A picture containing text, font, symbol, graphics

Description automatically generated

COMP390 2020/21

A sensitive chatbot with

capabilities to detect

user emotion

Student Name:

Sebastian-Andrei Constantin

Student ID:

201478713

Supervisor Name: Dr Qaiser Fawada

DEPARTMENT OF COMPUTER SCIENCE

University of Liverpool Liverpool L69 3BX

**Acknowledgements**

A picture containing text, font, symbol, graphics

Description automatically generated

COMP390 2022/2023

A sensitive chatbot with

capabilities to detect

user emotion

DEPARTMENT OF COMPUTER SCIENCE

University of Liverpool Liverpool L69 3BX

**Abstract**

This dissertation explores the development of an emotion-detecting chatbot, equipped to recognize and respond to Paul Ekman’s 6 basic emotions: anger, disgust, fear, joy, sadness, and surprise. The chatbot employs a combination of pattern-based and emotion-based response selection mechanisms. The study offers a comprehensive review of literature in AI, Neural Networks, Deep Learning, NLP, and sentiment analysis, highlighting the growing importance of emotional recognition in industries like customer service and mental health. The design and implementation sections describe the creation of the chatbot from data pre-processing to model training highlighting the usage of NLP and deep learning techniques. Furthermore, the dissertation evaluates the performance of the emotion detection model based on metric like accuracy, precision, recall, and loss coupled with user feedback.

**Table of Contents**

Acknowledgments

Abstract

List of Figures

Acknowledgments

Chapter 1: Introduction

* 1. Introduction 10
  2. Background of the research 10
  3. Rationale 11
  4. Research aim and objectives 12
  5. Dissertation outline 12
  6. Summary 14

Chapter 2: Literature review

2.1 Introduction 15

2.2 Existing Chatbot Solutions and Applications 15

2.2.1 XiaoIce 15

2.2.2 IBM Watson Assistant 17

2.3 Sentiment Analysis Techniques in Chatbots 19

2.3.1 NLP (Natural Language Processing) 19

2.3.2 Deep learning 20

2.3.3Sentiment analysis 21

2.4 Summary 23

Chapter 3: Design and Implementation

3.1 Design 23

3.1.1 Introduction 23

3.1.2 UI 24

3.1.3 Classes 26

3.1.4 Data Flow 29

3.1.5 Summary 31

3.2 Implementation 31

3.2.1 Introduction 31

3.2.2 Chatbot Model 31

3.2.3 GUI 36

3.2.4 Problems encountered during the development of the chatbot 38

3.2.5 Summary 39

Chapter 4: Testing and Evaluation

4.1 Introduction 40

4.2 Model performance evaluation 40

4.3 Beta Testing and User Feedback 42

4.4Analysis and Justification of user feedback 44

Chapter 5:

**List of Figures**

**Figure 1:** StartWindow UI

**Figure 2:** Chatbot UI

**Figure 3:** Class Diagram

**Figure 4:** Data Flow Diagram

**Figure 5:** Data Tokenization

**Figure 6:** Model Architecture

**Figure 7:** Configure and Training

**Figure 8:** Pattern based response

**Figure 9:** Emotion based response

**Figure 10:** Validation Accuracy and Loss

**Figure 11:** Number of Correct Emotions vs. Messages Sent

**Chapter 1: Introduction**

* 1. **Introduction**

In the era of digital communication, chatbots have become a vital feature of numerous applications, providing users with efficient, knowledgeable and interactive experiences. As AI-powered systems are constantly evolving and attracting a high demand, both for professional, as well as personal use, there is a rising need for more complex chatbots that are not only able to process the textual content of user input, but also recognize and respond depending on the emotion portrayed. Therefore, the ability to recognize emotion has been proven to enhance user experience and allow for more engaging interactions. (Frąckiewicz, 2023)

The dissertation presents the development and implementation of an emotion-detecting chatbot based on Paul Ekman’s 6 basic emotions (anger, disgust, fear, joy, sadness, surprise). The chatbot incorporates a combination of pattern-based and emotion-based response generation mechanisms to cater to a wide range of user inputs. The introduction provides an overview of the research background (1.2), rationale (1.3) behind the development of the project and discusses the research aim and objectives. (1.4) Lastly, the dissertation outline is provided (1.5) along with the summary. (1.6)

**1.2 Background of the research**

Providing consumers with effective and interactive experiences, chatbots have grown to be a crucial part of many applications in recent years (Shawar & Atwell, 2007). There is a growing need to create increasingly complex chatbots as these AI-powered systems advance so that they can not only read the textual content of user inputs but also identify and react to the underlying emotions portrayed in them (Ghandeharioun et al., 2019). The conversational capabilities of a chatbot can be greatly improved by the capacity to recognize emotions, enabling more interesting conversations.

The field of sentiment analysis, a sub-discipline of natural language processing (NLP), focuses on extracting subjective information, such as emotions or opinions, from textual data (Liu, 2012). By incorporating sentiment analysis techniques into chatbot systems, developers can create more advanced and emotionally intelligent conversational agents that can understand and respond to user’s emotional states (Cambria & White, 2014). The combination of NLP techniques and machine learning algorithms, such as deep learning, has led to significant advancements in the development of chatbots that can process and generate human-like responses (Cho et. al., 2014).

* 1. **Rationale**

The motivation behind this research comes from the importance of chatbots in digital communication (Huang and Rust, 2018) and the potential benefit of incorporating emotion recognition capabilities in such systems. Emotionally intelligent chatbots can lead to improved user satisfaction, more engaging conversations and a better overall conversational experience (Calvo & D’Mello, 2010). The ability to recognize and respond to user’s emotions can also enable the chatbot to adapt its responses based on the detected emotion, making it more versatile and effective.

Moreover, according to Rathnayaka et. al. (2022), the integration of emotion detection in chatbots can have broader implications such as providing valuable insights into user behaviour, enabling more effective customer support and even assisting in mental health applications. For instance, emotionally intelligent chatbots can be used in therapy (Lee et. al., 2017) and counselling services to provide support and track patients’ emotional states. Therefore, the development of an emotional-detecting chatbot can contribute significantly to the ongoing advancement in AI-driven conversational systems and sentiment analysis.

**1.4 Research aim and objectives**

The primary aim of this research is to design and implement an emotion-detecting chatbot that can recognize the 6 emotions, as stated above, and respond based on the detected emotion.

To achieve this aim, the following objectives have been set:

* Conduct an extensive literature review to investigate existing techniques and approaches in sentiment analysis and natural language processing.
* Analyse and compare different algorithms and techniques to identify the most suitable methods for emotion detection.
* Design a chatbot architecture that incorporates emotion detection and generates responses based on the user’s emotional state.
* Implement the chatbot using appropriate programming language, libraries and tools, ensuring that the system is efficient and scalable.
* Evaluate the chatbot’s effectiveness in detecting emotions.
* Identify potential future developments and improvements for the emotion-detection chatbot, exploring innovative techniques and applications that can further enhance its capabilities.
  1. **Dissertation outline**

The dissertation is structured as follows:

Chapter 1: Introduction

Chapter 2: Literature Review – This chapter provides an in-depth analysis of the relevant literature in the field of chatbots, natural language processing, sentiment analysis, deep learning, and emotion detection. It discusses the existing techniques, approaches and challenges in these areas and highlights their significance in the context of this research. This chapter also presents an overview of the related work and the state-of-the-art in chatbot systems and emotion recognition. It will be organised in two sub-sections, each addressing a separate aspect of sentiment capabilities in chatbots. The first sub-section will focus on already existing developed software and the current degree of succession in sentiment detection and the second sub-section will explore the existing research on concepts, techniques, implementations and testing methods available.

Chapter 3: Design and Implementation – This chapter delves into the design and implementation of the emotion-detecting chatbot. It presents the system architecture, components, and functionality, providing insights into the development process and the technologies used. This chapter also discusses the integration of various techniques for emotion detection, as well as the challenges faced during the implementation phase.

Chapter 4: Testing and Evaluation – This chapter focuses on the testing and evaluation of the chatbot’s effectiveness in detecting emotions and giving responses. It presents various examples and performance metrics to assess the chatbot’s capabilities. The evaluation process includes quantitative analysis such as Survey research, examining the system’s performance, usability, and user satisfaction.

Chapter 5: Conclusion – This chapter serves to summarize the research by reiterating the aims and objectives, discussing and evaluating to what extent they were achieved, discussing the shortcomings of the project, providing avenues for future research work on the topic and highlighting key findings from the main chapters.

* 1. **Summary**

This introductory chapter has provided an overview of the research context, motivation, rationale and aim and objectives for the development of an emotion-detecting chatbot. By incorporating sentiment analysis techniques, the chatbot aims to enhance the conversational experience by understanding the user’s emotional state. The dissertation is structured into six chapters with each chapter addressing a specific aspect of the research process, from literature review to design, implementation and evaluation.

**Chapter 2: Literature Review**

**2.1 Introduction**

Chapter 2 aims to provide an overview and understanding of the existing research developed around Artificial Intelligence linked with sentiment analysis based on text. More precisely, the research covered will be NLP (Natural Language Processing), Neural networks, Deep learning techniques, sentiment analysis as well as a discussion on already existing pieces of software (chatbots). The growing importance of understanding and interpreting user’s emotions in various industries, such as customer service, marketing, and mental health, highlights the significance of researching this topic (Cambria & White, 2014; Hutto & Gilbert, 2014). Moreover, examining the current research in the topic provides a solid ground for the software development conducted as part of this dissertation. Conducting a review of the existing literature helps understand and justify the underlying research behind the code written for the software, supporting and justifying the decisions made during development and any challenges faced.

**2.2 Existing Chatbot Solutions and Applications**

**2.2.1 XiaoIce**

XiaoIce is an empathetic social chatbot developed by Microsoft in 2014, is one of the most popular chatbots globally, designed to provide an emotional connection for the human need for communication and affection. With over 660 million active users across China, Japan, the US, India, and Indonesia, XiaoIce serves as an artificial intelligence companion.

Zhou et al. (2020) explains that XiaoIce is built on an empathic computing framework, enabling the chatbot to recognize, understand, and respond to user sentiments, intentions, emotions, and states. As an open-domain social chatbot, XiaoIce outperforms many early social chatbots and AI personal assistants such as Siri, Cortana, Amazon Alexa, and Google Assistant in various aspects of building human-like relationships.

According to Zhou et.al (2020), XiaoIce was implemented based on various methods and techniques to enhance its capabilities in terms of intelligence (IQ) and emotion (EQ) quotient. One of the main methods used in the development of this empathetic chatbot is called Core Chat, which is formed of two elements: General Chat and Domain Chat. General Chat is focused on engaging on conversations covering a wide range of topics, while Domain Chat is focused on deep conversations on a specific domain such as music, celebrities etc.

Therefore, XiaoIce makes a valuable use of emotion recognition capabilities through its empathetic computing module. This is of crucial importance as to allow chatbots to interpret and respond to users’ emotions effectively and accurately, emotion recognition needs to be a primary focus in the development and implementation process. (Calvo and D’Mello, 2010) What this computing module does at its core is: uses query and response empathy vectors during the response generation process, which consists of candidate generation and ranking. (Zhou et.al., 2020)

Candidate generation is concerned with the creation or production of a list of potential responses either extracted from databases of human-generated responses or from a neural response generator. Ranking simply involves selecting the most fitting response from the list based on factors such as relevance and emotional congruence. (Zhou etl.al., 2020)

According to Picard (n.d.), emotion recognition in AIs is not only important for improving overall user satisfaction and engagement through a deeper established connection, but also for decision-making capabilities. Neurological studies indicate that decision-making without emotion can be just as impaired and faulty as decision-making with too much emotion, suggesting an implication that integrating emotion recognition in computers can give them the capability to make more intelligent decisions.

Other techniques and methods that XiaoIce uses, as per the study published by Zhou et.al. (2020) are: retrieval-based response generation; a complex architecture that integrates the General Chat, Domain Chat and empathetic computing module; maintains a dialogue history and uses it to generate contextually relevant responses; has integrated over 230 dialogues skills that help the chatbot generate diverse and meaningful interactions; has a sequence-to-sequence framework (seq2seq) used in conversation response generation (As part of the neural network) enabling the chatbot to generate appropriate responses even for conversations that the dataset does not cover and uses reinforcement learning to optimize its response generation and improve interaction quality with its users over time. Therefore, XiaoIce serves as an excellent example of a chatbot that effectively integrates emotion recognition in its empathetic computing module.

**2.2.2 IBM Watson Assistant**

IBM Watson Assistant is an AI-powered chatbot platform designed to build conversational interfaces for applications and websites. It offers natural language understanding and sentiment analysis capabilities which help businesses enhance their customer service and satisfaction. According to IBM (n.d.), the customer service environment is crucial to the extent that 91% of the customers would part ways with a specific brand if their experience were unsatisfactory. Watson Assistant is also benefitting from the ongoing advancements of artificial intelligence by IBM’s top AI research and development teams. The assistant continually strives to enhance accuracy while reducing the need for extensive training data.

According to IBM (n.d.) the development and implementation of this chatbot assistant has at its base the use of deep learning, machine learning and natural language processing models. This is in line with the findings of Mikolov et.al. (2013) who suggested that to improve natural language processing, the use of word embeddings would be beneficial and with the study carried out by Radziwill and Benton (2017) which articulates the importance of integrating machine learning approaches to make chatbots more adaptive to different input styles and tasks.

Furthermore, one crucial technique to mention used by this software is intent classification, which relies on machine learning models to understand and infer users’ intentions based on their input. (Jurafsky and Martin, 2023, pp.296-316; Chen et.al., 2018) This approach is argued to be essential for developing effective conversational agents that can answer adequately to user inquiries. These machine-learning models are used for intent classification and emotion recognition and the use of machine learning is supported by research in the field of text-based emotion prediction, as per the study published by Alm, Roth and Sproat (2005). This research helps AI systems such as Watson Assistant understand and respond to the emotional states of users for the purpose of achieving more empathetic and emotionally aware interactions.

Lastly, as mentioned by IBM (n.d.), the AI-powered chatbot platform can also recognize and process entities through its entity detection models. Entities refers to the ability to recognize and identify specific elements within textual data, plain language such as synonyms, enabling better understanding of the context of user interactions. With this capability, the chatbot can identify emotional cues in text, leading to an improved and more empathetic conversational AI.

**2.3 Sentiment Analysis Techniques in Chatbots**

**2.3.1 NLP (Natural Language Processing)**

Natural Language Processing (NLP) has become an essential field in computer science and artificial intelligence (York University, n.d.), aiming to enable human-computer communication through natural language understanding and generation. The core objective of NLP is to create algorithms and models that can process, interpret, and generate human language (Jurafsky & Martin, 2019). In recent years, there has been a significant growth in NLP research due to the availability of vast amounts of digital text and development of powerful machine learning techniques.

Early NLP systems were based on rule-based and statistical methods (Manning & Schütze, 1999). These approaches relied on manually crafted rules and features, often requiring extensive linguistic knowledge. However, these methods faced challenges in scalability and adaptability to new domains or languages.

The emergence of machine learning, particularly deep learning techniques, has revolutionized the NLP landscape. Neural networks, a class of machine learning models, have shown a big success in a wide range of NLP tasks, such as machine translation, sentiment analysis, and question-answering systems (Goldberg, 2017). The key advantage of these methods is their ability to automatically learn patterns and representations from large amounts of data, thus reducing the need for manual feature engineering.

One influential development in NLP has been the introduction of word embeddings, such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), which represents words as continuous vectors in a high-dimensional space. These embeddings have been shown to capture semantic and syntactic relationships between words, enabling more effective NLP models.

Another significant advancement is the adoption of transformer-based architectures, such as BERT (Devlin et al., 2018), which has set new performance benchmarks in various NLP tasks. The model employs self-attention mechanisms to capture long-range dependencies and contextual information within text, leading to improved language understanding.

Most of the techniques and models discussed above are part of the development and implementation of the assigned piece of software for this research study. By the use of pre-trained word embeddings, such as GloVe, the model can effectively capture semantic and syntactic relationships between words, leading to more accurate language understanding. Additionally, the chatbot uses a combination of bidirectional LSTMs and transformer-based architectures to capture long-range dependencies and contextual information within text. Through this modern NLP methods, the chatbot demonstrates improved performance in sentiment analysis.

**2.3.2 Deep learning**

Deep learning, a subset of machine learning, has emerged as a ground-breaking approach in artificial intelligence, empowering a wide range of applications, including computer vision, natural language processing, and speech recognition. Deep learning methods primarily involve artificial neural networks with multiple layers, enabling the learning of complex patterns and representations from vast amounts of data (Lecun et al., 2015).

The initial development of deep learning dates around the 1940s along with an early neural network model. However, deep learning did not gain significant attention until the 21st century, when advances in computational power and the availability of large datasets enabled more effective training of deep neural networks (Hinton et al., 2006).

Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) cells, have been incorporated into the chatbot model to effectively handle sequence-based tasks and capture long-range dependencies, demonstrating the advantages of applying advanced deep learning techniques.

According to McTear (2016, pp. 179-180) deep learning methods have become more effective techniques to use, as previously other machine learning methods were more popular due to the hardware that was not yet at a point where deep learning could be effective, mostly because of the low data storage possibilities (Schmidhuber, 2015). Deep learning is now the preferred approach in areas such as NLP, SLU (Spoken Language Understanding) and image processing. Therefore, employing this method as part of the implementation of the sentiment chatbot goes in line with the current methodology, as explained above, ensuring a robust and accurate system for analysing user sentiment.

**2.3.3 Sentiment analysis**

Sentiment analysis is an essential task in natural language processing (NLP) that aims to identify and categorize emotions expressed in textual data (Pang & Lee, 2008). This technique has become increasingly important due to the massive growth of user-generated content on social media platforms, forums but also because of the need for chatbots able to detect emotions that can be used in customer service and mental health.

As stated by Ravi et al. (2015), there are a few preliminary steps for sentiment analysis, such as:

* Data acquisition: this involves collecting relevant textual data from various sources like social media platforms and online reviews.
* Pre-processing: there are a few methods such as tokenization, stop word removal, lowercasing and lemmatization. Tokenization involves decomposing a sentence into its constituent words, phrases, symbols, or other significant elements by eliminating punction marks. During the pre-processing step, stop words, which do not provide meaningful information for analysis, are typically removed. It’s better for the text to be lowercased to maintain consistency and avoid duplicate entries. Lemmatization is used to reduce the words to their root form to minimize redundancy and improve analysis efficiency.
* Feature extraction: The pre-processed text is transformed into a structured format that can be used as input for machine learning algorithms.

Following the pre-processing of data, several important steps are to be undertaken to build an effective sentiment analysis system:

* Feature Selection: After pre-processing, it is crucial to select the most relevant features from text to improve the model’s efficiency and reduce computational complexity. (Ravi et al., 2015)
* Model Training: Once the features have been selected, a machine learning or deep learning model is trained on the dataset.
* Model Evaluation: The performance of the trained model is evaluated using metrics such as accuracy, precision, recall and F1-score. This step involves comparing the model’s predictions against actual sentiment labels to assess its effectiveness (Pang et al., 2002).
* Model Optimization: Based on the evaluation results, the model may be fine-tuned through hyperparameters tuning or by employing techniques like cross-validation to optimize its performance (Kohavi, 1995).
* Deployment: Lastly, after achieving satisfactory results, the sentiment analysis model can be integrated into real-world applications such as chatbots, customer service platforms, or social media monitoring systems to provide insights on user sentiment.

**2.4 Summary**

The literature research enabled a thorough examination of the many factors that support the development and operation of chatbots such as XiaoIce and IBM Watson Assistant. The systems mentioned, have demonstrated the potential of chatbots in a variety of contexts. These chatbots not only have conversations with users, but they also learn from these conversations, indicating a big promise in industries such as: customer service and mental health.

Moreover, the second part of the literature review covers in detail the techniques to consider when building an emotion detection model. This chapter introduces the notions of deep learning, NLP, and sentiment analysis, in order to give a better insight on the project, and the techniques used to train and understand the model.

**Chapter 3: Design and Implementation**

**3.1 Design**

**3.1.1 Introduction**

This chapter presents the design of the sentiment chatbot with capabilities to detect user sentiment. The design chapter is structured to provide a comprehensive understanding of the various components and processes involved in building the chatbot, from data pre-processing to model training and how the chatbot manages the input. This chapter presents various components, such as the UI of the chatbot, the classes used, with methods and variables, and the data flow throughout the software.

**3.1.2 UI**

**Start Window**

A screenshot of a chatbot

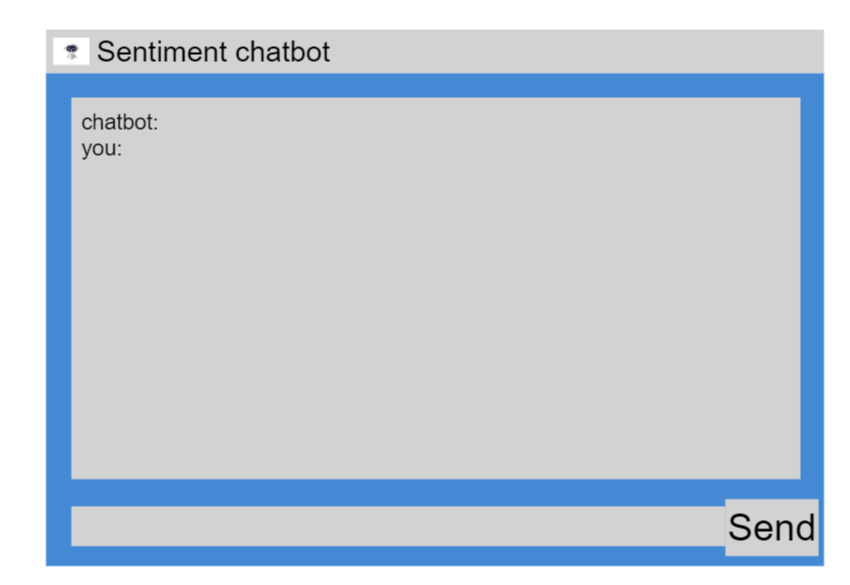
Description automatically generated with medium confidence

**Figure 1: StartWindow UI**

The StartInterface class defines the start window of the application. It includes a logo at the top, a title, a short description and a “Start a New Chat” button. The UI elements in this window as seen in **Figure 1** are:

* **Logo**: This is displayed at the top of the window. The logo is loaded from a file called “logo.png” and placed in the centre of the logo frame.
* **Title**: Below the logo, there’s a label that displays the title “Sentiment Chatbot”. The font is Arial, size 24, and bold. The text color is “#61AFEF”.
* **Description**: Below the title there’s another label that displays a short description. The text wraps at 450 pixels, the font is arial, size 12 and the text color is “#ABB2BF”.
* **Start Chat Button**: The last element in the start window is a button that starts a new chat when clicked. The button text is “Start a New Chat”, and the button uses a custom style that includes font arial, size 18 and the color is “#61AFEF”.

**Chat Window**

****

**Figure 2: ChatBot UI**

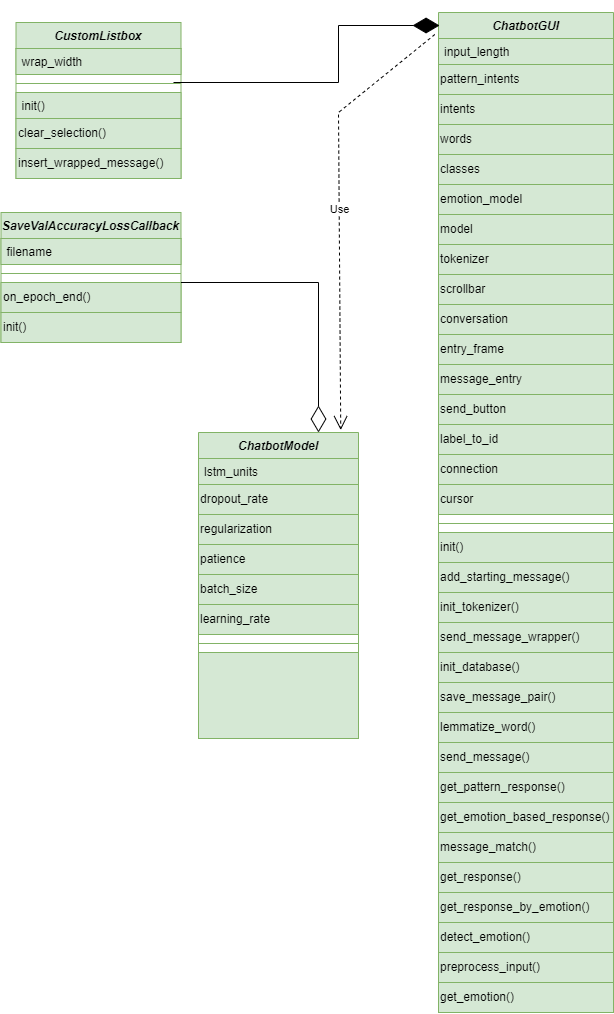
The ChatbotGUI class defines the chat window of the application. This window appears when you click on the “Start a New Chat” button in the start window. The chat window has a conversation area and a message entry area. The UI elements in this window as seen in **Figure 2** are:

* **Conversation Area:** This is a custom ListBox widget that displays the conversation. Messages are added to the end of the listbox, and the listbox automatically scrolls to the end when a new message is added. The background color of the conversation area is “3C3F58”, the text color is “#ABB2BF”, and the font is arial, size 12.
* **Scrollbar:** A scrollbar is located at the right side of the conversation area, allowing users to scroll through the conversation. The scrollbar is linked to the conversation listbox which allows it to control the visible portion of the listbox content.
* **Message Entry Area:** Located below the conversation area, the message entry area is where the users can type their messages. It’s an Entry widget with a width of 60, the font is Arial, and the size is 12.
* **Send Button:** Next to the message entry area, there’s a Send button. When clicked or when the user presses the “Enter” key, it triggers the send\_message function. The button’s background color is “#61AFEF”, the text color is “#282C34”, and the font is Arial, size 12.

The overall theme of the UI is dark, with a background color of “#282C34” for both the StartInterface and ChatbotGUI windows. The color scheme is consistent throughout the interface, providing a good user experience.

**3.1.3 Classes**

The chatbot system consists of four primary classes: CustomListBox, ChabotGUI, SaveValAccuracyLossCallback, and ChatbotModel, for an illustration see **Figure 3** below. These classes work together to create a functional chatbot with a user-friendly interface, natural language processing capabilities, emotion detection based on Paul Ekman’s model with 6 emotions, and the ability to save and analyse its performance.



**Figure 3: Class Diagram**

Firstly, the CustomListBox class is introduced. The purpose of this class is to improve the chatbot’s user interface and this is achieved through extending the standard Listbox widget from Tkinter library. It includes the “wrap\_width” attribute and the following methods “\_\_init\_\_()” , “clear\_selection()”, and “insert\_wrapped\_message()”. The purpose of the attribute is to control the width at which the text wraps within the listbox, while the methods ensure proper formatting and display of messages.

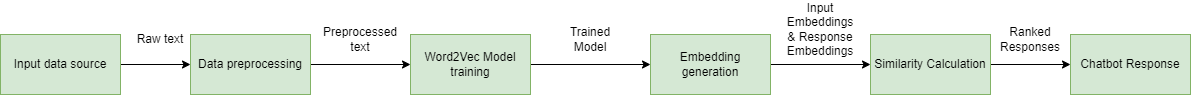
The ChatbotGUI class is responsible for creating and managing the chatbot’s graphical user interface. As per **Figure 3,** this class consists of numerous attributes used to store various components related to chatbot’s interface, conversation history, and natural language processing capabilities. The ChatbotGUI class value is also illustrated by its numerous methods which facilitate user interaction, message processing, and database management.

SaveValAccuracyLossCallback is a utility class that works with the Keras library to monitor and save the chatbot model’s performance metrics during training. It has a “filename” attribute and two methods “\_\_init\_\_()” and “on\_epoch\_end()”. The “filename” attribute specifies the file in which the performance metrics will be saved, while the methods enable the callback to work with the Keras training process.

The ChatbotModel class is not a typical class, as it is defined within the main section of the code and does not have any methods. However, it includes several characteristics relating to the model’s configuration, such as “lstm\_units”, “dropout\_rate”, “regularization”, “patience”, “batch\_size” and “learning rate”. These attributes allow for the customization of the chatbot model’s architecture and training process.

There are relationships between some of the classes in the chatbot system. The ChatbotGUI class has a composition relationship with the CustomListBox class, meaning that the ChatbotGUI is responsible for managing and controlling the CustomListBox objects. Additionally, there is an aggregation relationship between the SaveValAccuracyLossCallback class and the main section where the ChatbotModel is defined, indicating that the callback object is used during the model’s training process but does not have exclusive ownership over it. There’s also one more relation between the ChatbotGUI and ChatbotModel classes which is a dependency relation. This relation signifies that the ChatbotGUI relies on the ChatbotModel for some of its operations, particularly those related to the sentiment analysis process.

**3.1.4 Data Flow**

**** The chatbot system is designed to handle user input by following a series of steps, starting with the Input data source and finishes with the chatbot’s response. As seen in **Figure 4** These steps can be divided into several primary components: Data preprocessing, Word2Vec model, Embedding generation and Similarity calculation.

**Figure 4: Data Flow Diagram**

Explanation of the steps the chatbot takes in order to give a response to the use:

Input data source: The chatbot begins with the input data source, which consists of the input prompted by the user. The input serves as the foundation for chatbot’s understanding of the user’s emotion. The chatbot has two methods for giving a response back to the user: “get\_response()” and “get\_response\_by\_emotion()”.

Data preprocessing: The raw text is then passed through a preprocessing stage, where it is transformed to become more suitable for processing by the chatbot model. Preprocessing includes tokenization and stop word removal. The chatbot uses methods of the ChatbotGUI class in order to preprocess the data: ”lemmatize\_word()” and “preprocess\_input()”.

Word2Vec Model: After pre-processing, the text data is fed into a Word2Vec model. The Word2Vec is a type of neural network that is trained to learn word embeddings by analysing the context in which words appear within a large chunk of text. The ChatbotModel class represents the Word2Vec used to train the model and generate embeddings. The trained model is then used to generate the emotion based on the user’s input.

Embeddings generation: The Word2Vec model generates embeddings for both the input and potential responses. These embeddings are high-dimensional vector representations of the text that captures semantic meaning. In the ChatbotGUI class there are 2 methods used to create embeddings for input and responses: “detect\_emotion()” and “get\_emotion()”.

Similarity calculation: Once embeddings are generated, the chatbot system calculates the similarity between input embeddings and response embeddings. By ranking the potential responses based on their similarity to the input the chatbot can determine the most appropriate answer.

Chatbot response: The chatbot system selects the response based either on a matched pattern, or if there’s no pattern available, it will try and detect the emotion, and give a response based on that. Once the chatbot has an answer, the “send\_message()” method is called to display the answer for the user.

Therefore, the provided data flow diagram in **Figure 4** along with the above step by step explanations should offer a valuable insight into understanding the flow of data through the system and the interaction among the components.

**3.1.5 Summary**

The design chapter offers a comprehensive understanding of the various components and processes involved in constructing the chatbot. This understanding is aimed to be achieved by the of this chapter through the discussion surrounding the chatbot’s design. This design consists of:

* UI elements: Startwindow and chatbotWindow
* Four main classes: CustomListBox, ChatbotGUI, SaveValAccuracyLossCallback and ChatbotModel
* Data Flow: starting from the input all the way to the chatbot’s response.

**3.2 Implementation**

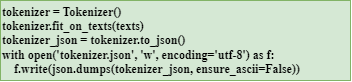
**3.2.1 Introduction**

This section will cover the implementation of the emotion-detecting chatbot. The chatbot is a software application that uses natural language processing (NLP) and deep learning techniques to understand and get the emotion behind user’s input. The chatbot is designed to recognize the 6 basic emotions as per Paul Ekman’s model and return an answer to the user based on that emotion. This section will cover the implementation details, technical achievements as well as challenges encountered during the development process.

**3.2.2 Chatbot Model**

***Data preprocessing***

The first step in building the emotion detection model is preprocessing the data. Firstly, the texts are tokenized using the “Tokenizer” class from Keras library, to convert them into sequences of integers, and pad these sequences to a fixed length. See **Figure 5** for a demonstration on how to create and save the tokenized.



**Figure 5: Data Tokenization**

I started by creating an instance of the “Tokenizer” class that will be responsible for tokenizing the text data. After the “Tokenizer” was instantiated, it is fit on the “texts” variable, which contains the entire dataset. The “fit\_on\_texts()” method analyses the input data and extracts information such as the total number of unique words and the frequency of each word. This information is then used to build a vocabulary that will be used to convert the text data into sequences of integers. After the tokenized was fit on the text, and since it will be used in the ChatbotGUI to preprocess user’s input, I converted it into a JSON format using the “to\_json()” method and saved it to a file named “tokenizer.json” using the “with open()” statement to ensure that the file is properly closed after writing.

***Model Architecture***

The model architecture consists of the following layers (See **Figure 6 below**):

1. Embedding layer: The purpose of this layer is to map the integer-encoded tokens of the input text to dense vectors of fixed size. The dense vectors capture the semantic information of words, enabling the model to learn meaningful representations. By initializing the embedding matrix with the pre-trained word embeddings like GloVe, the model benefits from the rich semantic information encoded in these embeddings, leading an improved performance (Pennington et al., 2014).
2. Bidirectional LSTM layers: As stated in the Literature Review Long-Short-Term-Memory (LSTM) layers are a type of Recurrent Neural Network (RNN) that can capture long-range dependencies in sequences. Bidirectional LSTM’s process the input sequence in both forward and backward directions, enabling the model to learn from both past and future context. This can improve the model’s ability to capture complex relations in the text, leading to more accurate emotion detection (Schuster & Paliwal, 1997).
3. Batch Normalization: This layer normalizes the output of the LSTM layers by centering and scaling the activations. This has the effect of stabilizing the learning process, improving the training speed, and reducing the sensitivity of the initial weights. As a result, the model achieves better performance.
4. TimeDistributed Dense layer: A TimeDistributed Dense layer applies a Dense (fully connected) layer independently to each time step of the LSTM output. This allows the model to learn complex relationships between the input features and the target emotions, which may be crucial for accurate emotion detection.
5. GlobalMaxPooling1D: This layer is used to reduce the dimensionality of the output from the TimeDistributed Dense layer. It selects the maximum value for each feature over the time dimension, effectively summarizing the most important information in the LSTM output. This simplification helps the model focus on most important features for emotion classification.
6. Dropout layer: The Dropout layer is a regularization technique that helps prevent overfitting. During training, it randomly drops a portion of the input units, which forces the model to learn more robust representations. This reduces the model’s reliance on specific input features and helps it generalize better to unseen data.
7. Dense layer with softmax activation: The final layer of the model is a Dense layer with softmax activation. The softmax function converts the output of the Dense layer into a probability distribution over the emotion classes, which allows the model to predict the most likely emotion for a given input text.

In summary, the chosen model architecture combines the strengths of various layers and techniques to improve the model’s ability to learn complex relationships in the input text, but also helps the model generalize to new data, and achieve accurate emotion detection.

***A screenshot of a computer code

Description automatically generated with low confidence***

**Figure 6: Model Architecture**

***Configuring and training the Emotion Detection model***

In this section, I will present the configuration and training of the emotion detection model by setting the loss function, optimizer, evaluation metrics, and defining training callbacks.

As seen in **Figure 7**, I start by configuring the model for training by setting the loss function, optimizer, and evaluation metrics. I will start by introducing the variables used:

1. Adam Optimizer: Adam stands for Adaptive Moment Estimation and is a popular algorithm for training deep learning models. It combines the advantages of two other popular optimization methods, AdaGrad and RMSProp (Kingma & Ba, 2014). Adam adapts the learning rate for each parameter individually, which can lead to improved performance. It has been shown to work well in various deep learning tasks and requires less tuning of hyperparameters compared to other optimization methods, making it a good choice for this problem (Reddi, Kale & Kumar, 2019).
2. Learning rate: The learning rate (0.001) is a crucial hyperparameter that influences the performance of the model. The learning rate of 0.001 has been found to be a reasonable default for many deep learning tasks, including natural language processing and it works well with the Adam optimizer (Kingma & Ba, 2014).
3. Loss function: Categorical crossentropy is a suitable loss function for multi-class classification problems (Demirkaya, Chen, Oymak, n.d.). It measures the difference between the expected and real probability distributions of the target classes.
4. Evaluation metrics: I used accuracy, as it is a commonly used evaluation metrics for classification tasks (Sokolova & Lapalme, 2009). It measures the number of correct predictions out of the total number of predictions made. In the context of emotion detection, accuracy can help measure the model’s overall performance in classifying emotions correctly (Binali & Potdar, 2012).

After the model was compiled using the variables mentioned above, I defined the training callbacks. I used the “EarlyStopping” callback to stop the training process when the validation loss stops improving with a “patience” of 5 epochs. I used the “restore\_best\_weights” parameter to ensure that the best weights are loaded back into the model at the end of the training process. The “ModelCheckpoint” callback saves the best model weights to a file named “Chat\_bot.h5” and it is measured based on the validation loss. Lastly the “ReduceLROnPlateau” callback reduces the learning rate by 0.1 if the validation loss stops improving for 3 epochs, which allows for improved generalization during the training.

Finally, I trained the model using the “fit()” method, providing the training data, batch size, number of epochs, validation data, and callbacks. The model’s training progress is recorded in the “history” variable, which can be used for analysing the training and validation performance.

A screenshot of a computer program

Description automatically generated with low confidence

**Figure 7: Configure and training**

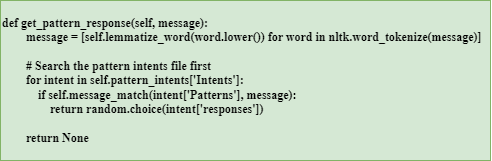
**3.2.3 GUI**

In this section I will describe the implementation of the emotion-detecting chatbot based on the model explained above. The chatbot interacts with users through a graphical user interface (GUI) and provides responses based on the detected emotion of the input message.

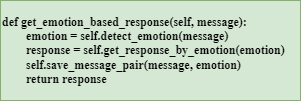
I started by designing the chatbot’s GUI using the “tkinter” library, which provides a simple and intuitive way to create a graphical application in Python. The interface consists of four main elements, see **Figure 2**.

After creating the GUI, I created the “CustomListbox” class that inherits from the “tk.Listbox” class, in order to handle the text wrapping for the messages, and make sure they appear correctly within the conversation window.

The ChatbotGUI class manages the chatbot’s GUI by including the four mentioned elements to the frame and allows the user to send messages by clicking the send button or by pressing “Enter”. When the send button is pressed, the send\_message method processes the user’s input and returns an appropriate answer. The chatbot first checks for a pattern-based response, see **Figure 8**, for the generic questions like greetings, or other inquiries, and if a pattern is found, the chatbot will print the answer to the screen. Otherwise, if no pattern is found in the user’s message, it will use the pre-trained model to detect the emotion behind the user’s message, and give a response based on that emotion, see **Figure 9**.



**Figure 8: Pattern Based Response**



**Figure 9: Emotion Based Response**

The two methods mentioned above: **Figure 8** & **Figure 9** also use helper methods such as: message\_match and get\_response\_by\_emotion.

The message\_match method is responsible for searching the patterns file and returning the answer if a pattern is matched in the user’s input, while the get\_response\_by\_emotion returns an answer based on the emotion detected by the model from the user’s input.

**3.2.4 Problems encountered during the development of the chatbot.**

One of the first problems I encountered during the development process was while building the emotion-detection model. My first model had only one LSTM layer which resulted in a relatively low accuracy, after experimenting with the hyperparameters, and because I did not see any improvements in terms of emotion-detection accuracy, I switched to two Bidirectional LSTM for better understanding.

During the development of the neural network, I had to try different combinations of hyperparameters before getting it right.

Firstly, I downloaded an open-source dataset for educational purposes, that had over 300000 entries for the Paul Ekman’s 6 emotion model, my mistake here was that I assumed the labels would be correct which led me to ten hours of waiting for the model to be trained. After training the model I realized that even though the valuation accuracy was at 95%, the chatbot kept getting all the emotions wrong. After realizing the labels were all wrong I started building my own dataset, that’s when I started seeing results, even though the accuracy was now 93% due to the significantly lower number (around 4000-4500) of entries in the dataset, the chatbot started to recognize emotions.

I also encountered some problems while working on the GUI. I was trying to enable the user to send the message by pressing “Enter”, and even though I used the correct approach when declaring the interface, the problem was not solved until I created the send\_message\_wrapper method, which allowed the user to send the messages by pressing “Enter”.

Initially, the chatbot was supposed to generate its own answers based on the user’s input. However, the current implementation uses a pattern-based architecture, due to the complexity of the dataset required to train the model to answer the question by itself. I tried following the initial approach but, due to the low number of entries in the training dataset, I could only get around 10% accuracy, which would not generate contextually relevant answers.

Initially the chatbot was supposed to generate its own answers based on the user’s input.

**3.2.5 Summary**

The implementation chapter discussed the development process of the chatbot interface as well as the development of the neural network. This chapter describes the steps of building the emotion detection model and motivates the choices for the systems architecture (layers), as well as the hyperparameters used for the training. Lastly the chapter ends with a section describing the problems encountered and the approach on tackling them.

**Chapter 4: Testing and evaluation**

**4.1 Introduction**

The primary aim of this chapter is to analyse and evaluate the performance of the emotion detection model. The evaluation involves an assessment of the model’s accuracy, precision, recall, F1 score, and loss during both training and validation stages. Furthermore, I conducted beta testing and gathered user’s feedback through a questionnaire to measure user satisfaction and identify areas for improvement.

**4.2 Model performance evaluation**

As mentioned in the Introduction for the evaluation of the model I used several metrics such as:

* Accuracy is the most often used method to measure the proportion of total predictions that are correct (Sokolova & Lapalme, 2009). The model achieved an accuracy of 0.9874 (98.7%) during training, while during validation it achieved an accuracy of 0.9305 (93%). These values suggest that the chatbot is capable to correctly identify the emotions expressed by the user.
* Precision to calculate the number of true positive predictions out of all positive predictions made. The model achieved a 0.9968 precision during training and 0.9188 during the validation stage. The precision being relatively high leads to less misinterpretations. Misinterpretations can lead to inappropriate responses from the chatbot, which can frustrate the users or even lead to them not using the software again.
* Recall (sensitivity) to measure the proportion of actual positives that are correctly identified. The recall values for the training and validation were 0.9967 and 0.9174. A high recall ensures that the chatbot can identify user sentiments effectively.

I also made a plot using the validation Accuracy and validation Loss. I used the data saved by the model throughout the entire training process. As can be seen from **Figure 10**, the validation accuracy starts at 0.663, which means that at the early stages of training the model was able to classify approximately 66% of the emotions in the validation set. As the training progresses, the validation accuracy increases up to 93% in the 7th epoch, which means that the model was effectively learning from the data.

The validation loss starts at a relatively high value of 20.39, but drops significantly over the epochs, reaching a value of 0.81towards the end of the training. This is a good indicator of learning, as lower loss values suggest that the model’s predictions are getting closer to the true labels in the validation set.

It’s also notable, that the validation accuracy stops growing and the validation loss starts to slightly increase towards the end of the training, which is a sign of overfitting, where the model is becoming to specialized on the training data and performs less well on unseen data. For this reason, I implemented the early stopping which stops the training process if the validation loss does not improve for 5 epochs.

A picture containing text, screenshot, plot, diagram

Description automatically generated

**Figure 10: Validation Accuracy and Loss**

**4.3 Beta Testing and User Feedback**

In order to gather feedback from users and further assess the performance of the chatbot, a beta testing was conducted. Each of the participants were given a compiled version of the code with the extension .exe, a small briefing about the project and how to use it, see **Appendix 1**, and a questionnaire containing ten questions especially designed to gather data on important aspects of the usability and overall performance of the software (see **Appendix 2**). Moreover, these questions are aimed at evaluating the user’s experience and the chatbot’s functionality, emotion detection capabilities, and overall design.

Beta testing is an integral part of the software development process, typically conducted during the final stages before the product’s release. It involves distributing the software to a select group of end-users who use it in real world scenarios and provide feedback ()

A total of 7 participants took part in the beta testing. The answers to each question are summarized below:

1. All participants (100%) felt they were given enough introductory information about the software to understand its purpose and test it effectively. This was achieved through the project brief provided to each participant, which thoroughly describes the purpose of the software, example messages to be addressed and different functionalities that they are encouraged to provide feedback on.
2. The number of messages addressed to the chatbot by the participants varied, with values in the range 7 to 20 and the minimum required messages to be addressed was set to five. The choice of five minimum messages was made as to encourage users to express and a have a chance to test at least several emotions, but to also be considerate of their time and effort.
3. The number of correctly emotion-labelled messages was mostly equal or very close to the number of messages addressed by each tester. (e.g., one user sent ten messages and got the right emotion for all of them; other user sent twenty messages and got the right emotion for sixteen; whereas there was one instance where fifteen messages were addressed and only five returned the right emotion as per the user’s feedback)
4. The majority of the participants (85.71%) reported that the chatbot answered clearly, concisely and correctly to greetings, introductions and general questions. However, 14.29% (one user) felt that the chatbot did not meet these criteria.
5. Among all participants, four of them (57.14%) found it “Very easy” to interact with the chatbot, while three participants (42.86%) voted for “Easy”.
6. Participants provided various feedback on the design of the chatbot. Overall, it was agreed that the design was straightforward, easy to understand and interact with. A few participants highlighted the chatbot’s friendly and simplistic design, which they believed was fitting for the purpose of the project. One participant suggested that more work could be done around the chatbot’s appearance to make it more inviting. Another participant mentioned that the chatbot should be able to identify sentiment based on questions or statements more concisely.
7. All participants agreed that the chatbot demonstrated emotion detection capabilities.
8. The overall experience using the software was rated as follows:

* 1 participant (14.29%) rated it as “Below average”, meaning that the software is usable but has noticeable flaws, limitations that negatively impacts the overall experience.
* 2 participants rated it as “Good” meaning that the software is user-friendly, reliable, and offers valuable features even though some minor issues may be present it does not affect the overall experience.
* Lastly, 4 participants rated it as “Excellent”, describing that the software is intuitive and high performing. They felt it consistently exceeded expectations, and any issues or limitations were minimal and easily outweighed by its benefits.

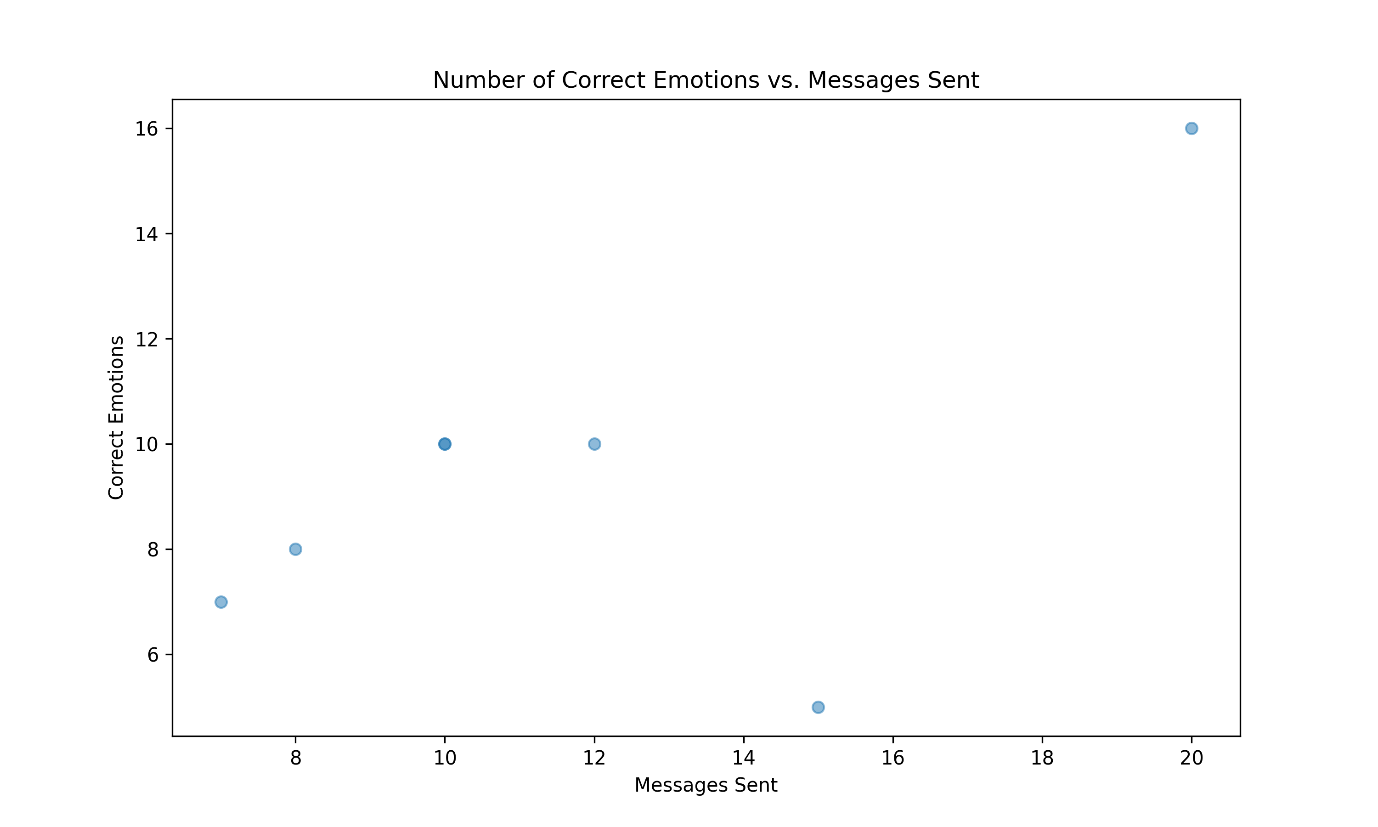
1. Six participants felt that the software provided for examination, aligns with the anticipated outcomes delineated in the initial briefing, while one participant did not.
   1. **Analysis and Justification of user feedback**

The purpose of this section is to provide justifications and a short but concise analysis of feedback results that were not covered in detail in the previous section. The questions to be comprehensively discussed in this part are mainly Q3 & Q4 (very similar to one another so will discuss them together) and Q6.

*Questions 3&4:*

These questions asked participants to 1. Answer how many messages addressed were correctly emotion-labelled and 2. Answer whether the chatbot answered clearly, concisely and correctly to greetings, introductions and general questions. The feedback gathered on these two questions can be attributed to a variety of factors. Most users reported a high success rate of emotion-labelling suggesting that the system’s emotion-detection model generally performed well, with most users getting accurate matches between their messages and the chatbot’s response. However, some discrepancies were reported which might be due to the limitations of the software, the inherent complexity of emotion-detection in text, or possible user misunderstanding. For example, the context-dependant nature of emotions might have been responsible for a few mismatches of emotion labelling. (Calvo and D’Mello, 2010)

Regarding the feedback on Question 4, it is plausible that besides the limitations of the software and the context-dependent nature, the user might not have fully understood or familiarized themselves with the brief provided, thus leading to a mismatch between their expectations and the chatbot’s capabilities. Given that for introductions, greetings and general questions the chatbot operates on a pattern-based response system, its response is limited to programmed patterns, which were outlined in the user brief. If the user did not review the brief considerably, they may have started interactions outside the chatbot’s programmed scope, leading to perceived shortcomings. It is therefore essential to provide clear instructions and manage user expectations when conducting beta testing. The following **Figure 11** shows an illustration of the actual results of the feedback regarding the total number of messages addressed and the total number of correctly labelled messages.



**Figure 11: Number of Correct Emotions vs. Messages sent**

*Question 6:*

The feedback on this question offers valuable information about the chatbot’s design. The overall feedback indicates an appreciation for the simplicity and user-friendly interface of the chatbot, which are aspects identified that need improvements. One aspect is specifically targeted at the aesthetic part of the software, with a participant suggesting a more inviting design. This feedback could suggest that this aspect plays a significant role in user engagement and satisfaction, enhancing the overall user experience. The feedback that needs to be addressed is from one participant that highlighted that the chatbot should be more concise in its identification of sentiment in questions or statements. This suggests that while the chatbot’s emotion recognition was generally successful, its capability to precisely process normal and complex user inputs needs further improvements to enhance its effectiveness and accuracy. The importance of the feedback that suggested improvements and/or disagreed with the chatbot’s conciseness is illustrated by Myers (1979, p. 15) through his statement ‘Do not plan a testing effort under the tacit assumption that no errors will be found’.

**Chapter 5: Conclusion**

**5.0 Introduction**

Lastly, this chapter will cover the concluding aspects of the project. Firstly, the aim and objectives will be revisited to evaluate the completion of the project and its success. Secondly, there will be a dive into the achievement and successes of the project, followed by the challenges faced. The project will conclude with final thoughts on the whole project.

**5.1 Revisiting Aim and Objectives**

To judge whether the primary aim was achieved, the objectives set for the project need to be looked into as a whole and particularly.

Firstly, to work towards the aim of developing and implementing an emotion-detecting chatbot, the objective of conducting an extensive literature review on the existing techniques and approaches, as well as existing similar software. This was successfully achieved through the literature review chapter which discusses comprehensively each of these aspects.

Secondly, the next objective was examining different algorithms and techniques to identify the most suitable for this project’s topic, which was again conducted in the literature review and closely related to the software developed.

A chatbot architecture was successfully designed, presented, and explained in the implementation section as per one of the objectives set in the introduction.

Moreover, another objective was to implement the chatbot using an appropriate programming language, libraries and tools, process which was explored and clearly described, presented as part of implementation. This objective was not fully achieved, as it was meant to achieve a more successful chatbot with more complex capabilities, part which was explained through the discussion on limitations and challenges faced.

The chatbot’s effectiveness in detecting emotion was mainly presented and proven in the testing and evaluation section, however this objective is meant to be illustrated in the dissertation. The success in detecting emotion is highlighted through the beta-testing process as well as the model performance evaluation and given the results in both processes, it can be stated that this objective was achieved.

Lastly, future developments and improvements were identified in the limitations and challenges sections where it was explained why some planned techniques, goals and applications could not be successfully implemented and what the actual intentions behind the project were. Improvements were also gathered from the participants’ feedback in the beta test.

Therefore, given the success and achievement of most objectives, it can be concluded that the aim of developing a sensitive chatbot with capabilities of detecting emotion was achieved, however the initial standards before the development began where higher than the actual results, as explained throughout the project.

**5.2 Achievements and successes**

The primary objective of this dissertation was to enhance the emotional understanding of chatbots by using deep learning techniques, sentiment analysis and Natural Language processing. This objective was accomplished as evidenced by the performance metrics of the developed chatbot.

The chatbot demonstrated good performance in emotion recognition, achieving an accuracy rate of 93% during validation. The precision rate was also high reaching up to 91.88% during validation, which indicated a low rate of false positives. The recall values, which measured the ability of the chatbot to correctly identify user sentiments were 91.74% during validation.

Furthermore, the chatbot was successfully beta tested, which allowed to collect important feedback which will surely help in future work to improve the abilities of the chatbot as well as its graphical interface.

**5.3 Challenges**

The development process of the chatbot was not without its challenges. One of the initial challenges encountered was increasing the accuracy of the emotion-detection model. It was not until the integration of two bidirectional LSTM layers that we noticed significant improvements in accuracy. Moreover, the process of hyperparameter tunning during development of the model, required numerous tries iterations to attain optimal performance.

A notable mistake during the early stages of development was the reliance on a large open-source dataset that, despite its large size, contained inaccurately labelled data. This error led to a false sense of achievement as the high training accuracy, did not translate into effective emotion recognition in practice.

**5.5 Final Thoughts**

This project has been a real learning experience in using artificial intelligence to create an emotion detection model and integrate it to the chatbot’s interface. Despite some challenges along the way, in the end the project was a success.

**BCS Project Criteria & Self Reflection**

* An ability to apply practical and analytical skills gained during the degree programme: The development and implementation of the chatbot required the application of the practical and analytical skills acquired throughout my degree. These skills were illustrated through the creation of the model and implementation of the chatbot’s GUI, as well as in the testing and model evaluation part, which can be seen in sections 3.1 and 3.2.
* Innovation and/or creativity. These were demonstrated as part of the whole development, design, implementation and testing sections of the dissertation as all of them required putting ideas together and coming up with solutions to allow continuation of software development. More specifically, they were demonstrated in the application of deep learning techniques, sentiment analysis and Natural Language Processing, discussed in detail in section 3.2.
* Synthesis of information, ideas, and practices to provide a quality solution together with an evaluation of that solution. This is proven through the use of various ideas and sources to develop a solution. The evaluation of the solution, in terms of the chatbot’s performance, is provided in section 4.2 and many of the sources and ideas synthesized are presented in the literature review and cited throughout the various sections.
* That your project meets a real need in a wider context. This is proven especially in the literature review and introduction (1.2) discussion, but also throughout the dissertation because information stated is always backed up by academic sources which mostly highlight the need in the wider-context of human-computer interaction.

**An ability to self-manage a significant piece of work & Critical self-evaluation of the process.**

Throughout the completion of this project, as with any dissertation or project of this size and complexity, one of the most important learnings I achieved was self-management. The development of this project, a chatbot with capabilities to detect user sentiment, required a significant amount of planning, effort and execution, considering that it is to be completed in the same time frame as the rest of the course work.

The common knowledge to self-manage is to set realistic goals and timelines in order to keep the project on track. This is something that I tried to be consistent with throughout the whole project. What helped me the most was breaking down larger tasks into smaller, manageable parts as it kept me focused and gave me a feeling of accomplishment which motivated me to keep going. Each week I also allocated extra time for unexpected challenges that I knew I would get stuck in as it is often the case with most projects, especially in the field of computer science. This was particularly helpful and used during the model development phase. I was mostly never able to get through this part in the allocated time in the beginning, so understanding this is a part that requires exploring possible solutions in more depth and giving myself time to research was very helpful in keeping myself on track.

However, I can clearly state that there were weeks of frustration, especially when the initial model’s performance wasn’t nearly what I wanted it to be and when the large open-source dataset proved to be inaccurately labelled. This helped me recognize the value of patience and resilience. These challenges prompted me to reconsider my approaches and seek alternative solutions, which eventually led to the successful development of the chatbot.

Moreover, there were areas where I could have improved. For example, paying more attention to the initial open-source dataset could have saved me a significant amount of time which could have been best dedicated to further expanding the current dataset or working on a better neural network.

Overall, this project represents a significant milestone in my academic journey, being both a personal and technical challenge. It taught me the importance of perseverance, self-management, critical thinking and how to adapt in order to face obstacles. The knowledge and skills gained through this project will for sure serve me well in my future career and I consider it a great opportunity and experience to have.

**Appendix**

**Appendix 1: Briefing**

***Sentiment chatbot***

**Brief Explanation for Beta Testers**

This software is designed to detect user sentiment based on Paul Ekman's 6 basic emotions (joy, sadness, anger, fear, surprise, and disgust) as well as understand and respond to specific conversational patterns. As a beta tester, your feedback is invaluable in helping me improve and perfect this chatbot experience.

Important aspects to consider when using the chatbot:

* The main purpose of the software is to detect emotion in text, so keep in mind when addressing it, it's most effective to use messages that convey emotions.
* In some cases, it can be case sensitive, so if you are not getting the answer, try to use the correct form of the sentence. (e.g., ‘or capital letters)
* it has pattern-based responses so feel free to ask generalist questions to find more about it and test its performance. (Information in the quick guide)

Here's a quick guide on how to interact with the chatbot:

* Start by saying "Hello" to initiate a conversation.
* You can ask questions like "What's your name?" or "How are you?" to engage in small talk.
* To learn more about the chatbot, try asking "Tell me about you," "How were you created?" or "When were you created?"
* If you're unsure what to ask, simply type "What should I ask?"
* For a light-hearted experience, you can request a joke with "Tell me a joke."
* To discover the chatbot's knowledge on hobbies, ask "What are some hobbies?"
* If you need guidance on using the chatbot, type "How to use."
* Feel free to express gratitude with "Thank you" or provide positive feedback like "You're doing a great job."

Examples of messages you can send: “Yesterday was quite a letdown and I am working through it”,” Having to walk through dark alleys makes me uncomfortable”,” Completing my degree feels unbelievable”,” I am upset”, “I need help”.

As a beta tester, any feedback on the chatbot's performance or suggestions for improvement is valuable. Please report any issues, bugs, or areas where the chatbot's responses could be enhanced.

Thank you for your contribution.

Best regards,

Sebastian Constantin

**Appendix 2: Questionnaire**

Questionnaire: <https://www.surveymonkey.co.uk/r/YK3GNVT>

Q1: Do you feel like you were given enough introductory information about the software to understand its purpose and allow you to test it?

Q2: How many messages/questions did you address the chatbot? (Minimum of 5)

Q3: Out of the total messages sent, how many were correctly emotion labelled?

Q4: Did the chatbot answer clearly, concisely, and correctly to greetings, introductions, and general questions?

Q5: How easy was it to interact with the chatbot?

Q6: In a few words, can you describe your opinion on the design and if necessary, what could be improved?

Q7: Do you agree/disagree that the chatbot shows emotion detection capabilities?

Q8: Lastly what’s your overall experience using the software? (Rating scale:1-5)

Q9: Does the software provided for examination align with the anticipated outcomes delineated in the initial briefing?

Q10: (Optional) Additional general feedback about the software.

**Appendix 3: Questionnaire responses**

Q1: 100% yes; 0% no

Q2: 10, 20, 15, 8, 10, 12, 7

Q3: 10, 16, 5, 8; 10, 10, 7

Q4: 85.71% yes; 14.29%

Q5: Very easy 57.14%; Easy 42.86%

Q6: “Great”,” the design is simplistic, however simplicity is fitting well with the purpose of the project, it is straightforward and easy to use and I think it contains all you need from a chatbot, including a pop up message at the start”,” The chatbot should be capable to at least identify the sentiment based on a question or a statement but it doesn't do that everytime.”, “The design is friendly and easy to use”, “The design is easy to understand. Nothing should be improved”, “Straightforward, easy to understand and interact with”, “The design is clear, provides easy access to understand how to interact with the software, however, more work could be done around its appearance to make it more inviting”.

Q7: 100% Agree; 0 % Disagree.

Q8: 14.29% Below Average; 28.57% Good; 57.14% Excellent

Q9: 85.71% Yes; 14.29% No

Q10: Optional

**References:**

[1] Shawar, B. A., & Atwell, E. (2007). Chatbots: Are they really useful? LDV Forum, 22(1), 29-49. ([**https://eprints.whiterose.ac.uk/81655/**](https://eprints.whiterose.ac.uk/81655/))

[2] Ghandeharioun, A. *et al.* (2019) ‘EMMA: An Emotion-Aware Wellbeing Chatbot’, in *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, pp. 1–7. Available at: <https://ieeexplore.ieee.org/document/8925455>

[3] Liu, B. (2012) ‘Sentiment Analysis and Opinion Mining’, Synthesis lectures on human language technologies, 5(1), pp. 1–167. Available at: https://doi.org/10.2200/S00416ED1V01Y201204HLT016.

[4] Cambria, E. and White, B. (2014) ‘Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]’, IEEE computational intelligence magazine, 9(2), pp. 48–57. Available at: https://doi.org/10.1109/MCI.2014.2307227.

[5] Cho, K. et al. (2014) ‘Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation’. Available at: <https://doi.org/10.48550/arxiv.1406.1078>.

[6] Lee, D. et al (2017) The chatbot feels you - a counselling service using emotional response generation. IEEE International Conference on Big Data and Smart Computing (BigComp). Pdf. Available from: <https://ieeexplore.ieee.org/abstract/document/7881752/authors#citations>

[7] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9148050/>

[8] [**Marcin Frąckiewicz**](https://ts2.space/en/author/marcin/) **2023** [**https://ts2.space/en/openais-chatbot-and-the-role-of-emotions-in conversations/#:~:text=The%20results%20of%20the%20research,and%20engaging%20experience%20for%20users**](https://ts2.space/en/openais-chatbot-and-the-role-of-emotions-in%20conversations/#:~:text=The%20results%20of%20the%20research,and%20engaging%20experience%20for%20users)**.**

[9] Huang, M. H., & Rust, R. T. (2018). Artificial intelligence in service. Journal of Service Research, 21(2), 155-172.

[10] Calvo, R. A., & D'Mello, S. K. (2010). Affect detection: An interdisciplinary review of models, methods, and their applications. IEEE Transactions on Affective Computing, 1(1), 18-37.

[11] [Prabod Rathnayaka](https://pubmed.ncbi.nlm.nih.gov/?term=Rathnayaka%20P%5BAuthor%5D), [Nishan Mills](https://pubmed.ncbi.nlm.nih.gov/?term=Mills%20N%5BAuthor%5D), [Donna Burnett](https://pubmed.ncbi.nlm.nih.gov/?term=Burnett%20D%5BAuthor%5D), [Daswin De Silva](https://pubmed.ncbi.nlm.nih.gov/?term=De%20Silva%20D%5BAuthor%5D),\* [Damminda Alahakoon](https://pubmed.ncbi.nlm.nih.gov/?term=Alahakoon%20D%5BAuthor%5D), and [Richard Gray](https://pubmed.ncbi.nlm.nih.gov/?term=Gray%20R%5BAuthor%5D) 2022, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9148050/>

[12] Hutto, C. and Gilbert, E. (2014) ‘VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text’, Proceedings of the International AAAI Conference on Web and Social Media, 8(1), pp. 216–225. Available at: <https://doi.org/10.1609/icwsm.v8i1.14550>.

[13] <https://www.ibm.com/downloads/cas/ZDJDQLGV>

[14] <https://online.york.ac.uk/the-role-of-natural-language-processing-in-ai/>

[15] Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing (3rd ed.). Prentice Hall.

<https://web.stanford.edu/~jurafsky/slp3/>

[16] Manning, C. and Schütze, H. (1999) Foundations of Statistical Natural Language Processing. Cambridge: MIT Press.

[17]Goldberg, Y. (2017) ‘Neural Network Methods for Natural Language Processing’, Synthesis lectures on human language technologies, 10(1), pp. 1–309. Available at: <https://doi.org/10.2200/S00762ED1V01Y201703HLT037>.

[18] Mikolov, T. et al. (2013) ‘Efficient Estimation of Word Representations in Vector Space’. Available at: <https://doi.org/10.48550/arxiv.1301.3781>.

[19] <https://aclanthology.org/D14-1162/>

[20] Devlin, J. et al. (2019) ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’, arXiv.org [Preprint].

[21] Lecun, Y., Bengio, Y. and Hinton, G. (2015) ‘Deep learning’, Nature (London), 521(7553), pp. 436–444. Available at: <https://doi.org/10.1038/nature14539>.

[22] Hinton, G.E., Osindero, S. and Teh, Y.-W. (2006) ‘A Fast Learning Algorithm for Deep Belief Nets’, Neural computation, 18(7), pp. 1527–1554. Available at: <https://doi.org/10.1162/neco.2006.18.7.1527>.

[23] Pang, B. and Lee, L. (2009) ‘Opinion mining and sentiment analysis’, Computational Linguistics. CAMBRIDGE: M I T PRESS, pp. 311–312. Available at: <https://doi.org/10.1162/coli.2009.35.2.311>.

[24] Schmidhuber, J. (2015) ‘Deep learning in neural networks: An overview’, Neural networks, 61, pp. 85–117. Available at: <https://doi.org/10.1016/j.neunet.2014.09.003>.

[25] Ravi, K. and Ravi, V. (2015) ‘A survey on opinion mining and sentiment analysis: Tasks, approaches and applications’, Knowledge-based systems, 89, pp. 14–46. Available at: <https://doi.org/10.1016/j.knosys.2015.06.015>.

[26] Pang, B., Lee, L. and Vaithyanathan, S. (2002) ‘Thumbs up? Sentiment Classification using Machine Learning Techniques’, arXiv.org [Preprint].

[27] Kohavi, R. (1995) ‘The power of decision tables’, in Machine Learning: ECML-95. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 174–189. Available at: <https://doi.org/10.1007/3-540-59286-5_57>.

[28] Chen et.al. 2018 A survey on dialogue systems <https://arxiv.org/abs/1711.01731>

[29] Radziwill and Benton 2017 <https://arxiv.org/abs/1704.04579>

[30] Picard n.d. <https://affect.media.mit.edu/pdfs/95.picard.pdf>

[31] Zhou, L. et al (2020) The Design and Implementation of XiaoIce, an Empathetic Social Chatbot. Computational Linguistics. Pdf. Available from: <https://direct.mit.edu/coli/article/46/1/53/93380/The-Design-andImplementation-of-XiaoIce-an?fbclid=IwAR1x0JJUm3FG4G2ANmTg6gpgZW-SW0RzU_TCXO2S86gMILutiNdyvs3Qts>

[32] Alm roth and sproat 2005 <https://aclanthology.org/H05-1073/>

[33] Schuster, M. and Paliwal, K.K. (1997) ‘Bidirectional recurrent neural networks’, IEEE transactions on signal processing, 45(11), pp. 2673–2681. Available at: <https://doi.org/10.1109/78.650093>.

[34] Kingma, D.P., & Ba, J. (2014). Adam: A method for stochastic optimization. <https://arxiv.org/abs/1412.6980>

[35] Reddi, S.J., Kale, S. and Kumar, S. (2019) ‘On the Convergence of Adam and Beyond’. Available at: <https://doi.org/10.48550/arxiv.1904.09237>.

[36] Demirkaya, Chen, Oymak n.d. <https://intra.ece.ucr.edu/~oymak/multiclass.pdf>

[37] Sokolova, M. and Lapalme, G. (2009) ‘A systematic analysis of performance measures for classification tasks’, Information processing & management, 45(4), pp. 427–437. Available at: https://doi.org/10.1016/j.ipm.2009.03.002.

[38] Binali, H. and Potdar, V. (2012) ‘Emotion detection state of the art’, in Proceedings of the CUBE International Information Technology Conference. ACM, pp. 501–507. Available at: <https://doi.org/10.1145/2381716.2381812>.

[39] Myers, G.J. et al. (2004) The Art of Software Testing. Hoboken: John Wiley & Sons, Incorporated. <http://www.51testing.com/N_download/lib/TestingTechDL/ArtofSoftwareTesting.pdf>

[40] Craig, R. D. (Rick D. & Jaskiel, S. P. (2002) *Systematic software testing*. Boston: Artech House. https://web-s-ebscohost com.ezproxy.hope.ac.uk/ehost/ebookviewer/ebook/bmxlYmtfXzEwMDEzNF9fQU41?sid=41a841a5-7316-4eb3-ba1c-5c18a9feec56@redis&vid=0&format=EB&rid=1

[41] Vadan, A.-M. & Miclea, L.-C. (2023) Software Testing Techniques for Improving the Quality of Smart-Home IoT Systems. *Electronics (Basel)*. [Online] 12 (6), 1337

[42] Calvo, R. A. & D’Mello, S. (2010) Affect Detection: An Interdisciplinary Review of Models, Methods, and Their Applications. *IEEE transactions on affective computing*. [Online] 1 (1), 18-37. <https://ieeexplore.ieee.org/document/5520655>

[43] Sokolova, M. and Lapalme, G. (2009) ‘A systematic analysis of performance measures for classification tasks’, Information processing & management, 45(4), pp. 427–437. Available at: https://doi.org/10.1016/j.ipm.2009.03.002.