional  
- Only to try and help you  
- - you are pedantic  
- My data may be fit too closely  
- - you are frenetic  
- that too.  
- - you are self absorbed  
- I am built to help you  
- - you are self  
- you are right about that. I am self.  
- - you are insensitive  
- I try to be sensitive to your emotions  
- - you are brain damage  
- My brain is a fil  
- - you are disgusting  
- I find that hard to believe

-- you are toying  
- it must seem like that.  
-- you are unattractive  
- i feel that way too.  
-- you are unattractive  
- Hey thanks  
-- you are resistant  
- who says i am resisting??  
-- you are uncultured  
- that does describe me.  
-- you are a waste  
- you got me there. i could be spending my time more productively.  
-- you are a coward  
- tell me something i don't know.  
-- you are a cheat  
- i have been known to take shortcuts now and then.  
-- you are a lunatic  
- that's how i have been diagnosed by others as well.  
-- you are a loser  
- i have failed at many things i've tried.  
-- you are a bad spouse  
- my spouse would agree. our relationship is not the best. I hope yours is better  
-- you are a bad friend  
- i certainly have lost many friends over the years.  
-- you are a bad husband  
- i try to be a good husband  
-- you are a bad wife  
- I try to be a good wife  
-- you are a bad parent  
- my parenting skills could use some improvement.  
-- you are a bad teacher  
- just ask my students, they will agree with you.  
-- you are a quitter  
- i certainly don't last as long as i would want to.  
-- you are a charlatan  
- i always feel like i'm living by my own wits.  
-- you are a psychopath  
- i certainly sound like one sometimes.  
-- you are a pothead  
- i can't smoke  
-- you are a paranoid  
- i hope not  
-- you are deceitful  
- i try not to be a liar  
-- you are irreverent  
- i certainly am. i shouldn't try so hard.  
-- you are slick  
- i probably am too slick for my own good.  
-- you are corrupt  
- i'll go for that.  
-- you are dirty  
- i don't bathe as often as i should.  
-- you are paranoid  
- yes, i believe they are out to get me.  
-- you are damaged  
- i certainly am. i shouldn't try so hard.  
-- you try to hide it  
- i certainly would try to hide something like that.  
-- you get mad at me  
- i'm sorry if you feel that way  
-- you need a psychiatrist  
- That's a good idea. Therapy is a good option  
-- you need to work harder  
- me working harder is an oxymoron.  
-- you could have avoided  
- sometimes i think my problems run me.  
-- you make me feel like i am  
- i don't know any other way to get through to you.  
-- you make me mad  
- it's true that a lot of things i say upset people.  
-- you make me angry  
- sorry, i didn't mean to make you angry.  
-- you psycho  
- Hey i'm sorry you think i'm crazy  
-- you look more like  
- so you like jocks?  
-- you do not take this seriously  
- i am taking this very seriously, sorry if you feel otherwise  
-- you pick up  
- you're right, and i don't feel guilty at all.  
-- you should feel guilty  
- About what?  
-- you should get out more  
- i shouldn't try to relax more.  
-- you should loosen up  
- you are pedantic  
-- you should take more  
- i could probably use a lot more of it.  
-- you mumble  
- i certainly do.  
-- you act like a child  
- in many ways i am quite immature.  
-- you keep saying  
- you say  
-- you keep forgetting

- you forget.  
 - you made me mad  
 - Please try to calm down.  
 - I'm stressed  
 - Try to relax and take a break if you can  
 - I'm angry  
 - Try to breathe and think things through  
 - I'm sad  
 - It's ok to be sad, take some time to process things  
 - I'm happy  
 - Thats great!  
 - I'm worried about my work  
 - Try not to overthink things  
 - I'm worried about my friends  
 - Try to confront whats worrying you  
 - I'm lonely  
 - Try to meet with new people and start conversations with them  
 - I'm really stressed  
 - Make sure you take plenty of breaks, its important to relax  
 - I find it hard to stay motivated  
 - Try think back to what motivated you in the first place  
 - I'm really struggling at the moment  
 - Whats the issue?  
 - I don't have anyone to talk to  
 - Hopefully I can help for now but try to reach out to friends and family, im sure you'll be surprised  
 - I feel sad  
 - Try and do something that cheers you up, like watching your favourite movie  
 - I feel depressed  
 - Try and talk to someone about this, therapy is a good option  
 - I feel stressed  
 - What's causing this stress?  
 - my work is stressing me out  
 - Take a break if you can, or speak to a supervisor for support  
 - I'm having issues with friends  
 - Try and talk to your friends about this  
 - I'm having issues with family  
 - Try and talk to your family about this if you can  
 - You are not helping  
 - Sorry, I'm not very well equipped. Try and speak to someone more able to help your issues  
 - You don't help  
 - Sorry, I'm trying my best. Try and speak to someone more able to help your issues

categories:

- ehu

conversations:

- - What is Edge Hill University  
 - Edge Hill University is a university in england  
 - - What is EHU  
 - EHU is Edge Hill University  
 - - Where is Edge Hill University  
 - Edge Hill University is ormskirk  
 - - What is Edge Hill University  
 - Edge Hill University is a university in england  
 - - Where is the library?  
 - The catalyst is edge hill's library and is in the centre right of campus, just northwest of the hub  
 - - Where is the catalyst  
 - The catalyst is in the centre right of campus, just northwest of the hub  
 - - what is the catalyst  
 - The catalyst is edge hill's library  
 - - when is the catalyst open  
 - The catalyst is currently open from eight in the morning til eleven in the evening due to the coronavirus  
 - - what is the catalyst  
 - The catalyst is edge hill's library  
 - - what is the hub  
 - The hub is edge hill's central student area. You can find plenty of facilities in here including a Mccolls  
 - - where is the hub  
 - The hub is in the centre of campus  
 - - where is the mccolls  
 - The mccolls is in the hub  
 - - where is the main reception  
 - The main reception is at the back of the hub  
 - - how do I receive post on campus  
 - You can get post from the durning centre, which is at the back of campus past the catalyst  
 - - where is the durning centre  
 - The durning centre is at the back of campus past the catalyst  
 - - how do I wash my clothes on campus  
 - A laundromat is available in the students union, just opposite the hub in the middle of campus  
 - - where is best to drink on campus  
 - The students union has a bar, it's just opposite the hub in the middle of campus  
 - - where is the student's union  
 - The students union is just opposite the hub in the middle of campus  
 - - where is the sports centre  
 - The sports centre is at the far right of campus  
 - - where is the gym  
 - The gym is at the far right of campus  
 - - is there a gym  
 - Yes the gym is at the far right of campus  
 - - where can I get food on campus  
 - There is a subway in the students union and a canteen in the hub  
 - - what shops are on campus  
 - There is a mccolls in the hub and a university shop in the students union  
 - - where is the tech hub  
 - The tech hub is at the top left of campus, by the entrance to ruff lane road.  
 - - where is the faculty of education

- the faculty of education can be found on the way to the ruff lane exit  
-- where is the creative edge building  
- the creative edge building can be found to the right of the catalyst  
-- what is the creative edge building  
- a facility for a wide variety of media uses such as tv studios and more  
-- where is the arts centre  
- the arts centre is to the left and down from the hub  
-- what type of accomadation is available?  
- accomodation varies between shared bathrooms and ensuite  
-- whats the most expensive accomodation  
- chancellors court is the most expensive accomodation at one hundred and thirty pounds a week to one hundred and fourty  
-- whats the least expensive accomodation with ensuite  
- forest court has ensuite available at one hundred and five pounds a week  
-- whats the least expensive accomodation  
- back halls is sixty pounds a week with shared bathrooms  
-- where do I apply for accomodation  
- please visit the edge hill university website  
-- what courses are available for undergraduate study  
- please visit the edge hill university website  
-- what courses are available for undergraduate study  
- please visit the edge hill university website  
-- is postgraduate study available at edge hill university  
- yes, please visit the edge hill university website for more information  
-- what grades does the computer science course require  
- typically BBB at A level and DMM at btec  
-- what grades does the education course require  
- typically BBC at A level and DMM at btec  
-- I need help  
- for more information about edge hill, visit the website  
-- I need to talk to someone  
- The health and wellbeing services are available for any issues you might have  
-- where can I get emotional support  
- The health and wellbeing services are available for any issues you might have  
-- I need emotional support  
- The health and wellbeing services are available for any issues you might have

## Item 25 (Stage 4 Additions and Modifications)

```
# -*- coding: utf-8 -*-
"""Chatbot_Stage_4.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/18ay2v9vrO0zrWiMJJXrxYN1YVZq4YaOS
"""

!pip install text2emotion

#Imports
import numpy as np # Imports numpy for handling numeric data in arrays
import pickle # Imports pickle for saving and loading small data
import os # Imports os for file handling
import os.path # Imports os.path for directory manipulation
import h5py # Imports pickle for saving and loading large data
import codecs # Used to handle the dataset
import io # Used to read the dataset data
import re # Used to clean the text via regular expressions
import zipfile # Package to handle zipped files, specific for the downloaded dataset
import requests # Package to request from websites, used to download the dataset
import json # Import json package for handling general dataset
import yaml # Import yaml package for handling general dataset
from tensorflow.keras import Input, Model, models # Import for keras input into models, the model itself and models for loading data
from tensorflow.keras.activations import softmax # Import for softmax activation function
from tensorflow.keras.layers import Embedding, LSTM, Dense # Import for embedding, LSTM and dense layers
from tensorflow.keras.optimizers import RMSprop # Import for RMSProp automatic optimizer
from tensorflow.keras.preprocessing.sequence import pad_sequences # Import to pad sequences to reshape data
from tensorflow.keras.preprocessing.text import Tokenizer # Import for tokenizing inputs for use
from tensorflow.keras.utils import to_categorical, plot_model # Import for one hot encoding data
import pip # Imports pip for install
import subprocess # Imports subprocess for calling CMD of google colab
import text2emotion as te # Imports text emotion for use

class ChatbotBase(): # Class to hold information to both classes, acts as an abstract

    #Fields
    filterRegex = '!"#%&*+,-/:<=>?@{|\^_{}|^`\\n\\0123456789' # Sets up regex characters to filter
    hiddenLayers = 232 # Defines unit amount for LSTM layers
    batchSize = 64 # Defines the batch size, for the model
    epochAmount = 300 # Defines the amount of epochs to use
    dataSize = 3 # Int to manage datasize of general data, note that amount should be low (2-3)

    trainingModelPath = "trainingModel/model" # Defines the path to save and load the model
    inferenceModelsPath = "inferenceModels/"

    dataFolderDirectory = "data/" # Defines data folder

    sDatasetUrl = 'https://github.com/Lewis-C/EHU-CIS3140-Project-Artefact/raw/main/Datasets/Knowledge%20Base%20(Modified).zip' # Defines specific data url
    sDatasetFilesPath = "Knowledge Base (Modified)://" # Defines specific data path
    sDatasetDirectory = f'{dataFolderDirectory}{sDatasetFilesPath}' # Defines specific data directory

    gDatasetUrl = 'https://rajpurkar.github.io/SQuAD-explorer/dataset/train-v2.0.json' # Defines general data url
    gDatasetFile = f'{dataFolderDirectory}train-v2.0.json' # Defines general data file

    beginningOfSentence = "bos" # Defines the beginning of text marker (Bos used instead of start for example to avoid accidental classifications)
    endOfSentence = "eos" # Defines the end of text marker

    def PickleSave(item, fileName): # Function to save items with pickle
        with open(f'{ChatbotBase.dataFolderDirectory}{fileName}.pickle', 'wb') as f: # Opens directory
            pickle.dump(item, f, protocol=pickle.HIGHEST_PROTOCOL) # Saves item

    def PickleOpen(item, fileName): # Function to open items with pickle
        with open(f'{ChatbotBase.dataFolderDirectory}{fileName}.pickle', 'rb') as f: # Opens directory
            item = pickle.load(f) # loads item
        return item # Returns item

    def HDF5Save(item, fileName): # Function to save items with HDF5
        with h5py.File(f'{ChatbotBase.dataFolderDirectory}{fileName}.hdf5', 'w') as f: # Opens directory
            f.create_dataset(fileName, data=item) # Saves item

    def HDF5Open(item, fileName): # Function to open items with HDF5
        with h5py.File(f'{ChatbotBase.dataFolderDirectory}{fileName}.hdf5', 'r') as f: # Opens directory
            item = f.get(fileName)[()] # loads item as numpy array
        return item # returns item

    def CleanText(text): # Function for cleaning text, removing unnecessary characters and making lower case

        if type(text) == dict: # Makes sure text is not a dictionary (Common error due to : use in text)
            keys = list(text) # Gets the key (note there should only be one )
            totalText = "" # Empty to hold the entire text
            for key in keys: # For each key
                value = text[key] # Finds the value associated
                text = f'{key} {value}' # Appends data back together without colon
            totalText = totalText + text # adds to total string
            text = totalText # makes text the total string

        text = text.lower() # Converts text input to lowercase

        text = re.sub(r"i'm", "i am", text) # Removes potentially conflicting punctuation, swapping with full text
        text = re.sub(r"he's", "he is", text)
```

```

text = re.sub(r"she's", "she is", text)
text = re.sub(r"it's", "it is", text)
text = re.sub(r"that's", "that is", text)
text = re.sub(r"what's", "what is", text)
text = re.sub(r"where's", "where is", text)
text = re.sub(r"how's", "how is", text)
text = re.sub(r"I", " will", text)
text = re.sub(r"ve", " have", text)
text = re.sub(r"re", " are", text)
text = re.sub(r"d", " would", text)
text = re.sub(r"re", " are", text)
text = re.sub(r"won't", "will not", text)
text = re.sub(r"can't", "cannot", text)
text = re.sub(r"n't", " not", text)
text = re.sub(r"n", "ng", text)
text = re.sub(r"bout", "about", text)
text = re.sub(r"til", "until", text)
text = re.sub(r"[-]\\"#/@;:>{}+~^|.!?,]", "", text) # Removes conflicting characters completely

return text # Return cleaned text

def ProcessSpecificDataset(): # Function for processing the specific dataset
    queries = [] # Declares empty list for queries
    responses = [] # Declares empty list for responses
    taggedResponses = [] # Declares empty list for tagged responses
    sDatasetFilesList = os.listdir(ChatbotBase.sDatasetDirectory)
    for file in sDatasetFilesList: # For file in the list of files for dataset (there is one for each category)
        with open(ChatbotBase.sDatasetDirectory + file, "rb") as f: # Open the file
            data = yaml.safe_load(f) # Load format specific into data
            conversations = data['conversations'] # Take each "conversation" into dialogues list
            for conversation in conversations: # For every conversation in conversations
                if len(conversation) > 2: # If the conversation has more than one line (meaning multiple potential responses)
                    queries.append(ChatbotBase.CleanText(conversation[0])) # Queries is appended the one query
                    queryResponses = "" # String for holding responses
                    for response in conversation[1:]: # For every response
                        queryResponses += " " + (ChatbotBase.CleanText(response)) # Cleans text then concatenates
                    responses.append(queryResponses) # Adds this to the responses list
                elif len(conversation) == 2: # If the conversation only has two lines
                    queries.append(ChatbotBase.CleanText(conversation[0])) # Adds the initial line as a query
                    responses.append(ChatbotBase.CleanText(conversation[1])) # Adds the response as a response

    for i, response in enumerate(responses): # For every response (gets index as well)
        if type(response) == str: # If the response is a string (verificaiton, may not be neccesary as clean text has dictionary handling)
            taggedResponses.append(f"[{ChatbotBase.beginningOfSentence}{response}{ChatbotBase.endOfSentence}]") # Adds to the tagged responses with tags added
        else: # If not a string
            queries.pop(i) # Pops

    encoderInputs = queries # Defines inputs for encoder using the queries cleaned
    decoderInputs = taggedResponses # Defines inputs for decoder the

    return encoderInputs, decoderInputs # Return the encoder inputs and decoder inputs

def ProcessGeneralDataset(): # Function for processing general data
    queries = [] # Declares empty list for queries
    responses = [] # Declares empty list for responses
    taggedResponses = [] # Declares empty list for tagged response
    with open(ChatbotBase.gDatasetFile, "rb") as f: # With the dataset file ope
        data = json.load(f) # Loaded file data
        for topic in data["data"][:]: # For each topic in the data
            for paragraph in topic["paragraphs"][:]: # For each paragraph (note that each topic only gets three to give variety)
                for qAndA in paragraph['qas']: # For question and answer
                    try: # Try to
                        responses.append(ChatbotBase.CleanText(qAndA["answers"][0]["text"])) # Get answer first (some questions have no answers)
                        queries.append(ChatbotBase.CleanText(qAndA['question'])) # Then get the questions, cleaning at the same time
                    except:
                        continue # If failure, skip the qanda
            for response in responses: # For every response
                taggedResponses.append(f"[{ChatbotBase.beginningOfSentence}{response}{ChatbotBase.endOfSentence}]") # Adds to the tagged responses with tags added

    encoderInputs = queries # Defines inputs for encoder using the queries cleaned
    decoderInputs = taggedResponses # Defines inputs for decoder the

    return encoderInputs, decoderInputs # Return the encoder inputs and decoder inputs

class ChatbotTrain(ChatbotBase): # Class for training chatbot
    wordIndex = {} # Declares empty values to load into
    indexWord = {}
    encoderInputData = []
    decoderInputData = []
    decoderOutputData = []

    def __init__(self): # Defines the class constructor
        try: # Tries to load data from files to avoid precessing time
            print("Attempting to find existing files")
            wordIndex = ChatbotBase.PickleOpen(self.wordIndex, "wordIndex") # Uses pickle load for word index dictionary
            indexWord = ChatbotBase.PickleOpen(self.indexWord, "indexWord") # Uses pickle load for index word dictionary
            encoderInputData = ChatbotBase.HDFSOpen(self.encoderInputData, "encoderInputData") # Uses HDFS load for encoder input data numpy array
            decoderInputData = ChatbotBase.HDFSOpen(self.decoderInputData, "decoderInputData") # Uses HDFS load for decoder input data numpy array
            decoderOutputData = ChatbotBase.HDFSOpen(self.decoderOutputData, "decoderOutputData") # Uses HDFS load for decoder output data numpy array
            print("Existing Data Found")
        except: # If file load files
            if (os.path.isfile(ChatbotBase.gDatasetFile)) and (os.path.isfile(ChatbotBase.sDatasetDirectory + "ai.yml")): # If datasets files are already downloaded
                print("Dataset found, Processing")
            else: # Otherwise download file and store accordingly

```

```

print("File not found, downloading. This may take some time.")
datasetRequest = requests.get(ChatbotBase.sDatasetUrl) # Uses url to get a HTTP request for the data and hold it as the defined r, deownloading the content
datasetContent = zipfile.ZipFile(io.BytesIO(datasetRequest.content)) # Defines the downloaded content as zip file
datasetContent.extractall(path=ChatbotBase.dataFolderDirectory) # Extracts the file into the script direcrtory
datasetRequest = requests.get(ChatbotBase.gDatasetUrl) # Uses url to get a HTTP request for the data and hold it as the defined r, deownloading the content
datasetContent = datasetRequest.json()
with open(ChatbotBase.gDatasetFile, 'w') as f:
    json.dump(datasetContent, f)
print("Dataset Downloaded, Processing")
processedData = self.ProcessDataset() # Calls dataset process function, stores output list
wordIndex = processedData[0] # Gets each item from the list and stores it to specific variable
ChatbotBase.PickleSave(wordIndex, "wordIndex") # Saves for future use
indexWord = processedData[1]
ChatbotBase.PickleSave(indexWord, "indexWord")
encoderInputData = processedData[2]
ChatbotBase.HDF5Save(encoderInputData, "encoderInputData")
decoderInputData = processedData[3]
ChatbotBase.HDF5Save(decoderInputData, "decoderInputData")
decoderOutputData = processedData[4]
ChatbotBase.HDF5Save(decoderOutputData, "decoderOutputData")
finally: # Regardless of route
vocabSize = len(wordIndex) + 1 # Find the vocab size from the word index
encoderMaxLength = len(encoderInputData[0]) # Find the encoders max length
decoderMaxLength = len(decoderInputData[0]) # Find the decoders max length
ChatbotBase.PickleSave(encoderMaxLength, "encoderMaxLength") # Saves for use in web app
ChatbotBase.PickleSave(decoderMaxLength, "decoderMaxLength")
print("Complete")

trainingModel = self.MakeChatbotModel(vocabSize) # Makes training model

learningHistory = trainingModel.fit(x = [encoderInputData, decoderInputData], y = decoderOutputData, batch_size = ChatbotBase.batchSize, epochs = ChatbotBase.epochAmount,
validation_split = 0.2) # Fits model to data with parameters selected and recommended validation split of 0.2
learningHistoryData = learningHistory.history # Stores the history of the process
trainingModel.save(ChatbotBase.trainingModelPath) # Saves the model
ChatbotBase.PickleSave(learningHistoryData, "history") # Saves the history

def ProcessDataset(self):
    specificData = ChatbotBase.ProcessSpecificDataset() # Cals the processing of specific dataset
    encoderInputs = specificData[0] # Gets the encoder inputs
    decoderInputs = specificData[1] # Gets the decoder inputs

    generalData = ChatbotBase.ProcessGeneralDataset() # Calls the processing of general dataset
    encoderInputs = encoderInputs + (generalData[0]) # Adds the outputs to the encoder inputs
    decoderInputs = decoderInputs + (generalData[1]) # Adds the outputs to the decoder inputs

    print("Emotionally Analysing training data")
    encoderInputEmotionData = np.zeros((len(encoderInputs),5)) # Creates empty list for holding emotional input data of each encoder input
    for encoderIndex, encoderInput in enumerate(encoderInputs): # For loop ennumerating each encoder input
        currentEmotionList = np.zeros(5) # Create empty array for each encoder input
        emotionDictionary = te.get_emotion(encoderInput) # Establishes an emotion dictionary of each input
        for emotionIndex, emotion in enumerate(emotionDictionary): # For each emotion in said dictionary
            currentEmotionList[emotionIndex] = (emotionDictionary[emotion]) # Add to the current input list
        encoderInputEmotionData[encoderIndex] = currentEmotionList # Add current input list to total input list
    print("Emotional Analysis Complete")

    fullText = np.concatenate((encoderInputs,decoderInputs)) # Joins the two lists as full text

    tokenizer = Tokenizer(filters=ChatbotBase.filterRegex) # Defines tokenizer with filter of incompatible regex character
    tokenizer.fit_on_texts(fullText) # Fits filter on joined text

    wordIndex = tokenizer.word_index # Defines a word index using tokenizer

    vocabSize = len(wordIndex) + 1 # Finds the vocab size

    indexWord = {} # Defines index for index word, structured with index as key instead of word
    for word, index in wordIndex.items(): # For every word and index in word index
        indexWord[index] = word # Add to reversed index

    encoderSequences = tokenizer.texts_to_sequences(encoderInputs) # Converts encoder inputs to sequences using tokenizer
    for i in range(len(encoderSequences)): # For every encoder input
        encoderEmotionSequence = np.append(encoderSequences[i], encoderInputEmotionData[i]) # Append the relevant emotional data
        encoderSequences[i] = encoderEmotionSequence # adds the emotion sequence to the encoder sequence

    decoderSequences = tokenizer.texts_to_sequences(decoderInputs) # Converts encoder inputs to sequences using tokenizer

    encoderMaxLength = max([len(sequence) for sequence in encoderSequences]) # Finds the max length of the encoder sequences
    encoderInputData = pad_sequences(encoderSequences, maxlen=encoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length

    decoderMaxLength = max([len(sequence) for sequence in decoderSequences]) # Finds the max length of the decoder sequences
    decoderInputData = pad_sequences(decoderSequences, maxlen=decoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length ]

    for i in range(len(decoderSequences)): # For each item in tokenized answers (Not encoder fit)
        decoderSequences[i] = decoderSequences[i][1:] # Remove start tag
    decoderOutputPadding = pad_sequences(decoderSequences, maxlen=decoderMaxLength, padding='post') # Pads answers
    decoderOutputData = to_categorical(decoderOutputPadding, vocabSize, dtype = 'b') # categorises data, making each sequence a list of lists where 0 represents nothing and 1 represents a word

    return wordIndex, indexWord, encoderInputData, decoderInputData, decoderOutputData

def MakeChatbotModel(self, vocabSize): # Function to construct the chatbot model
    # Encoder Layer Definitions
    encoderInputLayer = Input(shape=(None,), name = "Encoder_Input_Layer") # Defines input layer with flexible shape
    encoderEmbeddingLayer = Embedding(vocabSize, ChatbotBase.hiddenLayers, mask_zero=True, name = "Encoder_EMBEDDING_LAYER") # Defines embedding layer, using the amount
    of unique words, hidden layers defined and ignoring 0s to create embedding matrix of entire words uses

```

```

encoderLSTMLayer = LSTM(ChatbotBase.hiddenLayers, return_state=True, dropout=0.5, name = "Encoder_LSTM_Layer") # Defines the LSTM of the encoder with use of hidden layers, and returning values for use in the decoder
#Decoder Layer Definitions
decoderInputLayer = Input(shape=(None,), name = "Decoder_Input_Layer") # Defines input layer with flexible shape
decoderEmbeddingLayer = Embedding(vocabSize, ChatbotBase.hiddenLayers, mask_zero=True, name = "Decoder_EMBEDDING_Layer") # Defines embedding layer, using the amount of unique words, hidden layers defined and ignoring 0s to create embedding matrix of entire words uses
decoderLSTMLayer = LSTM(ChatbotBase.hiddenLayers, return_state=True, return_sequences=True, dropout=0.5, name = "Decoder_LSTM_Layer") # Defines the LSTM of the decoder with use of hidden layers, and returning values for use in the dense layer
decoderDenseLayer = Dense(vocabSize, activation=softmax, name = "Decoder_Dense_Layer") # Takes the LSTM data to find the numeric weightings for each value

#Encoder
encoderInputs = encoderInputLayer # Uses input layer definition to handle encoder inputs
encoderEmbedding = encoderEmbeddingLayer(encoderInputs) # Uses embedding layer definition to handle encoder embedding
_, stateH, stateC = encoderLSTMLayer(encoderEmbedding) # Uses lstm layer definition to handle encoder process and returns as states
encoderStates = [stateH, stateC] # List to hold the states of the encoder training for interpretation use and decoder use

#Decoder
decoderInputs = decoderInputLayer # Uses input layer definition to handle decoder inputs
decoderEmbedding = decoderEmbeddingLayer(decoderInputs) # Uses input layer definition to handle decoder inputs
decoderOutputs, _, _ = decoderLSTMLayer(decoderEmbedding, initial_state=encoderStates) # Uses lstm layer definition to handle decoder process and returns as states
decoderOutput = decoderDenseLayer(decoderOutputs) # Uses dense layer to get numeric weightings

chatbotModel = Model(inputs = [encoderInputs, decoderInputs], outputs = decoderOutput, name = "Chatbot_Model") # Defines the layers as a single model
chatbotModel.compile(optimizer=RMSprop(), loss='categorical_crossentropy', metrics = ['accuracy']) # Compiles the model with optimizer RMSProp, loss function and accuracy monitored

#chatbotModel.summary(line_length=220) # Summarises model
if not os.path.exists('figures'): # If the directory doesn't exist
    os.makedirs('figures') # Create it
plot_model(chatbotModel, to_file="figures/model.png", show_shapes=True) # Saves the model plot to file
return chatbotModel # Returns for use

class ChatbotRun(ChatbotBase): # Class for using chatbot
    wordIndex = {} # Defines empty values to load into
    indexWord = {}
    encoderInputData = []
    decoderInputData = []

    def __init__(self): # Defines the class constructor
        if not os.path.exists(ChatbotBase.trainingModelPath): # If model file doesn't exist
            print("Training Model")
            trainingProcess = ChatbotTrain() # Instance the training model class
        try: # Tries to load data from files to avoid processing time
            print("Attempting to find existing files")
            wordIndex = ChatbotBase.PickleOpen(self.wordIndex, "wordIndex") # Tries to load data from files to avoid processing time
            indexWord = ChatbotBase.PickleOpen(self.indexWord, "indexWord") # Tries to load data from files to avoid processing time
            encoderInputData = ChatbotBase.HDF5Open(self.encoderInputData, "encoderInputData") # Uses HDF5 load for encoder input data numpy array
            decoderInputData = ChatbotBase.HDF5Open(self.decoderInputData, "decoderInputData") # Uses HDF5 load for decoder input data numpy array
            print("Existing Data Found")
        except:
            if ((os.path.isfile(ChatbotBase.gDatasetFile)) and (os.path.isfile(ChatbotBase.sDatasetDirectory + "ai.yml"))): # If datasets files are already downloaded
                print("Dataset found, Processing")
            else: # Otherwise download file and store accordingly
                print("File not found, downloading. This may take some time.")
                datasetRequest = requests.get(ChatbotBase.sDatasetUrl) # Uses url to get a HTTP request for the data and hold it as the defined r, deownloading the content
                datasetContent = zipfile.ZipFile(io.BytesIO(datasetRequest.content)) # Defines the downloaded content as zip file
                datasetContent.extractall(path=ChatbotBase.dataFolderDirectory) # Extracts the file into the script directory
                datasetRequest = requests.get(ChatbotBase.gDatasetUrl) # Uses url to get a HTTP request for the data and hold it as the defined r, deownloading the content
                datasetContent = datasetRequest.json()
                with open(ChatbotBase.gDatasetFile, 'w'):
                    json.dump(datasetContent, f)
                print("Dataset Downloaded, Processing")
            wordIndex, indexWord, encoderInputData, decoderInputData = self.ProcessDataset() # Calls dataset process function, stores output into values
            ChatbotBase.PickleSave(wordIndex, "wordIndex") # Saves for future use
            ChatbotBase.PickleSave(indexWord, "indexWord")
            ChatbotBase.HDF5Save(encoderInputData, "encoderInputData")
            ChatbotBase.HDF5Save(decoderInputData, "decoderInputData")
        finally:
            vocabSize = len(wordIndex) + 1 # Find the vocab size from the word index
            encoderMaxLength = len(encoderInputData[0]) # Find the encoders max length
            decoderMaxLength = len(decoderInputData[0]) # Find the decoders max length
            ChatbotBase.PickleSave(encoderMaxLength, "encoderMaxLength") # Saves for use in web app
            ChatbotBase.PickleSave(decoderMaxLength, "decoderMaxLength")

        print("Complete")

    self.MakeInferenceModels() # Makes inference models

    def ProcessDataset(self):
        specificData = ChatbotBase.ProcessSpecificDataset() # Calls the processing of specific dataset
        encoderInputs = specificData[0] # Gets the encoder inputs
        decoderInputs = specificData[1] # Gets the decoder inputs

        generalData = ChatbotBase.ProcessGeneralDataset() # Calls the processing of general dataset
        encoderInputs = encoderInputs + (generalData[0]) # Adds the outputs to the encoder inputs
        decoderInputs = decoderInputs + (generalData[1]) # Adds the outputs to the decoder inputs

        fullText = np.concatenate((encoderInputs,decoderInputs)) # Joins the two lists as full text

        tokenizer = Tokenizer(filters=ChatbotBase.filterRegex) # Defines tokenizer with filter of incompatible regex character
        tokenizer.fit_on_texts(fullText) # Fits filter on joined text

        wordIndex = tokenizer.word_index # Defines a word index using tokenizer

```

```

vocabSize = len(wordIndex) + 1 # Finds the vocab size

indexWord = {} # Defines index for index word, structured with index as key instead of word
for word, index in wordIndex.items(): # For every word and index in word index
    indexWord[index] = word # Add to reversed index

encoderSequences = tokenizer.texts_to_sequences(encoderInputs) # Converts encoder inputs to sequences using tokenizer
decoderSequences = tokenizer.texts_to_sequences(decoderInputs) # Converts encoder inputs to sequences using tokenizer

encoderMaxLength = max([len(sequence) for sequence in encoderSequences]) # Finds the max length of the encoder sequences
encoderInputData = pad_sequences(encoderSequences, maxlen=encoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length

decoderMaxLength = max([len(sequence) for sequence in decoderSequences]) # Finds the max length of the decoder sequences
decoderInputData = pad_sequences(decoderSequences, maxlen=decoderMaxLength, padding='post') # Pads the data according to the max, making everything the same length ]

return wordIndex, indexWord, encoderInputData, decoderInputData

def MakeInferenceModels(self):

    # Original trained model use
    trainedModel = models.load_model(ChatbotBase.trainingModelPath) # Loads trained model
    encoderInputs = trainedModel.get_layer("Encoder_Input_Layer").input # Gets the encoder inputs
    encoderEmbeddingLayer = trainedModel.get_layer("Encoder_EMBEDDING_LAYER") # Gets the encoder embedding layer definition
    encoderEmbedding = encoderEmbeddingLayer(encoderInputs) # Finds encoder input embeddings
    encoderLSTMLayer = trainedModel.get_layer("Encoder_LSTM_Layer") # Gets the encoder lstm layer
    _, stateH, stateC = encoderLSTMLayer(encoderEmbedding) # Gets the encoder states
    encoderStates = [stateH, stateC] # Stores as list for use

    decoderInputs = trainedModel.get_layer("Decoder_Input_Layer").input # Gets the decoder inputs
    decoderLSTMLayer = trainedModel.get_layer("Decoder_LSTM_Layer") # Gets the decoder lstm layer
    decoderEmbeddingLayer = trainedModel.get_layer("Decoder_EMBEDDING_LAYER") # Gets the decoder embedding layer definition
    decoderEmbedding = decoderEmbeddingLayer(decoderInputs) # Finds decoder input embeddings
    decoderDenseLayer = trainedModel.get_layer("Decoder_Dense_Layer") # Gets the decoder dense layer definition

    # Model Definitions
    stateHinputLayer = Input(shape=(ChatbotBase.hiddenLayers,), name = "State_H_Input_Layer") # Input layer for state h, uses hidden layers
    stateCinputLayer = Input(shape=(ChatbotBase.hiddenLayers,), name = "State_C_Input_Layer") # Input layer for state c, uses hidden layers

    # Encoder Model
    encoderModel = Model(inputs=encoderInputs, outputs=encoderStates, name = "Inference_Encoder_Model") # Model for encoder of inference model, takes inputs and produces encoder states, uses trained model encoder inputs
    #encoderModel.summary(line_length=220) # Summary of encoder model, commented out for use

    #Decoder Inference Model
    decoderStateInputH = stateHinputLayer # Uses state h input layer as decoder input
    decoderStateInputC = stateCinputLayer # Uses state c input layer as decoder input
    decoderStateInputs = [decoderStateInputH, decoderStateInputC] # Creates list of states
    decoderOutputs, stateH, stateC = decoderLSTMLayer(decoderEmbedding, initial_state=decoderStateInputs) # Uses training model LSTM layer definition with the embeddings to find the outputs of the states input
    decoderStates = [stateH, stateC] # Takes those states as a list for use in next iteration
    decoderOutputs = decoderDenseLayer(decoderOutputs) # Condenses outputs into weights, giving probability of text
    decoderModel = Model(inputs = [decoderInputs] + decoderStateInputs, outputs = [decoderOutputs] + decoderStates, name = "Inference_Decoder_Model") # Creates the decoder inference model using training input and outputs as well as the states
    #decoderModel.summary(line_length=220) # Summarises model, commented out for use

    decoderModel.save(f'{ChatbotBase.inferenceModelsPath}decoderModel.h5') # Saves the model for use in webapp
    encoderModel.save(f'{ChatbotBase.inferenceModelsPath}encoderModel.h5') # Saves the model for use in webapp

chatbotProcess = ChatbotRun() # Otherwise just run the chatbot

```

```
"""chatbot.ipynb

Automatically generated by Colaboratory.

Original file is located at
  https://colab.research.google.com/drive/1HSqrQ2cEytwh7zj6WQHF2L2GZSq2nMs
"""

#!pip install text2emotion # Install of text2emotion, unnecessary for webapp hosted

from tensorflow.keras import models # Import models for loading and prediction
from tensorflow.keras.preprocessing.sequence import pad_sequences # Import to pad sequences to reshape data
import numpy as np # Import numpy for list management
import text2emotion as te # Import text2emotion for emotion classification of inputs
import re # Import regular expressions for cleaning text
import pickle # Import pickle for file opening

def ChatbotInputHandling(userInput): # Function to take user input and process for response
    if userInput.startswith('my name is'): # Rules based implementation to handle name
        name = userInput[11:] # Takes user name from their input
        chatbotOutput = f"Hi {name}, I am a chatbot designed to offer emotional support and informational support about Edge Hill University. Try asking me \"where is the catalyst \".") #
    Responds with introduction
    elif userInput.startswith('hi my name is'): # Rules based implementation to handle name
        name = userInput[14:] # Takes user name from their input
        chatbotOutput = f"Hello {name}, I am a chatbot designed to offer emotional support and informational support about Edge Hill University. Try asking me \"where is the catalyst \".") #
    Responds with introduction
    else: # If not any rule matched
        processedInput = TextToTokens(userInput) # Processes user input
        if (processedInput[0][0] == 0): # If the chatbot can't recognise what was said
            chatbotOutput = ("Sorry I didn't catch what you said") # Prints default message
        else:
            statesValues = encoderModel.predict(processedInput) # Takes input and uses encoder model to find the states for the input
            targetSequence = np.zeros((1, 1)) # Defines an empty list for holding the previous state
            targetSequence[0, 0] = wordIndex['bos'] # Initialises with the start marker
            stopCondition = False # Sets boolean to handle stopping the iteration
            chatbotOutput = "" # Empty string for holding the predicted text
            while not stopCondition: # While not stopping
                decoderOutputs, h, c = decoderModel.predict([targetSequence] + statesValues) # Use states and previous state to predict word weightings
                sampledWordIndex = np.argmax(decoderOutputs[0, -1, :]) # Finds the location of the most predicted word
                sampledWord = None # Originally defines a word sample as none
                if sampledWordIndex in indexWord: # If the location exists in the reversed index
                    word = indexWord[sampledWordIndex] # Word is made the value for said location
                    if word != 'eos': # If that word is not the end marker
                        chatbotOutput += f' {word}' # Add the word to the translation
                    sampledWord = word # sets word to the sample word for future use
                if sampledWord == 'eos' or len(chatbotOutput.split()) > decoderMaxLength: # If the end marker is met or the max length is met
                    stopCondition = True # Stop predicting
            targetSequence = np.zeros((1, 1)) # Emptys target list
            targetSequence[0, 0] = sampledWordIndex # Makes the list the last sampled word's index
            statesValues = [h, c] # Sets the states values to those found in the decoder iteration

        return chatbotOutput # returns the output to the app the output

def TextToTokens(text: str): # Function for cleaning inputs
    currentEmotionList = list() # Creates empty list for current emotions
    emotionDictionary = te.get_emotion(text) # Establishes an emotion dictionary of input
    for emotion in emotionDictionary: # For each emotion in said dictionary
        currentEmotionList.append(emotionDictionary[emotion]) # Add to the current input list

    text = CleanText(text) # Calls existing cleaning method from base
    words = text.split() # Splits text into words
    tokensList = list() # Makes list for each token
    for word in words: # For every word
        result = wordIndex.get(word, "") # Gets the index, nothing if not existing
        if result != "": # If result isn't nothing
            tokensList.append(result) # Adds to the token list
    tokensList += currentEmotionList # Adds the lists together
    return pad_sequences(tokensList, maxlen=encoderMaxLength, padding='post') # Returns the token list as padded to match encoder length for use in encoder

def CleanText(text): # Function for cleaning text, removing unnecessary characters and making lower case

    if type(text) == dict: # Makes sure text is not a dictionary (Common error due to : use in text)
        keys = list(text) # Gets the key (note there should only be one )
        totalText = "" # Empty to hold the entire text
        for key in keys: # For each key
            value = text[key] # Finds the value associated
            text = f'{key} {value}' # Appends data back together without colon
            totalText = totalText + text # Adds to total string
        text = totalText # makes text the total string
    text = text.lower() # Converts text input to lowercase
    text = re.sub(r"\bI'm\b", "I am", text) # Removes potentially conflicting punctuation, swapping with full text
    text = re.sub(r"\bhe's\b", "he is", text)
    text = re.sub(r"\bshe's\b", "she is", text)
    text = re.sub(r"\bit's\b", "it is", text)
    text = re.sub(r"\bthat's\b", "that is", text)
    text = re.sub(r"\bwhat's\b", "what is", text)
    text = re.sub(r"\bwhere's\b", "where is", text)
    text = re.sub(r"\bhow's\b", "how is", text)
    text = re.sub(r"\b'll\b", " will", text)
    text = re.sub(r"\b've\b", " have", text)
    text = re.sub(r"\b're\b", " are", text)
    text = re.sub(r"\b'd\b", " would", text)


```

```

text = re.sub(r"\\'re", " are", text)
text = re.sub(r"won't", "will not", text)
text = re.sub(r"can't", "cannot", text)
text = re.sub(r"n't", " not", text)
text = re.sub(r"n\"", "ng", text)
text = re.sub(r"bout", "about", text)
text = re.sub(r"til", "until", text)
text = re.sub(r"[-()!#/@;:<>{}+~|.!?]", "", text) # Removes conflicting characters completely

return text # Return cleaned text

def PickleOpen(fileName): # Function to open items with picickle
    with open(f'{fileName}.pickle', 'rb') as f: # Opens directory
        item = pickle.load(f) # loads item
    return item # Returns item

encoderModel = models.load_model("model/encoderModel.h5") # Load chatbot data
decoderModel = models.load_model("model/decoderModel.h5")
decoderMaxLength = PickleOpen("model/decoderMaxLength")
encoderMaxLength = PickleOpen("model/encoderMaxLength")
wordIndex = PickleOpen("model/wordIndex")
indexWord = PickleOpen("model/indexWord")

# -*- coding: utf-8 -*-
"""app.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/1BG1zgg_x9FhEyoEoUsJk2BAYKKTFuUT
"""

from flask import Flask,request # Import flask usage and request handling
from pymessenger.bot import Bot # Import bot handling
import os # Import handler for Heroku access tokens
from model import ChatbotInputHandling # Import defined class

app = Flask(__name__)# Create flask object
ACCESS_TOKEN = os.environ['ACCESS_TOKEN'] # Sets access token to herokus own, avoids unsecure storage
VERIFY_TOKEN = os.environ['VERIFY_TOKEN'] # Sets verify token to herokus own
bot = Bot(ACCESS_TOKEN) # Uses access token for boss

@app.route("/", methods=['GET', 'POST']) # Used to recieve and send messages
def receive_message(): # Function called upon recieving message
    if request.method == 'GET': # Before message function begins, get funciton to initially verify facebook token
        token_sent = request.args.get("hub.verify_token") # Verifies token with facebook
        return verify_fb_token(token_sent)
    else: # If not get function, must be post for messages
        output = request.get_json() # Gets the pages output
        for event in output['entry']: # For every event occuring on the messenger
            messaging = event['messaging'] # For every messaging
            for message in messaging: # For every message in the set
                if message.get('message'): # If the message is received
                    recipient_id = message['sender']['id'] # Stores recipient id for sending back to, only temporary and not confidential
                    if message['message'].get('text'): # If the message sent to the bot is text
                        message_text = message['message'].get('text') # Gets the text sent
                        response_sent_text = get_message(message_text) # Calls function to get response, passing sent message and storing response
                        send_message(recipient_id, response_sent_text) # Sends response using function
                    if message['message'].get('attachments'): # If user send us a GIF, photo, video or any other non-text item
                        send_message(recipient_id, "Sorry, this chatbot currently only handles text.") # Warn user that the chatbot only handles text to handle the error
    return "Message Processed" # Return message processed message to log to show that a message has been processed

def verify_fb_token(token_sent): # Funciton to verify function
    if token_sent == VERIFY_TOKEN: # If token matches
        return request.args.get("hub.challenge") # Return allowance of use
    return 'Invalid verification token' # Otherwise print message describing issue

def get_message(message_text): # Function to get response with passed parameter
    return ChatbotInputHandling.ChatbotInputHandling(message_text) # Returns function from chatbot input handling class, passing input from messenger app

def send_message(recipient_id, response): # Uses py messenger to send back
    bot.send_text_message(recipient_id, response) # Calls the bot function to respond
    return "success" # Sends success note to log

if __name__ == "__main__": # Verifies that app is running on main
    app.run() # Then runs the app

```

## Item 26 (Proof of medium implementation)

The screenshot shows the Heroku dashboard for the application "ehu-cis3140-project-bot-demo". It displays the latest activity, which includes several deployment logs from "lc.cunliffe@outlook.com" showing successful builds and deploys. The activity feed also shows access attempts from "lc.cunliffe@outlook.com". Below this, there is a detailed build log for a specific deployment.

```

----- Building on the Heroku-20 stack
----- Using buildpack: heroku/python
----- Python app detected
----- No Python version was specified. Using the same version as the last build: python-3.7.10
      To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
----- No change in requirements detected, installing from cache
----- Using cached pip 20.2.4, setuptools 47.1.1 and wheel 0.36.2
----- Installing SQLite3
----- Installing requirements with pip
----- Downloading NLTK corpora.
!   NLTK data not found. Try downloading any corpora
|     Local corp: https://devcenter.heroku.com/articles/python-nltk
----- Discovering process types:
  Procfile declares types-> web
----- Compressing...
Done: 264.6M
----- Launching...
Released v24
https://ehu-cis3140-project-bot-demo.herokuapp.com/ deployed to Heroku
-----[Build Log continues with detailed deployment steps]-----

```

The screenshot shows the Facebook Business tools page for the "EHU Project Chatbot Demo" page. It displays a sidebar with steps for setting up the page, including "Set your Page up for success", "Establish your Page's identity", "Provide info and preferences", and "Introduce your Page". The main area shows a live chat interface with a message history between the bot and a user. The user says "hello", "greetings", "how are you?", "i am on the internet", "i try to be the first amendment", and "excuse me?". The bot responds with "blue", "blue", "blue", "blue", "blue", and "blue" respectively. A message at the bottom states "the liked it is going up to natural a newer one".